PRISM: Procedural Reasoning with Iterative Self Monitoring for Multi-Hop RAG

Anonymous ACL submission

Abstract

Multi-hop retrieval-augmented generation (RAG) suffers from two well-documented failure modes: imperfect retrieval saturating around 0.45 recall on standard benchmarks and per-step reasoning errors that cascade through long chains. We propose **PRISM**, a procedural-memory-augmented system that detects and corrects errors at each reasoning step through a Build–Retrieve–Update cycle, accumulating error-correction lessons across questions. We evaluate PRISM against five re-implemented baselines (Vanilla RAG, MEMP, CRAG, ReAgent, HopRAG) under a single unified harness in which all systems share the retriever, LLM endpoint, and prompt-formatting code, so reported gaps reflect algorithmic differences alone. With `google/gemma-4-31B-it`, PRISM achieves F1 of 75.1%, 74.8%, 33.5%, and 19.0% on HotpotQA, 2WikiMultiHopQA, MuSiQue, and MultiHopRAG respectively—best in every cell. Across a 5-round protocol where procedural memory is *retained* between rounds (a downside-defense stress test that exposes only PRISM to memory-induced drift), PRISM is the only system with a non-negative F1–round Pearson correlation on every benchmark. Ablations show error detection contributes up to 18.1 F1 (MuSiQue), and a pooled step-depth analysis ($n \approx 5{,}000$) shows PRISM has the smallest 2-to-3-step relative EM drop (-21.8%) of all six systems—the next-best loses -27.1%.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) mitigates LLM hallucination by grounding generation in external evidence, but multi-hop question answering exposes two persistent weaknesses. First, even strong dense retrievers saturate around 0.45 recall on multi-hop benchmarks (Liu et al., 2025). Second, LLMs generate Chain-of-Thought errors of various types (Tong et al., 2024) and struggle to independently localise where in a chain a mistake occurred (Tyen et al., 2024). The result is reasoning chains that look superficially correct yet contain critical inconsistencies, undermining reliability on HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and MultiHopRAG (Tang and Yang, 2024).

We draw inspiration from human *procedural memory*—knowledge about "how to do things"—which recent work shows LLMs can extract from demonstrations of similar reasoning processes (Ruis et al., 2025). We propose **PRISM**, which couples procedural memory with real-time step-level error detection and correction in a Build–Retrieve–Update cycle. Unlike methods that target retrieval quality or final-answer voting alone, PRISM verifies consistency and logical validity at *every* reasoning step and applies type-specific correction strategies driven by memory.

Contributions. (1) A procedural-memory architecture for multi-hop QA, implementing a Build–Retrieve–Update cycle for continual learning from reasoning experiences. (2) A real-time error-detection mechanism combining algorithmic token-overlap verification with LLM-based semantic verification under a complexity-adaptive trigger. (3) Five error-type-specific correction strategies (retrieval failure, fact error, logical error, propagation error, unknown error). (4) A comprehensive evaluation under a unified re-implementation harness: every baseline is reproduced from the algorithmic core of its original publication inside the same harness, eliminating retrieval/prompt confounds. A 5-round protocol with memory retention additionally serves as a downside-defense stress test.

2 Related Work

Multi-hop reasoning. Multi-hop QA requires synthesising information across multiple steps.

Leading systems on HotpotQA reach ≈77–78 Joint F1 in the distractor setting and higher Answer-F1 alone under extractive setups, but model accuracy drops by 26–54 points when moving from extractive to generative formulations (Schnitzler et al., 2024). Recent work targets explicit chain interventions—ReAgent (Xinjie et al., 2025) introduces backtracking; Multi-Chain Reasoning (Yoran et al., 2023) aggregates independent chains; RichRAG (Wang et al., 2025) handles multi-faceted queries; HopRAG (Liu et al., 2025) traverses logic-aware passage graphs. Explainability-oriented approaches such as Han et al. (2025) identify rationale sentences with pointer networks. None of these systems, however, maintain a persistent procedural memory of *how to recover from* reasoning errors across questions. Absolute SOTA on HotpotQA is *not* claimed for PRISM; we report "best among the five compared baselines" under our unified harness.

RAG error detection and correction. CRAG (Yan et al., 2024) performs a three-stage document-confidence assessment with optional web-search fallback. Active Retrieval (Jiang et al., 2023) decides retrieval timing based on model uncertainty. These approaches operate at document or retrieval-timing granularity rather than at per-step reasoning granularity, and they do not retain a learnable correction policy across questions.

Procedural memory and memory-augmented networks. Ruis et al. (2025) show that LLMs can extract procedural knowledge from demonstrations. Memp (Fang et al., 2026) introduces task-agnostic Build–Retrieve–Update procedural-memory management. Memory-augmented networks (Graves et al., 2014; Weston et al., 2015; Sukhbaatar et al., 2015) provide foundational architectures for differentiable external memory. Self-consistency (Wang et al., 2023) improves final-answer reliability via path sampling, but operates only on the final answer. PRISM adapts the procedural-memory paradigm to multi-hop reasoning, with an explicit focus on *error patterns* and learned correction procedures, integrated with per-step error detection.

3 Methodology

3.1 Architecture Overview

PRISM (Figure 1) comprises four components: (i) a **procedural memory module** backed by vector DB with two collections, *procedures* (distilled from successful trajectories) and *lessons* (error-correction pairs); (ii) an **error detection and correction engine** that computes step-level confidence from four complementary signals and triggers a type-specific verifier with implementing five strategies; and (iii) a **reasoning chain manager** that orchestrates decomposition, retrieval, generation, and synthesis while tracking error statistics.

At inference time PRISM follows a Build–Retrieve–Update cycle: for each sub-question it retrieves documents and procedural guidance from memory, generates a candidate, runs error detection, optionally applies correction, and updates the lesson collection. After the full chain, a successful trajectory is distilled into a procedure and stored. The full algorithm and the success-labelling convention used for lesson storage are given in Appendix A.

3.2 Procedural Memory

Memory state $M_t = \{m_1, \ldots, m_n\}$ holds procedural memory units in the vector store with 2560-dim embeddings (`Qwen3-Embedding-4B`). Three update mechanisms are used.

Build (after trajectory). Each reasoning trajectory $\tau = \{(s_t, a_t, e_t)\}_{t=1}^{T}$ is decomposed into atomic operations, abstracted into three pattern categories—*retrieval patterns*, *reasoning templates*, and *error-correction patterns*—and encoded as structured text. Procedures with retrieval-time L2 distance above $\delta_{\text{proc}} = 1.2$ are filtered out, preventing irrelevant patterns from polluting guidance.

Retrieve (before each hop). For each sub-question q, PRISM retrieves the top-k most similar entries by cosine similarity in the 2560-dim space. The guidance passed to the generator is:

$$\begin{aligned}\text{guidance}(q, M) = [\,&\text{Procedures}(q)_{\leq k};\\ &\text{Lessons_success}(q)_{\leq k};\\ &\text{Lesson_avoid}(q)_{\leq 1}\,],\end{aligned} \tag{1}$$

with k=3. The AVOID hint, drawn from failed corrections, blocks repetition of correction strategies that previously did not work for similar questions.

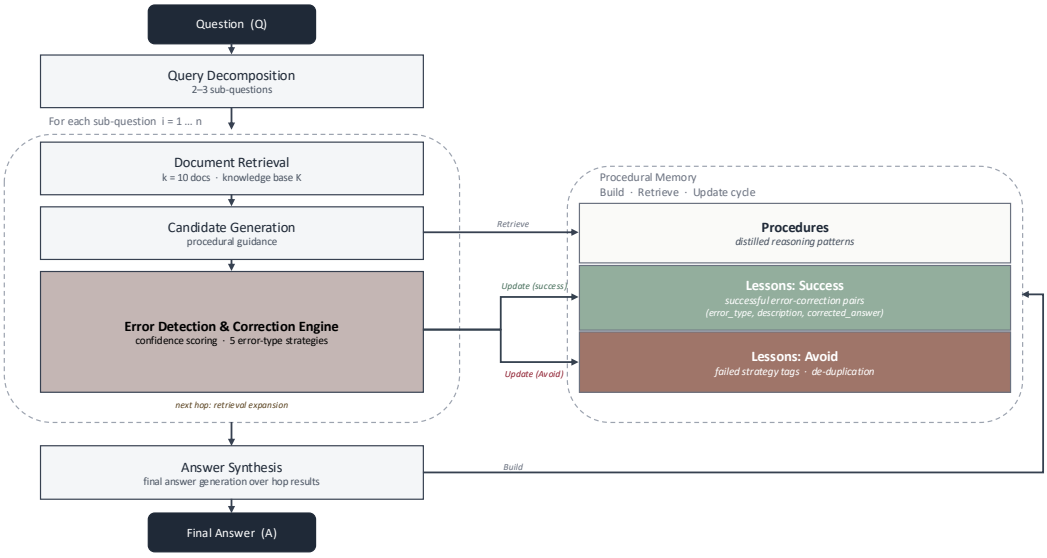

Figure 1: PRISM architecture. Per hop: retrieve documents and procedural guidance, generate a candidate, verify via algorithmic or semantic verifier, apply type-specific correction if flagged, then update memory.

Update (after each correction; after each trajectory). A correction outcome triggers lesson storage. Successful corrections (§3.4) are always stored; failed corrections are stored with an AVOID tag only when no near-identical AVOID lesson already exists, using L2 deduplication threshold δ_{lesson}=0.15. Successful trajectories (at least one hop with a valid answer) are distilled into a procedure text and appended to the procedure collection. The full deduplication rule is given in Appendix B.

3.3 Error Detection Engine

Step-level confidence. For each reasoning step, PRISM computes confidence as the *product* of four signals:

$$c(\text{step}_i \mid R, M) = c_{\text{retr}} \cdot c_{\text{cons}} \cdot c_{\text{mem}} \cdot (1 - p_{\text{err}}), \quad (2)$$

where c_{retr} measures how well the retrieved documents collectively cover the candidate-answer tokens; c_{cons} is a square-root-dampened Jaccard between the current and previous hop answers; c_{mem} is a $[0.5, 1.0]$-floored alignment score against retrieved procedural rules (defaulting to 1.0 when no rules are returned); and p_{err} is a logistic estimate combining a retrieval-confidence shortfall (weighted c_1=8) and a query-relevance shortfall (weighted c_2=2). The *multiplicative* form is deliberate: a near-zero factor collapses the product, so any *single* missing signal flags the step rather than allowing one strong signal to mask three weak ones. Full definitions of the four factors are given in Appendix C. A step is flagged when $c < \theta_{\text{confidence}}$=0.40.

Weighted verification score. A separate per-document evidence aggregate *verify_score* is computed for propagation-error detection; documents are weighted by length-based document confidence and rank-based source reliability (Appendix D).

Error classification. Errors are sorted into five types whose detection conditions and correction strategies are summarised in Table 1. Under the algorithmic verifier the type is selected by which sub-threshold trips first (read top-to-bottom). The semantic verifier delegates to the same five-row table; only the detection signal changes.

Complexity-adaptive verification. Decomposition yields 1–4 sub-questions; the 1-sub-question branch is observed in roughly 1–6% of pooled samples on the benchmarks and rises to $\approx$ 12% on MultiHopRAG. Single-sub-question steps use the algorithmic token-overlap verifier (`verify`). When $|\text{sub_queries}| \geq \theta_{\text{min_hops}}$=2, an LLM-based semantic verifier (`verify_semantic`) *replaces* the algorithmic verifier entirely; it returns a three-level verdict ("fully" / "partially" / "no") with a confidence score, and triggers an error only at thresholds θ_{no}=0.6 and θ_{partial}=0.3.

Table 1: Five PRISM error types: detection conditions under both verifier modes and correction strategies. The same `prior chain exists` guard gates `logical_error` and `propagation_error` under both verifiers, so a step with no prior context is never assigned a propagation error.

Error Type	Detection (algorithmic, single sub-q)	Detection (semantic, $\geq$2 sub-q)	Correction Strategy
`retrieval_failure`	Answer $\in$ {"I don't know", "unknown", ""} (checked before either verifier)	same	Best-guess re-prompt from partial evidence
`fact_error`	$c_{\text{retr}} < \theta_{\text{fact}}$=0.20	semantic verdict "no" with conf. $> \theta_{\text{no}}$=0.6 AND no prior chain	High-confidence document subset (top half by token recall)
`logical_error`	$c_{\text{cons}} < \theta_{\text{logic}}$=0.30 AND prior chain exists	semantic verdict "partially" with conf. $< \theta_{\text{partial}}$=0.3 AND prior chain	Atomic step decomposition prompt
`propagation_error`	*verify_score* $< \theta_{\text{prop}}$=0.20 AND prior chain exists	semantic verdict "no" with conf. $> \theta_{\text{no}}$=0.6 AND prior chain	Reinitialise from top-1 document only
`unknown`	$c < \theta_{\text{confidence}}$=0.40 (no specific sub-threshold triggered)	semantic verifier flags an error but none of the above apply	Majority vote across k=3 independent chains

3.4 Correction Executor

Retrieval failure retries with an explicit best-guess directive on the partial evidence. **Fact error** reranks documents by token recall and uses only the top half, with a strict "stated facts only" instruction. **Logical error** instructs the model to decompose reasoning into independently verifiable atomic steps. **Propagation error** reinitialises from the single top-ranked document, discarding the prior chain. **Unknown error** samples k=3 chains from disjoint document subsets and selects by pairwise Jaccard consensus.

Revert safeguard. After every correction, if (i) the corrected answer equals the original or (ii) the corrected answer is "I don't know" while the original was not, the correction is discarded and the original is retained. A failed lesson tagged AVOID is then stored (subject to the dedup rule above) to prevent repeating the same ineffective strategy on similar future questions.

4 Experiments

4.1 Setup

Datasets. We evaluate on four multi-hop QA benchmarks: HotpotQA (Wikipedia, 2 hops), 2WikiMultiHopQA (Wikipedia, 2 hops), MuSiQue (Wikipedia, 2–4 hops), and MultiHopRAG (news, 2+ hops). All evaluation samples are drawn from each dataset's published *validation* split; train splits are not used, as PRISM has no trainable parameters beyond the frozen LLM and embedding endpoints. Each evaluation round draws n=500 questions from each dataset's indexed validation pool (HotpotQA 7,405; MuSiQue 2,417; 2Wiki 5,000; MultiHopRAG 2,556), and the same 500-question subset is shared by all six systems within that round.

Baselines. PRISM is compared against **Vanilla RAG** (retrieve-k=10, generate), **MEMP** (Fang et al., 2026), **CRAG** (Yan et al., 2024) (web-search fallback disabled), **ReAgent** (Xinjie et al., 2025), and **HopRAG** (Liu et al., 2025). All baselines are re-implemented from the algorithmic core of their original publications within the same harness, sharing retriever, LLM endpoint, and prompt-formatting code with PRISM.

Metrics. Token-level F1 and Exact Match (EM) after lowercase + article/punctuation normalisation. Parse failures are scored as 0 rather than excluded; PRISM's internal error layer never removes a sample from the denominator. We report mean $\pm$ std over 5 independent rounds with seeds $\{42, 123, 456, 789, 1024\}$.

Implementation. Base LLM `google/gemma-4-31B-it`; embedding model `Qwen3-Embedding-4B` (2560-dim); retrieval k=10; ChromaDB memory with capacity 10,000 patterns/namespace; thresholds $\theta_{\text{confidence}}$=0.40 and $\theta_{\text{retrieval}}$=0.25 selected via grid search on a 50-sample held-out split. Memory is reset *once* at the start of round 1 and then *retained* across rounds 2–5; this protocol simultaneously functions as a **downside-defense stress test** for memory-induced drift (§4.2). Computing

Table 2: Main results (F1% / EM%), `google/gemma-4-31B-it`, mean ± std over 5 rounds, n=500/round.

System	HotpotQA		2WikiMultiHopQA		MuSiQue		MultiHopRAG	
	F1	EM	F1	EM	F1	EM	F1	EM
Vanilla RAG	43.6 ± 1.6	30.2 ± 1.9	31.9 ± 2.2	19.8 ± 1.7	10.1 ± 0.9	6.5 ± 0.9	3.0 ± 0.3	2.9 ± 0.3
MEMP	59.0 ± 1.1	47.9 ± 1.6	68.4 ± 1.5	61.7 ± 1.6	12.3 ± 1.9	9.1 ± 1.3	15.0 ± 1.8	14.8 ± 1.7
CRAG	61.0 ± 1.5	40.5 ± 1.8	42.1 ± 1.3	21.4 ± 1.4	12.8 ± 1.5	7.7 ± 0.8	3.0 ± 0.4	2.8 ± 0.4
ReAgent	63.6 ± 1.7	51.8 ± 2.0	65.4 ± 1.8	58.4 ± 1.8	21.3 ± 1.3	16.4 ± 0.7	12.5 ± 1.5	12.4 ± 1.4
HopRAG	61.6 ± 1.5	40.8 ± 2.3	42.6 ± 1.2	21.2 ± 1.3	13.1 ± 1.5	7.6 ± 1.3	4.2 ± 0.3	4.2 ± 0.3
PRISM	$\mathbf{75.1 \pm 0.7}$	$\mathbf{61.4 \pm 1.4}$	$\mathbf{74.8 \pm 1.4}$	$\mathbf{66.9 \pm 1.7}$	$\mathbf{33.5 \pm 1.1}$	$\mathbf{25.9 \pm 1.4}$	$\mathbf{19.0 \pm 1.9}$	$\mathbf{18.9 \pm 1.7}$

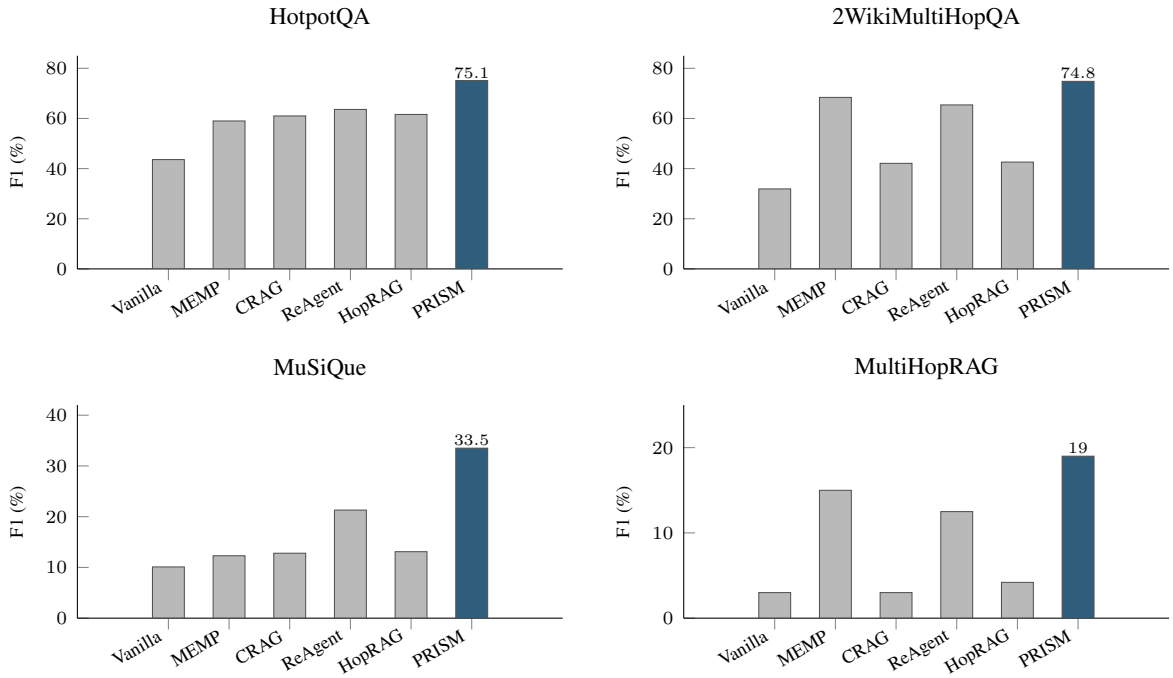

Figure 2: F1 (%) of the six compared systems on the four benchmarks (Table 2; `google/gemma-4-31B-it`, 5-round mean). PRISM ranks first on every benchmark; per-panel y-axis scales differ because the four benchmarks have different absolute-score floors (Appendix J).

infrastructure and the ≈720 GPU-hours used are documented in Appendix E.

4.2 Main Results

Table 2 reports F1/EM across all six systems and four benchmarks (mean ± std, 5 rounds, n=500/round). Run-level σ is at most 2.3 F1 in every cell and at most 1.9 F1 on PRISM rows (largest on MultiHopRAG); this PRISM-only bound is the reference scale used in the ablation deltas (§4.4).

PRISM achieves the best F1 and EM on every benchmark (Figure 2). Margins over the strongest other system per benchmark: HotpotQA +11.5 F1 (vs. ReAgent), 2Wiki +6.4 F1 (vs. MEMP), MuSiQue +12.2 F1 (vs. ReAgent), MultiHopRAG +4.0 F1 (vs. MEMP). The absolute SOTA on HotpotQA in the distractor setting (Joint F1 ≈ 77–78 / Answer F1 ≈ 79–84) is not claimed; PRISM is best *among the five compared baselines* under our unified harness.

MuSiQue and MultiHopRAG depress *absolute* scores via different structural mechanisms—MuSiQue through adversarial compositional construction with hard distractors, MultiHopRAG through retrieval-ceiling effects (≈ 0.45 recall) and short-answer token-F1 sensitivity (Appendix J). Within these constraints, the cross-system ordering and PRISM's relative margin are the load-bearing quantities: on MuSiQue PRISM is ≈3.3× Vanilla, ≈2.7× MEMP, and ≈1.6× ReAgent.

Table 3: Ablation (F1% / EM%). The largest deltas in every column are caused by removing error detection; all four drops exceed the PRISM $\sigma \leq 1.9$ F1 run-level variance of the 5-round main experiment.

Configuration	HotpotQA		2WikiMHQA		MuSiQue		MultiHopRAG	
	F1/EM	ΔF1	F1/EM	ΔF1	F1/EM	ΔF1	F1/EM	ΔF1
PRISM (Full)	75.1/61.1	—	73.8/66.2	—	32.2/25.7	—	19.6/19.2	—
w/o Doc Filter	74.2/59.8	−0.9	72.8/65.1	−1.0	31.8/25.3	−0.4	18.9/18.3	−0.7
w/o Error Detection	61.1/48.8	**−14.0**	66.7/60.5	**−7.1**	14.1/10.6	**−18.1**	15.4/15.1	**−4.2**
w/o Retrieval Expansion	74.2/59.7	−0.9	72.4/65.0	−1.4	29.9/23.2	−2.3	17.8/17.4	−1.8

4.3 Memory Accumulation: Downside-Defense Analysis

Recall that procedural memory is reset once before round 1 and *retained* across rounds 2–5. The five baselines carry no cross-round state, so their per-round trajectories should, a priori, be driven by sample variation alone. Only PRISM is exposed to the additional risk of noise drift, stale-lesson pollution, and over-fitting to early patterns—a deliberately adversarial condition.

The full per-round results for PRISM are in Appendix F. The two summary statistics that follow are sufficient for the downside-defense claim. First, PRISM's *minimum* round-level F1 is within 1.1 F1 of its round-1 cold start on every benchmark, and its R5 score is *above* R1 on every dataset ($\Delta = +0.83$ to $+3.94$ F1). Second, the F1–round Pearson correlation is non-negative on every benchmark for PRISM and strongly positive on two (r=0.87 on 2Wiki, r=0.76 on MuSiQue); by contrast, four of the five baselines exhibit a *negative* r on at least one benchmark (the severe case is MEMP MuSiQue $r = -0.60$ and HopRAG MuSiQue $r = -0.44$). Table 4 summarises.

Table 4: F1–round Pearson r (5 rounds, n=500/round). PRISM is the only system non-negative on every benchmark, despite carrying memory-drift risk that the baselines do not.

System	HotpotQA	2Wiki	MuSiQue	MHR
Vanilla RAG	+0.21	+0.21	−0.07	−0.40
MEMP	+0.25	+0.74	−0.60	+0.20
CRAG	+0.39	+0.75	−0.24	+0.17
ReAgent	+0.59	+0.46	+0.14	+0.49
HopRAG	+0.35	−0.17	−0.44	−0.38
PRISM	**+0.37**	**+0.87**	**+0.76**	**+0.47**

The PRISM–strongest-baseline gap is preserved or widened on three of the four benchmarks from R1→R5 (2Wiki +5.79 → +6.16 pp, MuSiQue +12.58 → +13.64 pp, MultiHopRAG +2.20 → +3.89 pp) and narrows slightly on HotpotQA (+13.12 → +11.43 pp)—consistent with PRISM operating near the harness-saturated regime on that benchmark. We deliberately do *not* claim a strictly monotonic upward trajectory; the load-bearing claim is the defensive one—PRISM is the only system that absorbs both sample variation *and* memory-accumulation risk without entering a negative-correlation regime on any benchmark.

4.4 Ablation Study

We test three variants: `no_filter` (remove document pre-filtering), `no_verify` (remove all error detection and correction), `no_expand` (remove single-step retrieval expansion between hops). Results are averaged over 2 single-round runs per variant, n=500/run; memory is cleared per run, so differences from Table 2 reflect both the 2-run protocol and the absence of cross-round memory growth.

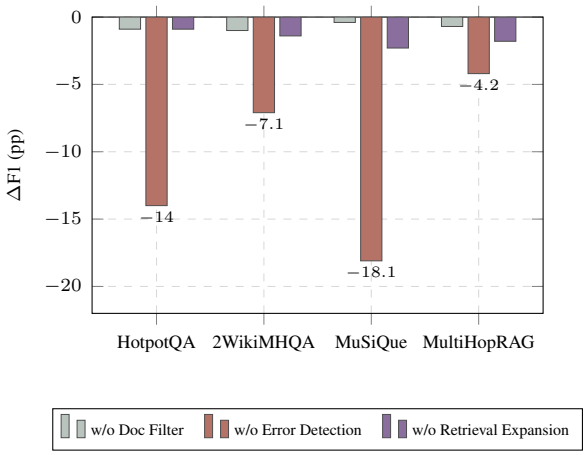

Figure 3: Ablation effects, ΔF1 (pp) versus PRISM (Full), by benchmark (Table 3). All bars are negative (every ablation strictly hurts). Removing error detection produces the largest drop on every benchmark, peaking at −18.1 on MuSiQue—more than the PRISM-versus-MEMP gap on the same dataset.

Error detection is the most critical component (Figure 3): removing it costs −14.0, −7.1, −18.1, and −4.2 F1 across the four benchmarks. The `no_verify` F1 on HotpotQA (61.1%) closely

matches MEMP (59.0%), confirming that the verifier is the primary differentiator between PRISM and MEMP. Retrieval expansion contributes meaningfully on the harder benchmarks (−2.3 MuSiQue, −1.8 MultiHopRAG, −1.4 2Wiki). Document filtering provides uniformly small benefits (−0.4 to −1.0 F1).

4.5 Computational Efficiency

Table 5: Runtime on HotpotQA. *Runtime (s)* is the mean wall-clock time per 500-sample run, averaged over the 5 rounds of the main experiment. CRAG's web-search fallback is disabled; enabling it would make CRAG slower still.

System	Runtime (s)	Rel.	ms/Q
Vanilla RAG	112	1.0×	≈224
MEMP	617	5.5×	≈1,234
HopRAG	631	5.6×	≈1,262
ReAgent	648	5.8×	≈1,296
PRISM	**1,829**	**16.3×**	**≈ 3,658**
CRAG	6,306	56.3×	≈12,612

PRISM introduces a 16.3× overhead over Vanilla RAG, of which roughly 54% is base LLM inference (decomposition + per-hop generation + synthesis) and 46% is PRISM-specific (memory retrieval, error detection, correction, memory update). The full per-component breakdown is in Appendix G. PRISM remains ≈ 3.5× faster than CRAG (56.3×) while achieving substantially higher accuracy (Table 2), a favourable trade-off where answer quality is paramount.

5 Analysis

5.1 Step-Depth Degradation

We measure how Exact Match varies with chain step depth—the number of sub-questions produced by each system's decomposition step. Because Vanilla RAG, CRAG, and HopRAG do not decompose, all of their questions are 1-step by definition. To make relative degradation directly comparable across all six systems, we pool the per-question results of the 5-round main experiment across the two deepest benchmarks (MuSiQue + MultiHopRAG, $n \approx 5{,}000$). HotpotQA and 2Wiki are excluded from this aggregate: HotpotQA's 2→3-step shift is within ±7% relative for every system (rare chains beyond depth 2), and the 2Wiki 3+-step bucket is dominated by yes/no comparison questions that hit a ≈ 90% ceiling for every decomposing system. The per-dataset breakdown is in Appendix H.

Table 6: Step-depth degradation on MuSiQue + MultiHopRAG pooled. Δ rel is the 2→3-step EM drop expressed as a percentage of 2-step EM, computed on unrounded per-question pooled means.

System	1-step n=435	2-step n=2,015	3-step n=2,542	Δ rel 2→3
Vanilla RAG	4.8	7.8	3.2	−58.9%
CRAG	4.1	6.6	3.3	−49.9%
HopRAG	6.4	7.7	4.4	−43.6%
MEMP	17.7	14.8	8.8	−40.7%
ReAgent	16.3	16.7	12.2	−27.1%
PRISM	**22.5**	**25.4**	**19.8**	**−21.8%**

Every baseline loses 27−59% of its 2-step accuracy when chains extend to three steps—consistent with error propagation through multi-hop chains. PRISM retains the largest fraction (−21.8%), the smallest drop of all six systems. The 5.3-pp gap in relative drop between PRISM and the next-best system (ReAgent, −27.1%) is on the same order as the gap between ReAgent and the four non-decomposing/non-correcting baselines (−40% to −59%), so the additional resilience from PRISM's combined detection-and-correction mechanism is comparable to that from decomposition with backtracking alone. PRISM also starts from the highest 2-step base *and* loses the smallest relative fraction of it—the expected signature of late-step error propagation being suppressed rather than merely local retrieval being improved.

Gold intermediate-step labels are not provided by the four benchmarks, so we do not report per-step accuracy in isolation. Jeong et al. (2025) formalise intermediate-step error analysis in the knowledge-editing setting and report that intermediate errors occur in ≈20% of cases even when the final answer is correct; adapting this step-resolved supervision to RAG is a natural direction for future work.

5.2 Model Scalability

To verify that PRISM's gains are model-agnostic in direction, we evaluate the 3-system comparison (Vanilla RAG, MEMP, PRISM) across seven additional backbones on all four benchmarks. Full per-cell results are in Appendix I; the per-backbone ΔF1 averaged across the four datasets is given in Table 7.

PRISM outperforms both Vanilla RAG and MEMP on the 4-dataset average for every one of the 8 backbones (Figure 4), with Δ vs. Vanilla in $[+10.9, +28.4]$ and Δ vs. MEMP in

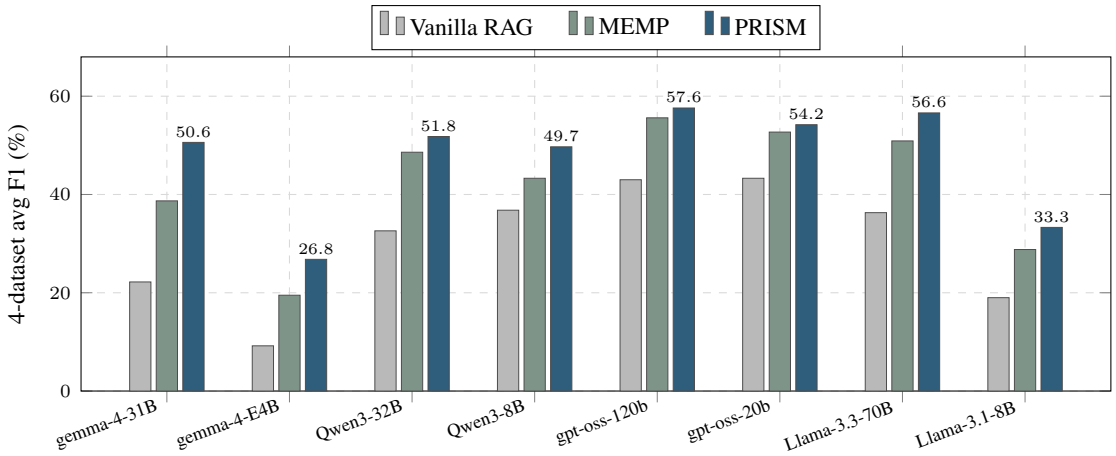

Figure 4: Model scalability: 4-dataset average F1 of Vanilla RAG, MEMP, and PRISM across 8 backbones (Table 7; full per-cell numbers in Appendix I). Backbones are grouped by family (gemma, Qwen, gpt-oss, Llama). PRISM is the highest bar in every backbone group; the PRISM–MEMP margin narrows on the strongest backbones (`gpt-oss-20b`, `gpt-oss-120b`) but never inverts.

Table 7: ΔF1 vs. Vanilla RAG and vs. MEMP, averaged across the four datasets. † denotes single-run preliminary numbers (full 5-run averages deferred to future work).

Model	Δ vs. Vanilla	Δ vs. MEMP
`gemma-4-31B-it`	+28.4	+11.9
`gemma-4-E4B-it`	+17.6	+7.3
`Qwen3-32B`†	+19.2	+3.2
`Qwen3-8B`†	+12.9	+6.4
`gpt-oss-120b`†	+14.6	+2.0
`gpt-oss-20b`†	+10.9	+1.5
`Llama-3.3-70B`†	+20.2	+5.7
`Llama-3.1-8B`†	+14.4	+4.6

$[+1.5, +11.9]$. The PRISM–MEMP margin narrows on the strongest backbones (`gpt-oss-20b`, `gpt-oss-120b`) but never inverts. Per-backbone HotpotQA-only views are misleading: on HotpotQA the PRISM–MEMP gap shrinks to under 1 pp for several strong backbones, but on harder benchmarks (2Wiki, MuSiQue, MultiHopRAG) the procedural-memory contribution remains visible. PRISM's gains are therefore model-agnostic in direction but model-dependent in magnitude.

6 Conclusion

We proposed PRISM, a procedural-memory error-detection-and-correction system for multi-hop RAG. PRISM achieves consistent improvements across four benchmarks against five competitively re-implemented baselines under a single unified harness, with the largest absolute and relative gains on the deepest benchmarks (MuSiQue, MultiHopRAG). Under a 5-round protocol where procedural memory is retained between rounds—a downside-defense stress test that exposes only PRISM to memory-induced drift—PRISM is the only system whose per-round F1 trajectory is non-negative on every benchmark, while four of the five no-memory baselines drift downward on at least one dataset. Ablations show error detection contributes up to 18.1 F1 (MuSiQue), and a pooled step-depth analysis shows PRISM retains the largest fraction of its 2-step EM at depth 3 (-21.8% vs. -27.1% for the next-best system). The improvements are model-agnostic in direction across 8 backbones spanning 4B–120B parameters.

Limitations

Current limitations. (1) **Language coverage:** English-only evaluation; performance on morphologically rich or low-resource languages is unexplored. (2) **Domain generalisation:** general-knowledge questions only; specialised domains (medical, legal, scientific) are not systematically evaluated. (3) **Computational overhead:** $16.3\times$ Vanilla RAG latency on HotpotQA (§4.5); the same relative ordering holds on the other datasets but absolute multipliers may differ, especially on deeper-chain benchmarks like MuSiQue where corrections fire more often. Early-exit detection and cached memory retrieval are natural mitigations. (4) **Memory management:** fixed-capacity memory (10,000 patterns) may not scale to very large-scale or highly diverse deployments. (5) **Fixed thresholds:** $\theta_{\text{confidence}}=0.40$ and $\theta_{\text{retrieval}}=0.25$ may not

transfer across domains; adaptive thresholds are a future direction. (6) **Bias and representation:** all four benchmarks draw from English Wikipedia and English news, so the system inherits any topical, geographic, and demographic skew present in those sources; we discuss the related risk of bias accumulation in procedural memory in the *Ethical Considerations* section below.

Future directions. Cross-lingual procedural memory; domain-adaptive few-shot procedural learning; lightweight error detection (early-exit, cascade classifiers) and memory retrieval acceleration for conversational settings; continual learning with consolidation to avoid catastrophic forgetting; and interpretability tooling for the detection-correction pipeline.

Ethical Considerations

Datasets. All four evaluation datasets are publicly released academic benchmarks distributed for research use: HotpotQA (Yang et al., 2018) (CC BY-SA 4.0), 2WikiMultiHopQA (Ho et al., 2020) (Apache 2.0), MuSiQue (Trivedi et al., 2022) (CC BY 4.0), and MultiHopRAG (Tang and Yang, 2024) (released under the terms specified by the original authors). The corpora derive from publicly available Wikipedia content (HotpotQA, 2Wiki, MuSiQue) and publicly available news articles (MultiHopRAG); PII and offensive-content screening is handled by the original dataset curators at source-curation time, and we did not collect, annotate, or release any new human-generated data. Each dataset is used within its stated license.

Models. Experiments use openly available LLMs served via a local vLLM-compatible endpoint, each used under its respective license for non-commercial academic research: `google/gemma-4-31B-it` and `gemma-4-E4B-it` (Google DeepMind, 2026) (Apache 2.0), `meta-llama/Llama-3.3-70B-Instruct` and `Llama-3.1-8B-Instruct` (Grattafiori et al., 2024) (Llama 3 Community License), `Qwen/Qwen3-32B` and `Qwen3-8B` (Yang et al., 2025) (Apache 2.0), and `openai/gpt-oss-120b` / `gpt-oss-20b` (OpenAI et al., 2025) (Apache 2.0). The embedding model `Qwen3-Embedding-4B` (Zhang et al., 2025) is also Apache 2.0.

Intended use and risks. PRISM is a research artefact for factuality and error correction in retrieval-augmented multi-hop QA. It is *not* a safeguarded production pipeline and should not be deployed in safety-critical domains (medical, legal, financial) without domain-specific calibration of confidence thresholds, independent evaluation, and human oversight. PRISM reduces but does not eliminate hallucination (Table 2); careless deployment as authoritative could amplify rather than mitigate misinformation harms.

Procedural memory and bias accumulation. Because PRISM's procedural memory accumulates lessons from prior interactions (§4.1), incorrect or biased lessons can persist. Deduplication and AVOID-tagging (§3.2) limit but do not eliminate this. Long-running deployments should periodically audit the `procedures` and `lessons` collections and consider scheduled resets for sensitive domains.

Compute and environmental impact. PRISM incurs $\approx 16.3\times$ Vanilla RAG latency (§4.5); full computing infrastructure and the ≈ 720 GPU-hour budget are documented in Appendix E. No pre-training or fine-tuning was performed—all reported compute is inference-only. Per-question runtimes are reported so downstream users can estimate the energy trade-off on their own hardware.

Human-subject involvement. No human annotators or subjects were employed. All evaluation labels come from the released gold annotations of the four public benchmarks.

Use of AI assistants. Generative AI assistants were used for light language polishing of the manuscript text (grammar, phrasing, and spelling); they were not used for research ideation, experimental design and so on.

Code and artefacts. Full source code, experiment-runner scripts, result-export utilities, and pre-built Langflow flow definitions will be released upon acceptance at **[anonymous repository - link provided upon acceptance]**. The repository includes (a) core implementations of all systems and the procedural/episodic memory modules; (b) a benchmark runner CLI for all four datasets with configurable sample size, seed, and worker count; and (c) Langflow JSON flow definitions for interactive demonstration. All experiments were conducted with `google/gemma-4-31B-it` served via a vLLM-compatible OpenAI endpoint, configurable through

the LLM_BASE_URL and LLM_MODEL environment variables.

References

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2026. Memp: Exploring agent procedural memory. *Preprint*, arXiv:2508.06433.

Google DeepMind. 2026. Gemma 4: Open multimodal models.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Kyubeen Han, Youngjin Jang, and Harksoo Kim. 2025. Explainable multi-hop question answering: A rationale-based approach. *Big Data and Cognitive Computing*, 9(11).

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Geunyeong Jeong, Juoh Sun, and Harksoo Kim. 2025. Watch your step: A fine-grained evaluation framework for multi-hop knowledge editing in large language models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 4842–4846, New York, NY, USA. Association for Computing Machinery.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

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. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. 2025. HopRAG: Multi-hop reasoning for logic-aware retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1897–1913, Vienna, Austria. Association for Computational Linguistics.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Laura Ruis, Maximilian Mozes, Juhan Bae, Siddhartha Rao Kamalakara, Dwaraknath Gnaneshwar, Acyr Locatelli, Robert Kirk, Tim Rocktäschel, Edward Grefenstette, and Max Bartolo. 2025. Procedural knowledge in pretraining drives reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*.

Julian Schnitzler, Xanh Ho, Jiahao Huang, Florian Boudin, Saku Sugawara, and Akiko Aizawa. 2024. Morehopqa: More than multi-hop reasoning. *Preprint*, arXiv:2406.13397.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2440–2448, Cambridge, MA, USA. MIT Press.

Yixuan Tang and Yi Yang. 2024. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080, Bangkok, Thailand. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13894–13908, Bangkok, Thailand. Association for Computational Linguistics.

Shuting Wang, Xin Yu, Mang Wang, Weipeng Chen, Yutao Zhu, and Zhicheng Dou. 2025. RichRAG:

Crafting rich responses for multi-faceted queries in retrieval-augmented generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11317–11333, Abu Dhabi, UAE. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhao Xinjie, Fan Gao, Xingyu Song, Yingjian Chen, Rui Yang, Yanran Fu, Yuyang Wang, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. 2025. ReAgent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop QA. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4089, Suzhou, China. Association for Computational Linguistics.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

A PRISM Algorithm

The full PRISM inference-time procedure, including the success-labelling convention for stored lessons, is given in Algorithm 1.

Algorithm 1 PRISM Multi-Hop Reasoning with Error Detection and Correction

Require: Question q, Knowledge base K, Procedural memory M
Ensure: Answer a, Reasoning trace T, Error stats E_{stats}

sub_queries $\leftarrow$ Decompose(q) ▷ 1–4
use_semantic $\leftarrow$ $|\text{sub_queries}| \geq \theta_{\text{min_hops}}$
hop_results, reasoning_chain, extra_docs $\leftarrow$ [], [], []
for all sub-question sq_i in sub_queries (1-indexed) **do**
  docs $\leftarrow$ Retrieve(K, sq_i, k=10)
  docs $\leftarrow$ docs $\cup$ $\{d \in \text{extra_docs} : d \notin \text{docs}\}$
  docs $\leftarrow$ FilterDocs(sq_i, docs)
  guidance $\leftarrow$ RetrieveProcedures(M, sq_i) $\cup$ RetrieveLessons(M, sq_i)
  cand $\leftarrow$ Generate(sq_i, docs, guidance)
  if cand $\in \mathcal{D}$ **then** ▷ $\mathcal{D}$=DONT_KNOW
    $v \leftarrow$ {is_error:$\top$, type:"retrieval_failure"}
  else if use_semantic **then**
    $v \leftarrow$ VerifySemantic(...)
  else
    $v \leftarrow$ VerifyAlgorithmic(...)
  end if
  if v.is_error **then**
    $c \leftarrow$ Correct(sq_i, docs, cand, v)
    corr $\leftarrow$ c.corrected_answer
    bad $\leftarrow$ (corr$\in\mathcal{D} \wedge$ cand$\notin\mathcal{D}$) $\vee$ (corr = cand)
    if bad **then**
      StoreLessonAVOID(M, sq_i, v, c)
      corr $\leftarrow$ cand
    else
      StoreLessonSuccess(M, sq_i, v, corr)
      E_{stats}.corrected += 1
    end if
    cand $\leftarrow$ corr
  end if
  if $i < |\text{sub_queries}|$ **then**
    extra_docs $\leftarrow$ ExpandRetrieval(sq_i, cand, docs)
  end if
  hop_results.append({sq_i, cand, docs})
  reasoning_chain.append(cand)
end for
$a \leftarrow$ Synthesize(q, hop_results)
BuildMemory(M, q, hop_results)
StoreEpisodic(T)
return a, T, E_{stats}

Success label. The `StoreLessonSuccess` branch is reached when the correction (i) differs from the original bad candidate and (ii) is not a `DONT_KNOW` abstention. Gold answers are not available at inference time, so this label is operationally a *non-degradation* criterion. Using gold answers to label lessons would leak supervision into the procedural-memory loop and would not be available in any realistic deployment. The AVOID-tagging safeguard (§3.4) and the lesson-deduplication rule (Appendix B) together prevent the small fraction of mis-labelled "successes" from accumulating, consistent with the empirical absence of downward drift under 5-round memory accumulation (§4.3).

B Procedural Memory Equations

Pattern extraction. Each trajectory τ is decomposed as $\tau = \{(s_t, a_t, e_t)\}_{t=1}^{T}$ where s_t is the reasoning state at step t, a_t is the retrieval/generation action, and e_t encodes any detected error and its resolution. The extraction map is:

$$P(\tau) = \text{extract_procedure}(\tau). \tag{3}$$

Lesson update rule. Let $d^*(l_t) := \min_{m \in M_t} \|l_t - m\|_2$ denote the L2 distance from the candidate lesson l_t to its nearest neighbour in M_t. After every correction attempt, the lesson collection is updated according to the outcome:

$$M_{t+1}^{\text{lesson}} = \begin{cases} M_t^{\text{lesson}} \cup \{l_t\} & \text{if success,} \\ M_t^{\text{lesson}} \cup \{l_t\} & \text{if failed and} \\ & d^*(l_t) \geq \delta_{\text{lesson}}, \\ M_t^{\text{lesson}} & \text{otherwise.} \end{cases} \tag{4}$$

Here $\delta_{\text{lesson}} = 0.15$ in L2 distance over 2560-dim embeddings.

Procedure text format. Successful trajectories (at least one hop with a valid answer) are distilled into a procedure structured as:

```
For questions like: '<query>'
  Step 1: To answer '<sub-q1>', retrieve
          passages [Sources: ...] -> answer found
  Step 2: To answer '<sub-q2>', retrieve
          passages [Sources: ...] -> answer found
```

Procedures with retrieval-time L2 distance exceeding $\delta_{\text{proc}} = 1.2$ are filtered out.

C Confidence Component Details

The four signals in Eq. (2) are defined as follows. The threshold parameters used throughout are $\theta_{\text{retrieval}} = 0.25$, $\theta_{\text{rel}} = 0.10$, $c_1 = 8.0$, and $c_2 = 2.0$.

Component 1 — Retrieval confidence. Token recall over the union of all retrieved documents:

$$c_{\text{retr}}(\text{step}_i, K) = \frac{|\text{tokens(ans)} \cap \bigcup_{d \in K} \text{tokens}(d)|}{|\text{tokens(ans)}|}. \tag{5}$$

A low value indicates the answer makes factual claims not supported by any retrieved document.

Component 2 — Consistency score. Square-root-dampened Jaccard between current and previous hop answers:

$$c_{\text{cons}}(\text{step}_i, R) = \Big[\alpha \cdot \text{Jaccard}\big(\text{tokens}(\text{step}_i), \text{tokens}(\text{step}_{i-1})\big)\Big]^{\beta}, \tag{6}$$

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{7}$$

with $\alpha = 1.0$ and $\beta = 0.5$ (dampening to avoid over-penalising lexically diverse but semantically coherent answers). For the first sub-question (i=1), c_{cons} defaults to 1.0.

Component 3 — Memory support. Let M_{ret} be the set of retrieved procedural rules. Define rule overlap as:

$$\text{rule_overlap}(\text{step}_i, M) = \frac{|\text{tokens(ans)} \cap \text{tokens}(M_{\text{ret}})|}{|\text{tokens(ans)}|}. \tag{8}$$

The memory-support score is:

$$c_{\text{mem}}(\text{step}_i, M) = \begin{cases} 0.5 + 0.5 \cdot \text{rule_overlap} & M_{\text{ret}} \neq \emptyset, \\ 1.0 & \text{otherwise.} \end{cases} \tag{9}$$

This yields $c_{\text{mem}} \in [0.5, 1.0]$ when procedural rules are retrieved and 1.0 when none are, so absence of memory does not penalise cold-start steps.

Component 4 — Error probability. The query-relevance signal is:

$$\text{query_relevance}(\text{step}_i) = \text{Jaccard}\big(\text{tokens}(\text{ans}), \text{tokens}(\text{q})\big). \tag{10}$$

The detector score combines a retrieval shortfall and a relevance shortfall:

$$s(\text{step}_i) = c_1 \cdot (\theta_{\text{retrieval}} - c_{\text{retr}}) + c_2 \cdot (\theta_{\text{rel}} - \text{query_relevance}), \tag{11}$$

$$p_{\text{err}}(\text{step}_i) = \sigma\big(s(\text{step}_i)\big). \tag{12}$$

Both shortfalls are positive when their underlying signal is weak, so a positive detector score pushes p_{err} toward 1. Retrieval coverage is weighted $4\times$ more heavily than answer–query overlap because retrieval shortfalls are the dominant empirical failure mode in multi-hop RAG.

D Weighted Verification Score

In addition to the four confidence components, the algorithmic verifier computes a per-document evidence aggregate used for propagation-error detection:

$$\text{verify_score}(\text{step}_i) = \frac{\sum_{d \in K} w_d \cdot \text{match}(\text{ans}, d)}{\sum_{d \in K} w_d}, \tag{13}$$

$$\text{match}(\text{ans}, d) = \frac{|\text{tokens}(\text{ans}) \cap \text{tokens}(d)|}{|\text{tokens}(\text{ans})|}, \tag{14}$$

$$w_d = \underbrace{\min\Big(1, \tfrac{|\text{tokens}(d)|}{200}\Big)}_{\text{doc-confidence}} \cdot \underbrace{\tfrac{1}{1+\text{rank}(d)}}_{\text{source reliability}} \cdot \underbrace{1}_{\text{temporal}}. \tag{15}$$

The temporal factor is fixed at 1 for the QA benchmarks used in this work; it is retained in the formulation to make the framework directly applicable to time-stamped corpora in future work. A low *verify_score* indicates that no high-quality, top-ranked document strongly supports the answer—the operational definition of a propagation error.

Threshold relationships. $\theta_{\text{fact}} = 0.20$ used in Table 1 is the *classification* threshold on the raw retrieval-confidence signal and is distinct from $\theta_{\text{retrieval}} = 0.25$ used in the logistic transform. The two thresholds operate on the same input signal but serve different purposes: θ_{fact} gates a hard error-type assignment, while $\theta_{\text{retrieval}}$ shapes the smooth contribution of retrieval coverage to the overall confidence product. When multiple sub-thresholds trigger on the same step, the Table 1 ordering favours retrieval-related assignments over chain-coherence assignments, since retrieval shortfalls are the dominant empirical failure mode.

E Computing Infrastructure

All experiments were run on a local on-premises GPU cluster with two vLLM-compatible OpenAI-format service endpoints. The base LLM endpoint (serving the primary `google/gemma-4-31B-it` backbone for the main results and the seven additional backbones in §5.2, parameter counts 4B–120B) ran on $2\times$ NVIDIA H200 GPUs with tensor parallelism across the two cards. The embedding endpoint (serving `Qwen3-Embedding-4B`) ran on $1\times$ NVIDIA RTX 6000 GPU. The full evaluation suite—comprising the main experiment ($\times 4$ benchmarks $\times 6$ systems $\times 500$ samples), the ablation study, and the 8-backbone scalability sweep—was completed in approximately 10 days of continuous service time, corresponding to roughly $2 \times 24 \times 10 \approx 480$ H200 GPU-hours for LLM inference plus $1 \times 24 \times 10 \approx 240$ RTX 6000 GPU-hours for embedding inference ($\approx$720 GPU-hours combined). No model pre-training, fine-tuning, or RL adaptation was performed; all reported compute is inference-only.

F Round-Level F1 Trajectory

Table 8 reports PRISM's per-round F1 and F1–round Pearson correlation for the 5-round protocol described in §4.1. Procedural memory is reset once before round 1 and retained across rounds 2–5; the variance therefore captures both sample variation and memory-growth effects.

G Per-Component Latency Decomposition

Table 9 decomposes the per-question latency of PRISM by pipeline component on HotpotQA. Rows are ordered by pipeline phase (the order in which each component executes during one query);

Table 8: PRISM round-level F1 under cumulative procedural memory (n=500/round). Minimum round-level F1 is within 1.1 F1 of round 1 cold start on every benchmark; R5 is above R1 on every dataset. MHR = MultiHopRAG.

Dataset	R1	R2	R3	R4	R5	Δ	r
HotpotQA	74.11	75.20	76.03	75.14	74.94	+0.83	+0.37
2Wiki	73.86	73.47	74.72	74.75	77.08	+3.22	**+0.87**
MuSiQue	32.89	32.54	32.73	35.10	34.32	+1.43	**+0.76**
MHR	17.53	20.58	17.40	18.20	21.47	+3.94	+0.47

the five components sum to the 16.3× total in Table 5 by construction. The single largest contributor is base LLM inference (8.9× Vanilla), not the PRISM-specific verification or memory machinery. MEMP inherits the same decomposition overhead and lands at 5.5× Vanilla; the residual ∼750 ms gap to PRISM is attributable to additional verification-prompt regenerations during correction.

Table 9: Per-component PRISM latency (HotpotQA, per question). *Base LLM Inference* aggregates decomposition, per-hop generation, and synthesis.

Component	ms	%	×Van.
Base LLM Inference	1,986	54.3%	8.9×
Memory Retrieval	497	13.6%	2.2×
Error Detection	681	18.6%	3.0×
Correction Execution	391	10.7%	1.7×
Memory Update	103	2.8%	0.5×
Total (PRISM)	**3,658**	**100%**	**16.3×**

H Per-Dataset Step-Depth Breakdown

Table 10 reports the per-dataset step-depth breakdown for the three decomposing systems (MEMP, ReAgent, PRISM) used to compute the aggregate degradation in Table 6. "3+-step" groups all chains of depth 3 or more. Cells show EM (%) with the per-step pooled sample size n in parentheses. A ‡ marks any cell whose pooled sample size is $n < 30$; such cells are statistically unstable and should not be read as evidence of a per-system advantage on their own. Δ is PRISM − MEMP in percentage points.

The per-dataset view supports three observations. (1) **PRISM's advantage is largest on the deepest benchmark and widens at 3+-step.** On MuSiQue, the PRISM−MEMP gap grows from +14.7 pp at 2-step to +20.9 pp at 3+-step—the expected signature of error correction suppressing late-step error propagation. The PRISM−ReAgent gap at MuSiQue 3+-step is +10.1 pp. (2) **The gap is preserved on HotpotQA** (+13.8 → +12.7 pp) and widens on MultiHopRAG (+2.4 → +3.9 pp). On 2WikiMultiHopQA the 3+-step gap (+3.8 pp) is smaller than 2-step (+5.9 pp) because the 3+-step bucket on 2Wiki is dominated by yes/no comparison questions that hit a ≈ 90% ceiling for *every* decomposing system (MEMP 89.5, ReAgent 89.7, PRISM 93.3). (3) **The 1-step rows are noisier** because decomposers produce 1-step plans only on a small subset of questions ($n \leq 144$ vs. $n \geq 371$ elsewhere). ReAgent emits no 1-step plans on 2Wiki and aggregates only $n = 9$ on HotpotQA 1-step; the 66.7% EM on n=9 has a 95% Clopper–Pearson interval ≈ [30%, 93%] and overlaps both MEMP and PRISM, so we do not read this row as evidence of a ReAgent advantage. Table 6 provides the load-bearing 1-step anchor.

I Full Scalability Table

Table 11 reports the full per-cell F1 for the 8-backbone scalability sweep summarised in Table 7. The two `google/gemma-4-*` models report 5-run averages; the remaining six models report a single 500-sample run per dataset and are marked †.

J Why MuSiQue and MultiHopRAG Absolute Scores Are Low

Both benchmarks depress absolute F1 via structural mechanisms that operate independently of any individual system; the cross-system ordering is therefore the load-bearing quantity in §4.2.

MuSiQue. (1) *Shortcut-resistant compositional construction.* MuSiQue is built by composing 2-, 3-, and 4-hop questions from validated single-hop building blocks with explicit filtering to remove questions answerable by surface heuristics. The per-question success rate is therefore roughly the *product* of per-hop success rates, which compresses absolute scores on strict EM/F1. (2) *Variable-hop chains amplify error propagation.* Unlike HotpotQA/2Wiki which are fixed-2-hop, MuSiQue contains 3- and 4-hop chains where a single misretrieval at any step yields a wrong final answer. The Table 10 breakdown shows MuSiQue 3+-step EM at 25.6% (PRISM) vs. MEMP 4.7% and ReAgent 15.5%. (3) *Hard distractor passages.* MuSiQue's per-question candidate pool intentionally includes topically adjacent paragraphs that do not support the final answer; without explicit step-level error

Table 10: Per-question EM by chain step depth, 5-round pooled ($n \approx 2{,}500$/dataset).

Dataset	Depth	MEMP	ReAgent	**PRISM**	Δ
HotpotQA	1-step	46.2(n=39)	66.7‡(n=9)	50.0(n=34)	+3.8
HotpotQA	2-step	48.2(2,032)	50.8(2,063)	**62.0**(2,053)	+13.8
HotpotQA	3+-step	46.9(429)	56.8(428)	**59.6**(413)	+12.7
2WikiMHQA	1-step	90.5‡(21)	$-^{\ddagger}$(0)	92.3‡(26)	+1.8
2WikiMHQA	2-step	56.1(2,087)	52.6(2,112)	**62.0**(2,103)	+5.9
2WikiMHQA	3+-step	89.5(392)	89.7(388)	**93.3**(371)	+3.8
MuSiQue	1-step	5.7(141)	3.3(60)	**10.4**(144)	+4.7
MuSiQue	2-step	13.1(1,291)	17.7(1,293)	**27.8**(1,300)	+14.7
MuSiQue	3+-step	4.7(1,068)	15.5(1,147)	**25.6**(1,056)	+20.9
MultiHopRAG	1-step	19.3(327)	9.2(109)	**28.5**(291)	+9.2
MultiHopRAG	2-step	18.4(700)	14.0(755)	**20.8**(715)	+2.4
MultiHopRAG	3+-step	12.2(1,473)	11.8(1,636)	**16.1**(1,494)	+3.9

detection, any generator is forced to commit to whichever distractor the retriever surfaces.

MultiHopRAG. (1) *Open-corpus retrieval bottleneck.* MultiHopRAG does not provide per-question gold-document context. The system must dense-retrieve over the full news corpus, and the multi-hop recall ceiling on the corpus is well below 1.0 (the 0.45 saturation reported by Liu et al. (2025)). Any retrieval-conditioned generator inherits this ceiling as a hard upper bound. (2) *Short reference answers inflate token-F1 penalty.* Many MultiHopRAG gold answers are short noun phrases; a single token mismatch can collapse F1 from 1.0 to 0.5 or 0 even when the answer is semantically correct. (3) *News-domain entity ambiguity.* Articles contain many co-referential surface forms for the same entity, but gold annotations select one canonical form; token-overlap signals systematically under-credit semantically correct answers.

Joint reading. The two benchmarks depress absolute scores via different mechanisms but the cross-system PRISM-vs-baseline ordering is preserved on *both*. Treating MuSiQue + MultiHopRAG jointly as "the difficulty-stressed half of the benchmark suite" is the basis for the pooled step-depth analysis in §5.1 (Table 6).

Table 11: PRISM scalability across model backbones (F1 %, n=500/dataset; † = single-run preliminary). Bold marks the best system within each model block.

Model	System	HotpotQA	2WikiMHQA	MuSiQue	MultiHopRAG	Avg
google/gemma-4-31B-it	Vanilla RAG	43.6	31.9	10.1	3.0	22.2
	MEMP	59.0	68.4	12.3	15.0	38.7
	PRISM	**75.1**	**74.8**	**33.5**	**19.0**	**50.6**
google/gemma-4-E4B-it	Vanilla RAG	21.6	12.2	2.9	0.1	9.2
	MEMP	46.3	23.7	4.2	3.8	19.5
	PRISM	**55.8**	**34.8**	**10.0**	**6.4**	**26.8**
Qwen/Qwen3-32B†	Vanilla RAG	56.9	51.8	15.3	6.5	32.6
	MEMP	66.7	68.9	28.9	29.9	48.6
	PRISM	**71.4**	**70.3**	**33.7**	**31.8**	**51.8**
Qwen/Qwen3-8B†	Vanilla RAG	61.4	57.3	18.8	9.7	36.8
	MEMP	68.2	51.1	22.2	31.7	43.3
	PRISM	**69.1**	**67.4**	**28.8**	**33.7**	**49.7**
openai/gpt-oss-120b†	Vanilla RAG	70.8	67.5	28.4	5.2	43.0
	MEMP	70.9	72.7	40.2	38.5	55.6
	PRISM	**73.1**	**73.4**	**41.2**	**42.5**	**57.6**
openai/gpt-oss-20b†	Vanilla RAG	66.8	69.0	26.4	10.9	43.3
	MEMP	69.1	68.1	32.7	**40.7**	52.7
	PRISM	**70.0**	**70.9**	**38.5**	37.4	**54.2**
Llama-3.3-70B-Instruct†	Vanilla RAG	69.3	56.4	15.9	3.9	36.3
	MEMP	72.1	60.1	29.1	42.1	50.9
	PRISM	**72.5**	**67.8**	**41.8**	**44.2**	**56.6**
Llama-3.1-8B-Instruct†	Vanilla RAG	46.4	18.1	7.7	3.6	19.0
	MEMP	46.6	14.2	17.4	36.8	28.8
	PRISM	**49.4**	**22.9**	**22.2**	**38.9**	**33.3**