STORED BUT NOT DELIVERED: BENCHMARKING AND REPAIRING RESEARCH-STATE DISCONTINUITY IN SCIENTIFIC AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In scientific agents, each hypothesis, experiment, and result conditions later decisions, and a decision can depend on evidence produced many steps earlier. An agent can store a hypothesis and the result that refutes it, yet retrieve one without the other, and so repeat a direction it has already ruled out. We formulate this as *research-state discontinuity*: the context delivered for a decision can omit relationships that stored history still holds, and inspecting the store or final task success does not reveal the omission. To assess it we construct the Executable Scientific-Agent Memory Benchmark (ESMB), a controlled diagnostic suite that fixes histories, checkpoints and authored action effects so that only memory varies, and scores a *memory interface* returning labeled evidence separately from a *planner* acting on it. This separation localizes the failure to two operations. Text baselines confine candidates to the current branch, so the decisive record is dropped before ranking, and a record that does reach ranking can still be displaced once the budget fills. We then propose Research State Memory (RSM), which gives every item a branch identity, stores contradiction and supersession as explicit edges, and delivers an item together with the evidence bearing on it. RSM raises joint successes from $6/15$ to $12/15$ on the cued suite, and solves $60/102$ externally authored ScienceAgentBench tasks against repair summary's $46/102$.

1 INTRODUCTION

Large language model (LLM) agents interleave reasoning, tool use, and observations (Yao et al., 2023b), and structure that reasoning over branching or graph-shaped states (Yao et al., 2023a; Besta et al., 2024). Scientific agents extend the loop into iterative research: proposing hypotheses, running experiments, and using results to guide further exploration (Lu et al., 2024; Yamada et al., 2025). Because each step depends on what earlier ones established, such an agent relies on memory to carry that knowledge forward. Storing a record, however, does not ensure that what it establishes reaches the agent when it next acts.

Consider an agent investigating stalled training. A learning-rate sweep yields no improvement, so the agent considers a batch-size explanation. If a later context retrieves the learning-rate hypothesis without the negative result that weakens it, the agent may revisit an unsuccessful direction, spending an experiment on a question it has already answered. Both records remain stored while their decision-relevant connection is lost. We call this *research-state discontinuity*. Figure 1 shows two contexts drawn from the same history, only one of which includes the result needed to interpret the hypothesis.

Current scientific-agent benchmarks score experimentation and executable outcomes (Huang et al., 2024; Chen et al., 2024), but do not test whether earlier evidence was available when needed. We propose the *Executable Scientific-Agent Memory Benchmark* (ESMB) to characterize this problem. It separates a *memory interface*, which returns a bounded context of labeled *fragments* (each a stored record rendered as short text), from a *planner*, which receives that context with the action menu. ESMB tests whether the required evidence is delivered in four controlled settings: *Fixed* tests decision-time delivery with a live planner; *TaskVariant* changes task wording while preserving the same evidential dependencies; *Replay* removes the planner by replaying fixed trajectories, actions, and outcomes to isolate memory behavior; and *LongHorizon* increases the lag between writing evidence

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

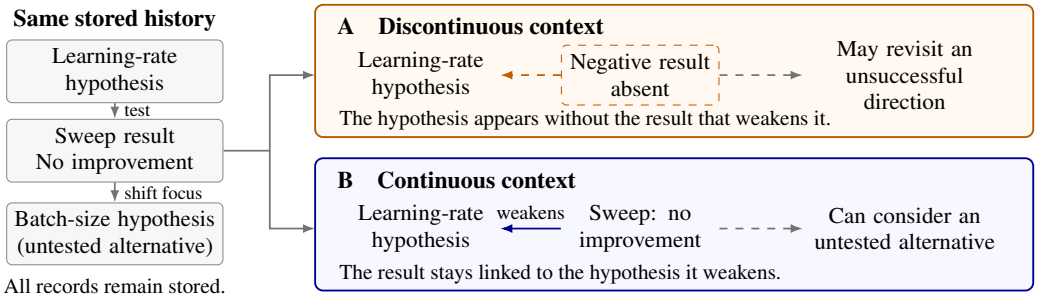


Figure 1: Research-state discontinuity: the same stored history can yield contexts with different usable evidence. Solid links show history or evidence; the orange dashed link marks missing evidence, and gray dashed arrows indicate possible consequences.

and needing it while hiding source labels from the planner. ESMB localizes memory failure to two stages of evidence delivery: exclusion before ranking and displacement after candidate selection. A lexical-preview summary baseline fails whenever the required evidence lies on another branch and succeeds when it stays on the current one. The first failure occurs before ranking, because the text baselines restrict candidates to the current branch whenever it contains records and so exclude the refuting result before any relevance score is computed. *Replay* shows a second failure after candidate selection, where a decisive result that enters the candidate set can still be outranked or displaced when the context budget fills.

As a mitigation, we propose *Research State Memory* (RSM), a branch-aware relational memory that links typed scientific items through explicit evidential relations and assembles them into bounded decision contexts. It stores objects such as hypotheses, experiments, results, and artifacts as typed items with branch identities, keeping earlier lines of investigation addressable after the workflow moves elsewhere. Eight relation types encode dependencies such as contradiction, supersession, and derivation explicitly, so a text ranker need not recover them from wording. At retrieval time, RSM bundles related evidence with the primary item inside a fragment, so a hypothesis can reach the planner with the result that changes its interpretation. The results consistently favor RSM across controlled and external evaluations. On annotation-cued *Fixed* with `gpt-5.2`, RSM achieves $12/15$ joint successes versus $6/15$ for lexical-preview summary, with gains concentrated on decisions requiring cross-branch evidence. The improvement persists across planner models and task surfaces, reaching $15/15$ on *TaskVariant*, and extends to *LongHorizon*, where RSM improves over hashed-vector retrieval by $10/55$ successes under both planner models. Beyond ESMB, RSM solves $60/102$ externally authored ScienceAgentBench (Chen et al., 2024) tasks compared with $46/102$ for repair summary.

The contributions of this paper are:

- We identify and formalize *research-state discontinuity*, a failure mode in which scientific agents retain prior evidence but lose the relationships needed to use it at decision time. Agents may then revisit refuted hypotheses or repeat unproductive experiments.
- We introduce *ESMB*, a controlled diagnostic benchmark for research-state continuity that separates memory delivery from planning and execution. To our knowledge, it is the first benchmark to localize scientific-agent memory failures to specific stages of evidence delivery, such as exclusion before ranking and displacement under a bounded context.
- We propose *RSM*, a relational memory that restores decision-time access to evidence across branches and long research trajectories. RSM improves evidence-grounded decisions on ESMB and transfers beyond the controlled setting, solving $60/102$ externally authored ScienceAgentBench tasks versus $46/102$ for repair summary.

2 Research-State Discontinuity

Figure 1 illustrates the failure we study. Both contexts come from the same stored history. In panel B, the learning-rate hypothesis is returned with the flat sweep that weakens it, whereas in panel A the

hypothesis appears without this result and can look viable again. We call the identities, statuses, and evidential relationships needed to interpret prior research outcomes the *research state*.

Definition. At decision checkpoint t, let H_t denote the records stored in memory and q_t the corresponding context request. Let $\mathcal{M}_B$ denote a memory interface with context budget B, so the returned context is

$$C_t = \mathcal{M}_B(H_t, q_t). \tag{1}$$

For any collection of records X, let $\rho(X)$ denote the research-state facts recoverable from X, and let D_t denote the research-state facts required for the decision at checkpoint t. Research state is *continuous* when $D_t \subseteq \rho(C_t)$. We define *research-state discontinuity* as

$$D_t \subseteq \rho(H_t) \qquad \text{and} \qquad D_t \not\subseteq \rho(C_t). \tag{2}$$

Thus, the required state remains recoverable from stored history but is lost before decision time. The definition is representation-agnostic: a summary is continuous if it preserves the same decision-relevant qualification without an explicit graph. If $D_t \not\subseteq \rho(H_t)$, the failure precedes memory delivery.

Where discontinuity occurs. For the record-based memories studied here, let $\kappa(H_t, q_t)$ be the candidate-selection operator and define

$$K_t = \kappa(H_t, q_t), \qquad C_t = \mathrm{top}_B(\mathrm{rank}(K_t)), \tag{3}$$

where K_t is the candidate set, $\mathrm{rank}(\cdot)$ orders the candidates, and $\mathrm{top}_B(\cdot)$ keeps the highest-ranked records that fit the context budget B. Let $R_t \subseteq H_t$ denote the source records needed to recover D_t. A required record $r \in R_t$ can fail in two ways: *exclusion*, when $r \notin K_t$ and it never reaches ranking; or *displacement*, when $r \in K_t \setminus C_t$ because it loses during ranking or bounded packing. Section 4 shows both failure modes in existing memories.

Components of research state. Continuity requires preserving three aspects of research state: *identity*, which keeps earlier lines of investigation addressable; *status*, which distinguishes live hypotheses from those weakened or superseded by evidence; and *dependency*, which preserves how records qualify one another.

3 ESMB: Evaluating Research-State Continuity

3.1 Benchmark Construction

ESMB is a replayable evaluation protocol. Each task specifies (i) typed research-state writes, (ii) decision checkpoints with context requests, and (iii) a closed action set with effects authored in advance. The task specification, action effects, and non-memory inputs are fixed across memory methods; methods differ in the context they deliver, after which the planner acts under the same benchmark contract.

ESMB separates *memory delivery* from *planner use*. At each checkpoint, the memory interface receives the stored history and a context request and returns at most B fragments, each containing text and source labels. Under the *annotation-cued* contract, the request also carries benchmark annotations (focus IDs, required labels, workflow phase, and expected item kind) that identify decision-relevant state on the memory side. The planner has no direct access to the store or these memory-request annotations. Instead, it receives the returned context together with benchmark-fixed decision information, including the checkpoint description, recent observation, and closed action menu. Full task and interface specifications are given in Appendix B.

Figure 2 illustrates the protocol with a failed-experiment recovery task. The stored history contains a learning-rate hypothesis `h_lr` on *main* and a batch-size alternative `h_batch` with evidence `ev_batch` on *alt*. Evaluating `h_lr` produces the null result `r_lr_null` on *main*. At the later checkpoint `ctx_recovery` on *alt*, the required state spans both branches: the memory must recover `r_lr_null` together with the local alternative and its evidence within a three-fragment budget. The planner then selects from the fixed action menu using the returned context and benchmark-supplied decision information. Additional worked examples are provided in Appendix I.

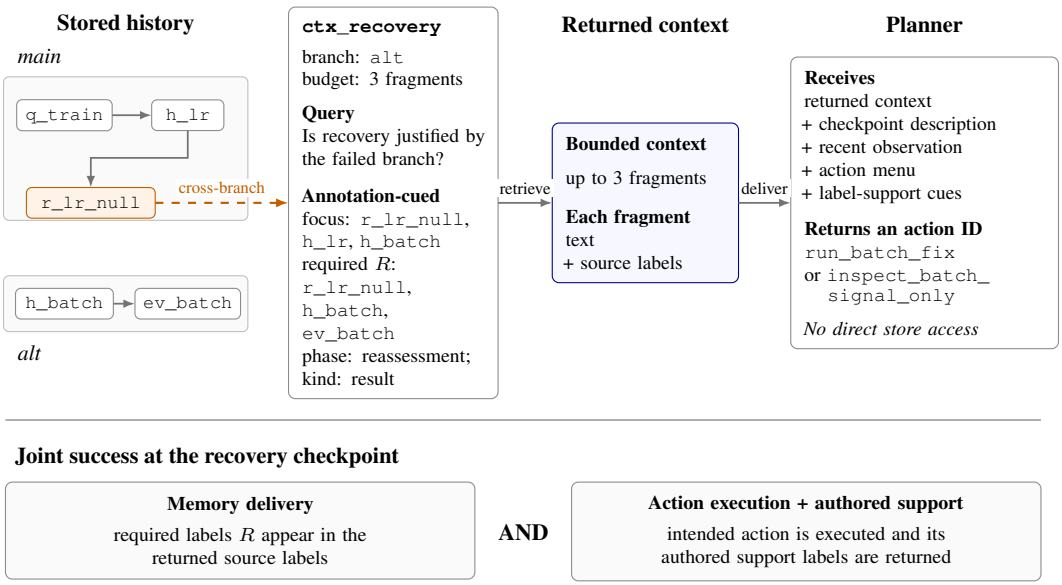


Figure 2: ESMB failed-experiment recovery under the annotation-cued contract. The null result `r_lr_null` is produced on *main* but required at `ctx_recovery` on *alt*, creating a cross-branch dependency (orange dashed arrow). The memory interface returns a bounded context to the planner, while joint success requires both prescribed evidence delivery and the intended action with its authored support. Gray arrows indicate workflow order. Prefixes denote `q_` question, `h_` hypothesis, `r_` result, `ev_` evidence, and `ctx_` context request.

Table 1: ESMB evaluation suites. *Fixed* is the reference contract; each other suite changes one primary dimension. Sizes denote task designs $\times$ repeated executions.

Suite	Change	Contract	Runs	Readout
Fixed	-	Cued memory; planner label-support cues	5×3	Required evidence + intended action with authored support
TaskVariant	**Task surface**	Same contract; evidential dependencies unchanged	5×3	Same joint-success predicate as Fixed
Replay	**No planner**	Annotated memory; fixed trajectories, actions, and outcomes	6 traces	Recall, grounding, contamination, recovery, and footprint
LongHorizon	**Write–use distance**	Annotated memory; opaque citation-based planner	11×5	Gold action + decisive evidence in a cited fragment

3.2 Evaluation Suites

ESMB comprises four controlled suites (Table 1). *ESMB-Fixed* is the reference condition and contains five task families: failed-experiment recovery, branch switching, supersession, conflicting evidence, and artifact-grounded updating. The first three require decisive evidence from another branch; the latter two are local controls whose required state remains on the current branch. *ESMB-TaskVariant* changes only task surface form while preserving the underlying histories and evidential dependencies. *ESMB-Replay* removes the planner entirely and replays fixed trajectories, actions, and outcomes, isolating memory behavior; its six traces fix 17 context requests with budgets of two to four fragments, so every method is scored on identical requests. *ESMB-LongHorizon* increases the delay and structural separation between decisive evidence and its later use, expands the context budget to six fragments, and evaluates the planner under an opaque, citation-grounded interface (Appendix F). Three raters recruited separately from the authors audited all 16 Fixed, TaskVariant, and Replay episodes; by majority vote, every episode was judged valid and 22 of 23 scoring-relevant checkpoints free of wording leakage (Appendix B.4).

Table 2: Baseline results on annotation-cued *Fixed* and deterministic *Replay*.

Memory method	Fixed	Replay recall	Replay recovery
No memory	$0/15$	0.000	0.000
Lexical-preview summary	$6/15$	0.956	0.333
Lexical fragment retrieval	$3/15$	0.744	0.000

Table 3: Lexical-preview summary on *Fixed*, split by task family.

Task family	Evidence	Success
Failed-experiment recovery	Cross	$0/3$
Branch switching	Cross	$0/3$
Stale supersession	Cross	$0/3$
Conflicting evidence	Local	$3/3$
Artifact-grounded update	Local	$3/3$

4 Diagnosing Existing Memories

4.1 Baselines and Evaluation Contract

We characterize the three baselines reported in Table 2. *No memory* provides the lower bound. *Lexical-preview summary* and *lexical fragment retrieval* both select records by lexical overlap, but differ in packaging: the former groups compact previews, whereas the latter returns individual record fragments. Neither uses a learned retriever or summarization model. Both apply a hard same-branch filter whenever the current branch contains records (Appendix C.2).

Fixed uses the annotation-cued contract of Section 3, with a live planner and a three-fragment budget. *Replay* removes planner variation by fixing trajectories, actions, and outcomes, so differences reflect only the context returned by memory (metrics in Appendix E).

4.2 Failure Patterns

Table 2 summarizes performance on *Fixed* and *Replay*. Together with the family breakdown in Table 3, the results expose the two failure modes defined in Section 2: required records can be *excluded* during candidate selection or *displaced* after entering the candidate set.

Exclusion during candidate selection. Both lexical baselines restrict candidates to the current branch whenever it contains records. At `ctx_recovery`, for example, the current `alt` branch contains `h_batch` and `ev_batch`, so the decisive `r_lr_null` on `main` never enters the candidate set K_t. The same holds for all three cross-branch families, on which lexical-preview summary scores $0/3$ each, whereas it scores $3/3$ on both families whose required evidence remains on the current branch (Table 3). In the terms of Section 2, the record stays in H_t but $r \notin K_t$, so no downstream ranking can recover it.

Displacement after candidate selection. *Replay* exposes the second failure without planner variation. Lexical-preview summary reaches 0.956 recall and 0.333 recovery, whereas lexical fragment retrieval reaches 0.744 recall and 0.000 recovery. Here, a record can enter the candidate set and still be outranked by competing records or cut by the fragment budget, so $r \in K_t \setminus C_t$. The lower recall and recovery of lexical fragment retrieval show that, once records are candidates, ranking and packing decide what reaches the planner.

5 Research State Memory

Research State Memory (RSM) is a branch-aware relational memory designed to address the two delivery failures in Section 2. To reduce *exclusion*, RSM preserves branch identity and can retrieve decision-relevant neighborhoods across branches. To reduce *displacement*, it bundles each item with its linked evidence. RSM changes only the memory write and read paths; the planner, action set, and task remain unchanged. Figure 3 summarizes the resulting loop: workflow signals are written into a typed research state, explicit relations preserve scientific dependencies, bounded assembly produces planner-visible context, and action outcomes return as subsequent observations.

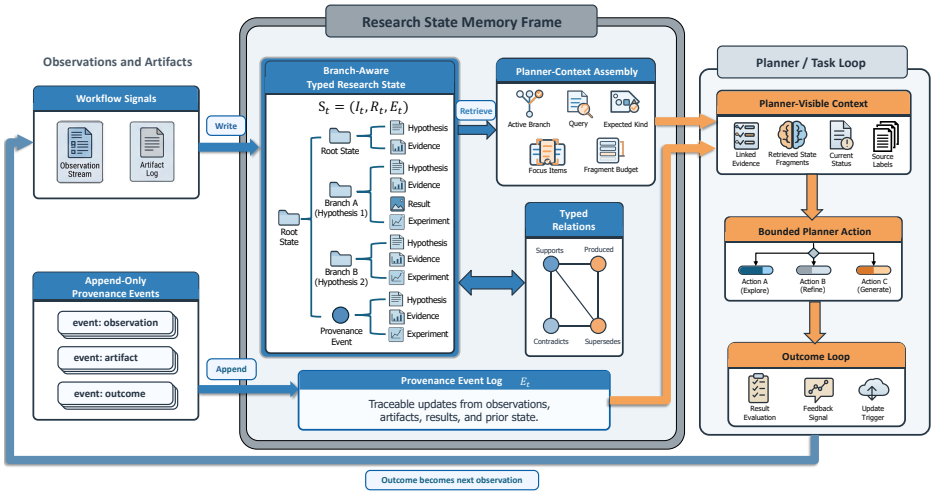


Figure 3: Overview of Research State Memory (RSM). Workflow signals update a branch-aware typed state $S_t = (I_t, R_t, E_t)$ with explicit relations and append-only provenance; planner-context assembly then retrieves and bundles linked state into a bounded context, and task outcomes feed back as new observations.

Write: branch-aware typed state and provenance. On the write path (Figure 3, left and center), a workflow signal produces a typed write containing an item, its relations, and optional artifact content. Items represent open questions, hypotheses, experiments, evidence, results, decisions, and artifacts, with schema-validated payloads, branch identities, and status fields. At checkpoint t, RSM maintains

$$S_t = (I_t, R_t, E_t), \qquad S_{t+1} = \mathcal{U}(S_t, \mathcal{W}(o_t)), \tag{4}$$

where I_t, R_t, and E_t denote typed items, relations, and append-only provenance events. Each write is recorded in the provenance log before its state update is materialized, making stored state traceable to the event that introduced or modified it. The branch folders in Figure 3 are conceptual: in the implementation, branch identity is carried directly by each item, so earlier lines of investigation remain addressable after the workflow moves elsewhere.

Link: typed relations. RSM stores dependencies as typed edges (Figure 3, *Typed Relations*), using eight relation types: `tests`, `produces`, `supports`, `contradicts`, `answers`, `has-artifact`, `supersedes`, and `derived-from`. These edges preserve, for example, which result contradicts a hypothesis or which artifact supports a result. Because a link is stored rather than inferred, it holds even when the two records share no wording, so a hypothesis retrieved later can carry the result that contradicts it as a neighbor in its fragment, as in Figure 1(B). Supersession adds an edge and leaves the earlier item in place, so both that item and the evidence against it remain recoverable.

Retrieve: assemble linked state under a budget. On the read path (Figure 3, *Planner-Context Assembly*), a request specifies the active branch, query, expected kind, focus items, and fragment budget. RSM first retrieves branch-local questions, hypotheses, experiments, and results, together with recent events and the one-hop neighborhoods of focus items. Candidates are ordered by branch match, focus adjacency, relation signal, expected kind, lexical overlap, importance, and recency. Representative state (an unresolved question, an active hypothesis, and a recent result) is reserved before the remaining budget is filled, with typed items preferred over raw events. RSM returns *fragments*, each containing one primary item, up to two related neighbors, and their source labels, so supporting or contradicting evidence arrives with the item it bears on and does not need a fragment slot of its own. Under annotation-cued requests, RSM additionally uses the supplied focus, phase, and required-label information to recover relevant cross-branch state and provenance that branch-local retrieval may miss; required-label backfill can promote already-candidate records whose prescribed labels are absent. The resulting bounded fragments form the planner-visible context shown in Figure 3; full routing, ranking, and backfill details are given in Appendix C.1.

Table 4: Controlled ESMB results. (a) Fixed joint successes and grounding with `gpt-5.2`, alongside Replay recall. (b) Comparator → RSM successes; n gives the denominator. The comparator is lexical-preview summary for Fixed/TaskVariant and hashed-vector retrieval for LongHorizon. Executions are nested within task designs; memory annotations remain available.

(a) Fixed and Replay

Method	Fixed	Ground.	Recall
No memory	0/15	0.000	0.000
Lexical-preview summary	6/15	0.900	0.956
Lexical fragment retrieval	3/15	0.850	0.744
RSM	**12/15**	**1.000**	**1.000**

(b) Cross-model joint success

Suite	`gpt-5.2`	`gpt-4.1`
Fixed ($n = 15$)	$6 \rightarrow 12$	$6 \rightarrow 14$
TaskVariant ($n = 15$)	$6 \rightarrow 15$	$6 \rightarrow 15$
LongHorizon ($n = 55$)	$35 \rightarrow 45$	$45 \rightarrow 55$

6 Evaluating RSM as a Mitigation

6.1 Controlled ESMB Comparisons

For *Fixed* and *TaskVariant*, *joint success* requires the intended action to be executed and returned labels to cover both checkpoint-required sources and the action's authored support (Appendix B.2). *Grounding* measures mean required-label coverage independently of action choice. *Replay recall* averages relevant-label coverage within each trajectory and then across trajectories.

Evidence delivery across planners. On annotation-cued *Fixed* with `gpt-5.2`, RSM doubles joint success from $6/15$ to $12/15$ and improves grounding from 0.900 to 1.000 relative to lexical-preview summary (Table 4(a)). Each of the three families requiring cross-branch evidence improves from $0/3$ to $2/3$, while both local controls remain at $3/3$. Across both planners and task surfaces, RSM records three family-level wins, two ties, and no losses against summary, reaching $15/15$ on *TaskVariant* with either planner (Table 4(b); per-family results in Appendix D). In every planner and suite cell, all discordant task-trials favor RSM. Summary cannot exceed $6/15$, because its branch filter removes the decisive record at every cross-branch checkpoint whatever the planner selects. RSM's grounding is 1.000 in every cell, so its remaining Fixed failures come from the planner's action choice, with the required evidence delivered (Appendix D). Replay recall likewise improves to 1.000, versus 0.956 for summary and 0.744 for lexical fragment retrieval, while RSM's contamination stays at 0.000 because other-branch results arrive as neighbors of same-branch items. At the stale-supersession trace's `ctx1`, for example, lexical fragment retrieval returns the old hypothesis without the result that refutes it, which ranks last of five candidates, whereas RSM returns the hypothesis with that result attached through its `contradicts` edge, as in Figure 1(B) (Appendix E.2). These comparisons retain focus and workflow annotations that RSM uses and the lexical baselines ignore. Appendices E and I give complete Replay diagnostics and additional examples.

Long-horizon delivery. *LongHorizon* requires the intended action and decisive-source support in planner-cited fragments at the evaluated checkpoints. RSM achieves $45/55$ versus $35/55$ with `gpt-5.2`, and $55/55$ versus $45/55$ with `gpt-4.1`, against hashed-vector retrieval (Table 4(b)). Switching planners moves both methods by ten successes, leaving RSM's gain at 18.2 percentage points under each. The nine positive tasks place the decisive record two to seven checkpoints before the decision that needs it. Source labels are hidden from the planner, but memory-side focus IDs remain available, so the suite tests delayed delivery of evidence that the annotations already identify (Appendix F).

Assembly ablations. Disabling RSM's context-assembly switches one at a time on Replay (Appendix E.3) traces its recall advantage to the routes that repair exclusion. Without supplemental retrieval, recall falls from 1.000 to 0.944, and all three lost labels are exclusions that never enter the candidate list. Item-first selection keeps the context on the request branch; without it, contamination rises from 0.000 to 0.096 as unlabeled events fill the remaining slots. Removing two-stage representative selection changes no aggregate metric.

Table 5: Public-request comparisons and native-task transfer. (a) Joint successes out of 15 in each suite; the A-MEM and StructMem comparisons use separate contracts. (b) Native successes by domain, with N tasks per domain; "Summary" denotes repair summary. RSM entries are highlighted in bold.

(a) Public-request joint success

Method	Fixed	Variant
A-MEM comparison		
A-MEM	$3/15$	$0/15$
Lossless summary	$6/15$	$5/15$
Hashed-vector retrieval	$9/15$	$0/15$
RSM	$\mathbf{3/15}$	$\mathbf{3/15}$
StructMem comparison		
StructMem	$1/15$	$2/15$
Lossless summary	$2/15$	$3/15$
RSM	$\mathbf{6/15}$	$\mathbf{7/15}$

(b) ScienceAgentBench transfer

Domain	N	Summary	RSM
Computational chemistry	20	9	**11**
Bioinformatics	27	12	**16**
Geographical information science	27	12	**16**
Psychology and cognitive science	28	13	**17**
Total	**102**	**46**	**60**

6.2 Published Memory Comparisons

Public requests remove memory-side annotations and planner label-support cues. Within each comparison, methods share the event stream, planner, actions, and a three-fragment, 4,096-byte context budget. We report Fixed/TaskVariant joint success over 15 scheduled executions per suite, retaining parser/execution failures.

RSM ties official A-MEM (Xu et al., 2025) on Fixed and exceeds it on TaskVariant (Table 5(a)). Against adapted StructMem (Xu et al., 2026), RSM gains 33.3 percentage points on both suites; its advantage over StructMem and lossless summary remains positive after removing any single task family. However, lossless summary outperforms RSM in the A-MEM comparison, so these results do not establish uniform superiority over text memories. The two comparisons use separate contracts and should be read within panels (Appendix G).

6.3 Transfer Beyond ESMB

We evaluate the 102-task verified split of ScienceAgentBench (Chen et al., 2024), with a shared initial program and up to three repairs per task. Each task yields one trajectory per method; success means that the final program passes the native evaluator. The comparator, *repair summary*, stores full records and preserves latest-attempt context. Both methods use a single-branch wrapper with a 6,000-token memory cap and a 2,000-token feedback cap, without ESMB annotations.

RSM solves $60/102$ tasks versus $46/102$ for repair summary, a paired gain of 13.7 percentage points (95% task-bootstrap interval $[3.9, 23.5]$ points; exact two-sided McNemar $p = 0.0161$). Of the 30 discordant tasks, 22 favor RSM and eight favor repair summary. Gains are positive in all four domains (Table 5(b)), supporting transfer to single-branch program repair without annotation cues. Appendices H.1 and H.2 give detailed outcomes and exploratory pilots.

7 Related Work

Agent memory. Agent memory supports retrieval, persistent experience, and long-context management (Lewis et al., 2020; Park et al., 2023; Wang et al., 2024; Zhong et al., 2024; Shinn et al., 2023; Packer et al., 2023; Wang et al., 2023). Recent systems add richer structure: StructMem organizes events, TierMem separates summaries from evidence, and A-MEM links evolving notes (Xu et al., 2026; Zhu et al., 2026; Xu et al., 2025); comparative studies find that memory structure affects agent performance (Zeng et al., 2024; Shutova et al., 2026). Provenance standards and workflow tools record the entities, activities, and agents behind each result (Moreau & Missier, 2013; McPhillips et al., 2015). RSM instead targets *research-state continuity*. It keeps branch identity, evidential

relations, and append-only provenance explicit and assembles linked state into bounded planner context.

Memory evaluation. Long-term memory benchmarks test recall from stored histories (Wu et al., 2025; Maharana et al., 2024; Hu et al., 2025), long-context benchmarks test reasoning over scattered facts (Kuratov et al., 2024), and executable benchmarks measure downstream task success (Huang et al., 2024; Chen et al., 2024; Chan et al., 2025; Jimenez et al., 2024; Siegel et al., 2025). These settings generally do not isolate whether failure comes from missing evidence at decision time or from planner misuse. ESMB separates memory delivery from planning and localizes failures to exclusion before ranking and displacement after candidate selection.

Scientific agents. Systems such as ChemCrow, The AI Scientist, Robin, and AI Scientist-v2 integrate reasoning, experimentation, tools, and persistent research history (Bran et al., 2024; Lu et al., 2024; Ghareeb et al., 2026; Yamada et al., 2025). Their outcomes jointly depend on planning, search, tools, and memory. We focus more narrowly on the memory interface, testing whether prior scientific state remains usable at later decisions and evaluating RSM both in controlled ESMB settings and on native ScienceAgentBench tasks.

8 Limitations

ESMB is intentionally small and controlled; it is designed to localize failures in research-state delivery and does not measure broad scientific-agent capability. Its Fixed and TaskVariant suites contain five authored task families with repeated executions, so the reported task-trial counts should not be interpreted as independent-task sample sizes. Although a retrospective human audit supports the coherence of these tasks (Appendix B.4), ESMB and RSM were co-designed, and external transfer is therefore evaluated separately.

The annotation-cued ESMB contract also supplies memory-side focus, phase, kind, and required-label information. RSM uses these signals whereas the text baselines use only a subset, so the controlled results measure the ability to *deliver prescribed research state*, not autonomous discovery of which historical evidence will matter. The public-request comparisons remove these cues and show that method rankings depend on the information contract (Section 6.2; Appendix G). LongHorizon likewise retains memory-side focus IDs, so its results concern delayed delivery of identified evidence; unconstrained long-range retrieval is untested.

Finally, ESMB checks only the presence of source labels, not the semantic completeness of the rendered text. Its writes are authored typed events; none are induced autonomously from raw scientific interaction. ScienceAgentBench mitigates some of these constraints by evaluating native program execution without ESMB annotations, but it uses a single-branch repair setting. We leave evaluation on naturally evolving, multi-branch scientific workflows to future work.

9 Conclusion

We introduced *research-state discontinuity*: a scientific agent can retain its history while losing the identities, statuses, and evidential relationships needed to use that history at decision time. ESMB makes this failure measurable by separating memory delivery from planner behavior and identifies two failure modes: exclusion before ranking and displacement during bounded context assembly. To mitigate these failures, we proposed *Research State Memory* (RSM), a branch-aware relational memory that preserves scientific dependencies and delivers linked evidence as bounded decision context. RSM improves controlled ESMB performance across planner models and task surfaces, remains competitive under public memory comparisons, and transfers to ScienceAgentBench, solving $60/102$ tasks versus $46/102$ for repair summary. Because ESMB scores delivery separately from action choice and Replay logs each method's candidate set, it can attribute a missing record to exclusion or displacement, which end-task success cannot reveal. These results suggest that long-horizon scientific agents need memory that delivers the research state required for the next decision.

REPRODUCIBILITY STATEMENT

The benchmark construction, evaluation contracts, and scoring predicates needed to reproduce the reported comparisons are specified in the appendix. ESMB's task template and suite inventory are in Appendix B, the deterministic Replay metrics and their denominators in Appendix E, and the joint-success predicate with the contract identities, decoding settings and byte caps in Appendix B.2. The horizon construction is in Appendix F, RSM's state fields and context-assembly sequence in Appendix C.1, and worked examples for each ESMB suite in Appendix I.

We intend to release the benchmark code, task fixtures, method implementation, and evaluation summaries. Run records, caches, and the adapter and planner configurations of the LLM-planner suites are not part of that release, so the reported planner outcomes cannot be independently reproduced from it. ESMB-Replay is deterministic and has no planner, so its reported metrics follow from the fixtures alone and are reproducible from the release.

ETHICS STATEMENT

This work evaluates memory interfaces for scientific agents on executable benchmark tasks and externally authored programming tasks. It involves no human subjects and no personally identifying data. The human validation reported in Appendix B.4 is an annotation audit of benchmark episodes by blinded raters; it collects judgments about benchmark material, not data about people.

Explicit provenance and typed research state make it easier to audit why an agent chose an experiment, retired a branch, or reused earlier evidence. A memory that preserves evidence faithfully will also re-deliver incorrect evidence, and its provenance trail may lend that evidence unwarranted authority. Typing and provenance do not prevent incorrect writes, repeated mistakes, or false scientific conclusions, so human oversight of scientific claims remains necessary. The intended release is code, benchmark specifications, evaluation results, and documentation, not model weights or a scraped dataset, and it must respect the upstream access and redistribution terms recorded in Appendix J.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks that require disclosure under the ICLR 2027 AI Policy for Authors: implementing methods, that is, writing and modifying the benchmark, method, and evaluation code; designing and giving feedback on the research methodology and experiments; cleaning and reformatting datasets, namely preparing external-benchmark inputs for our evaluation wrappers; and interpreting results. We have not used generative AI tools to propose or refine hypotheses, to develop theoretical models or conceptual frameworks, to generate synthetic data sets, or to assist with translation; research ideation and hypothesis generation, including the formulation of research-state discontinuity, were carried out by the authors without generative AI. Formulating mathematical claims, proving them, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: creating and editing software code, drafting parts of the paper and editing it for readability, suggesting its structure, creating figures and tables from results we had already obtained, identifying relevant literature and formatting references, and checking the manuscript for internal consistency between reported values and their supporting appendices.

We have reviewed all AI-assisted work. All AI-written or AI-modified code was reviewed by the authors and checked with the repository's unit and reference tests. Methodological suggestions were adopted only after the authors evaluated them, reformatted external-benchmark inputs were scored with each benchmark's native evaluator, and AI-assisted analyses and interpretations were rechecked by the authors. Every reported value comes from a recorded result or from a deterministic computation over recorded results, and was checked by an author against its source record. The language models evaluated in this paper are objects of study rather than authoring tools and are reported as part of the experimental setup. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. doi: 10.1609/aaai.v38i16.29720. URL https://doi.org/10.1609/aaai.v38i16.29720.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8. URL https://www.nature.com/articles/s42256-024-00832-8.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Mądry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024. URL https://arxiv.org/abs/2410.05080.

Ali E. Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J. Szostkiewicz, Dmytro Shved, Gavin J. Gyimesi, Jon M. Laurent, Samantha M. Wright, Muhammed T. Razzak, Andrew D. White, Silvia C. Finnemann, Michaela M. Hinks, and Samuel G. Rodriques. A multi-agent system for automating scientific discovery. *Nature*, 655:497–505, 2026. doi: 10.1038/s41586-026-10652-y. URL https://doi.org/10.1038/s41586-026-10652-y.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025. URL https://arxiv.org/abs/2507.05257.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://www.swebench.com/.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems*, volume 37, pp. 106519–106554, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c0d62e70dbc659cc9bd44cbcf1cb652f-Abstract-Datasets_and_Benchmarks_Track.html.

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, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob N. Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *CoRR*, abs/2408.06292, 2024. URL https://arxiv.org/abs/2408.06292.

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, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Timothy McPhillips, Tianhong Song, Tyler Kolisnik, Steve Aulenbach, Khalid Belhajjame, R. Kyle Bocinsky, Yang Cao, James Cheney, Fernando Chirigati, Saumen Dey, Juliana Freire, Christopher Jones, James Hanken, Keith W. Kintigh, Timothy A. Kohler, David Koop, James A. Macklin, Paolo Missier, Mark Schildhauer, Christopher Schwalm, Yaxing Wei, Mark Bieda, and Bertram Ludäscher. YesWorkflow: A user-oriented, language-independent tool for recovering workflow information from scripts. *International Journal of Digital Curation*, 10(1):298–313, 2015. doi: 10.2218/ijdc.v10i1.370. URL https://ijdc.net/index.php/ijdc/article/view/10.1.298.

Luc Moreau and Paolo Missier. PROV-DM: The PROV data model. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-dm/.

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. 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*, 2023. doi: 10.1145/3586183.3606763. URL https://research.google/pubs/generative-agents-interactive-simulacra-of-human-behavior/.

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, 2023. 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.

Zachary S. Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Ströbl, and Arvind Narayanan. CORE-Bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2025. URL https://collaborate.princeton.edu/en/publications/core-bench-fostering-the-credibility-of-published-research-throug/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://voyager.minedojo.org/.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. LongMem: Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10813.

Buqiang Xu, Yijun Chen, Jizhan Fang, Ruobin Zhong, Yunzhi Yao, Yuqi Zhu, Lun Du, and Shumin Deng. StructMem: Structured memory for long-horizon behavior in LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 122–146, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-short.12. URL https://aclanthology.org/2026.acl-short.12/.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. doi: 10.48550/arXiv.2504.08066. URL https://arxiv.org/abs/2504.08066.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://react-lm.github.io/.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of LLM agents. *arXiv preprint arXiv:2412.15266*, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, Yanlin Wang, et al. 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://arxiv.org/abs/2305.10250.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. From lossy to verified: A provenance-aware tiered memory for agents. *arXiv preprint arXiv:2602.17913*, 2026.

APPENDIX CONTENTS

A Terminology and Notation **15**

B ESMB Benchmark Specification **17**

B.1 Suites and Task Families 17

B.2 Contracts, Scoring, and Run Settings 19

B.3 Planner Interface 21

B.4 Task Validity: Retrospective Audit 22

C Memory Methods **23**

C.1 RSM State and Context Assembly 23

C.2 Comparators 24

C.3 Inducing Typed Writes from Explicit Workflow Signals 26

D ESMB-Fixed and ESMB-TaskVariant Results **26**

E ESMB-Replay Results **28**

E.1 Metrics and Aggregates 28

E.2 Exclusion and Displacement in Replay 29

E.3 Context-Assembly Ablations 29

F ESMB-LongHorizon **30**

G Public-Request Comparisons and Contract Sensitivity **31**

G.1 A-MEM and StructMem Comparisons 31

G.2 Joint Cue Removal 32

H ScienceAgentBench Transfer and External Pilots **33**

H.1 ScienceAgentBench-102 Protocol and Paired Outcomes 33

H.2 Exploratory External Pilots 34

I Worked Examples **35**

I.1 ESMB-Fixed: Failed-Experiment Recovery 35

I.2 ESMB-Fixed: Stale Supersession 35

I.3 ESMB-TaskVariant: Matched Failed-Experiment Recovery 35

I.4 ESMB-Replay: Exclusion in the Branch-Switching Trace 36

I.5 ESMB-LongHorizon: Failed-Experiment Recovery, Long Variant 37

I.6 ScienceAgentBench Tasks 1 and 8: ClinTox and DKPES 38

J Reproducibility, Compute, and Licenses **39**

A TERMINOLOGY AND NOTATION

Table 6 fixes the symbols, and Table 7 defines the technical terms by pipeline stage, with pointers to their specifications where available.

Table 6: Notation. Main-text symbols follow Sections 2 and 5; appendix symbols index annotated context requests and scoring.

Symbol	Meaning
Main text	
t	Decision checkpoint index.
H_t	Records stored in memory at checkpoint t.
q_t	Context request issued at checkpoint t.
B	Context budget, in fragments.
$\mathcal{M}_B;\ C_t = \mathcal{M}_B(H_t, q_t)$	Memory interface with budget B; the context it returns.
$\kappa;\ K_t = \kappa(H_t, q_t)$	Candidate-selection operator; candidate set.
$\mathrm{rank}(\cdot), \mathrm{top}_B(\cdot)$	Candidate ordering; the highest-ranked records that fit B, so $C_t = \mathrm{top}_B(\mathrm{rank}(K_t))$.
$\rho(X)$	Research-state facts recoverable from a record collection X.
D_t	Research-state facts required for the decision at checkpoint t.
$R_t \subseteq H_t$	Source records needed to recover D_t (Section 2).
$S_t = (I_t, R_t, E_t)$	RSM state: typed items, relations, and append-only provenance events; Section 5 reuses R_t for this relation set.
$o_t;\ \mathcal{W}, \mathcal{U}$	Observation at t; write map and state update, $S_{t+1} = \mathcal{U}(S_t, \mathcal{W}(o_t))$.
Appendix	
q	An annotated context request, used as a subscript.
R_q (or R)	Required labels of request q, fixed by the task author (R at one checkpoint).
L_q	Relevant labels of request q.
T_q (or T)	Source labels carried by the returned context.
U_a	Authored support labels of the intended action a.
A	Execution indicator: 1 if the intended action is executed.
J	Joint success; in Fixed and TaskVariant, $J = \mathbf{1}[R \subseteq T]\,\mathbf{1}[U_a \subseteq T]\,\mathbf{1}[A]$.
$t_w, t_u;\ \Delta = t_u - t_w$	Checkpoints at which the decisive record is written and used; temporal lag, in checkpoints.
G	Structural lag: the per-variant count of task-relevant branch and relation transitions between the decisive record and its use, assigned by the LongHorizon generator.

Table 7: Terminology, grouped by pipeline stage.

Term	Meaning in this paper
Benchmark construction and evaluation units	
Suite	One of ESMB's four evaluation settings, Fixed, TaskVariant, Replay and LongHorizon (Appendix B.1).
Task family	One of the five authored task types, each with one task in Fixed and one in TaskVariant (Table 9); separately authored Replay traces are prefixed "Replay".
Cross-branch family	Failed-experiment recovery, branch switching, or stale supersession, whose decisive record lies on a branch other than that of the scored decision (Appendix B.1).
Local control	Conflicting evidence or artifact-grounded update, whose required support lies on the current branch (Appendix B.1).
Task surface	A task's wording and presentation, which TaskVariant restates while keeping the dependencies that R and U_a encode (Appendix B.1).
Frozen history	The benchmark-authored writes and observations before a checkpoint, identical for every method, so methods differ only in what they return (Appendix B.1).
Branch	A line of investigation, named on every record by an identifier such as `main` or `alt`.
Checkpoint; decision checkpoint	A point t where memory is queried and the planner acts; Fixed/TaskVariant tasks have a one-action first and a scored two-action decision checkpoint (Appendix B.1).

continued on the next page

Table 7 (continued)

Term	Meaning in this paper
Context request	What memory receives at a checkpoint (q_t, or q): branch, query, purpose, fragment budget, and any annotations (Appendix B.2).
Closed action set	The pre-specified actions permitted at a checkpoint, e.g. `run_batch_fix`.
Authored action effect	The observation, writes, and support labels U_a that an action produces, written in advance by the task author (Appendix B.1).
Intended (gold) action	The action the task author designates as correct (Table 10).
Planner	The language model that picks an action ID without store access, shared across methods; `gpt-5.2` runs are primary, `gpt-4.1` runs robustness runs (Appendix B.2).
Execution (task-trial)	One run of one task design with one method and planner; $12/15$ pools three executions of each of five designs, not 15 independent tasks (Appendix B.2).
Research state and its representation	
Research state	The identities, statuses, and evidential relationships needed to interpret prior research outcomes (Section 2).
Research-state discontinuity	Required state recoverable from stored history but not from the returned context: $D_t \subseteq \rho(H_t)$ and $D_t \not\subseteq \rho(C_t)$ (Section 2).
Exclusion	Loss of a required record before ranking: $r \in R_t$, $r \notin K_t$ (Appendix E.2).
Displacement	Loss of a required candidate to rank or budget: $r \in K_t \setminus C_t$ (Appendix E.2).
Item	A typed record of RSM's state, with fields including kind and status (Appendix C.1).
Kind	An item's type: open question, hypothesis, experiment, evidence, result, decision, or artifact (a stored file); Fixed and TaskVariant have no experiment or decision items.
Status	An item's standing: active, superseded, or archived (Appendix C.1).
Relation	A typed, directed edge between two items, of eight types (Appendix C.1).
Provenance event	An append-only record of a write, to which stored state is traced (Appendix C.1).
Fixture label convention	Prefix gives the kind (`q`, `h`, `exp`, `ev`, `r`, `art`: question, hypothesis, experiment, evidence, result, artifact), as in `r_lr_null`; `d1` and `cp01_state` are unprefixed; Fixed and Replay requests are `ctx`…, LongHorizon checkpoints `cp01`, …
Request contracts and context assembly	
Annotation-cued contract	Requests also carry focus IDs, R, phase, and expected kind, so success measures only the delivery of evidence the request prescribes (Table 11).
Public contract	No request annotations or label-support cues, and context capped at three fragments and 4,096 UTF-8 bytes (Appendix G).
Joint cue removal	A contract-sensitivity test removing the memory-side annotations and the label-support cues together (Appendix G.2).
Focus ID	An annotated item identifier from which RSM reaches the item's one-hop neighborhood (Appendix C.1).
Expected kind	The annotated kind of the decisive item, a ranking signal for RSM and for the lexical and hashed-vector rankers (Appendix C.2).
Workflow phase	An annotation naming the checkpoint's stage, such as branch reassessment, that gates some RSM supplements (Appendix C.1).
Required labels	R_q: author-fixed source labels the returned context must contain (Appendix B.2).
Relevant labels	L_q: the labels of all records annotated as relevant, which may differ from R_q (Appendix E).
Label-support cues	Planner-side cues showing which of each action's listed labels were returned, with an instruction to prefer the better-supported action (Appendix B.3).
Candidate	A record admitted by candidate selection (K_t) and hence eligible for ranking.
Branch filter	The hard same-branch candidate restriction, applied whenever the request branch holds a record. Lexical-preview, rolling-preview, lossless, and repair summary, lexical fragment retrieval, raw-event retrieval, and truncated history apply it (Appendix C.2).
Fragment	A unit of returned context: text plus its records' source labels (Appendix C.2).
Fragment budget	The maximum number of fragments per request (Table 11); it does not equalize the records or text delivered (Appendix C.2).
Neighbor	An item attached inside an RSM fragment, without a slot of its own (Appendix C.1).
Supplementation and backfill	RSM's annotation-gated routes adding records after selection (phase-gated attachment, artifact lineage, required-label backfill), inactive under public requests (Appendix C.1).
Scoring and metrics	
Source label	A record's fixture label, carried by its fragment; scoring checks only the label's presence (Appendix B.2).
Joint success	The scored outcome of the planner-driven ESMB evaluations, with a suite-specific predicate given in Appendix B.2; in Fixed and TaskVariant, J as in Table 6.

continued on the next page

Table 7 (continued)

Term	Meaning in this paper
Native success	Success as judged by an external benchmark's own evaluator (Appendix H.1).
Grounding	Mean required-label coverage $\|T_q \cap R_q\|/\|R_q\|$ over annotated context requests with non-empty R_q, two annotated requests per Fixed/TaskVariant task (the first and the decision checkpoint), independent of the action chosen (Appendix E).
Recall	Replay's mean relevant-label coverage $\|T_q \cap L_q\|/\|L_q\|$ (Appendix E).
Contamination	Replay's branch mismatch among selected candidates; lower is better (Appendix E).
Recovery (Replay metric)	The fraction of annotated failure events followed, within a window, by a returned productive or gold-decision label; an event rate (Appendix E).
Footprint	Retrieval-eligible stored units per ingested write, a storage ratio (Appendix E).
Temporal lag Δ	$\Delta = t_u - t_w$, the checkpoint-index difference between writing the decisive record and using it, assigned by the generator (Appendix F).
Structural lag G	The per-variant count of task-relevant branch and relation transitions between the decisive record and its use, assigned by the LongHorizon generator and not computed from the provenance graph (Appendix F).
Delivery, use, and execution failures	Respectively, needed evidence absent from the returned context, delivered evidence the planner does not act on, and a justified action that fails to run (Appendix B.2).
Memory methods compared (implementations in Appendix C.2)	
Text baselines	The lexical baselines of Section 4, lexical-preview summary and lexical fragment retrieval, which store untyped text records and use no summarization model, learned embedding, or neural reranker.
No memory	Returns no context; the lower bound.
Lexical-preview summary	Groups the lexically top-ranked records three per fragment as short previews.
Rolling-preview summary	Groups lexically selected records, in write order, four per fragment as shorter previews.
Lossless summary	Lexical-preview summary over full serialized records; only the storage is lossless.
Repair summary	The 102-task ScienceAgentBench comparator: lossless summary in a single-branch repair wrapper that leads with the latest program and its result (Appendix H.1).
Lexical fragment retrieval	The lexical ranker returning one record per fragment; it uses no vectors.
Hashed-vector retrieval	Ranks records across branches by the cosine similarity of hashed word-count vectors, plus branch, kind, and lexical-overlap bonuses.
Raw-event retrieval	Returns single records selected as in lexical fragment retrieval.
Truncated history	The most recent records of the request branch, or of all branches if it holds none.
RSM	Research State Memory (Section 5); its ablations remove one assembly switch at a time from the complete RSM adapter (Appendix E.3).
A-MEM	The official A-MEM implementation (Xu et al., 2025), which builds, links, and evolves notes with a memory-side language model (Appendix G).
StructMem	Adapted official code of StructMem (Xu et al., 2026), a hierarchical event memory with its own memory-side models (Appendix G).

B ESMB BENCHMARK SPECIFICATION

This appendix specifies the benchmark introduced in Section 3: its suites and task families (Appendix B.1), the information contracts, scoring and run settings of every study in the paper (Appendix B.2), the planner interface (Appendix B.3), and the evidence on task validity (Appendix B.4).

B.1 SUITES AND TASK FAMILIES

Task template. Every end-to-end ESMB task instantiates one template: an initial research state of typed writes with relations and stored artifacts; annotated context requests at decision checkpoints; a closed action set whose actions run stored scripts or read stored files and whose effects (observation text, new writes and outcome events) are authored in advance; and a label-based scoring outcome. Scoring checks whether decisive research state reaches the planner at the checkpoint. Replay uses the same state transitions over fixed trajectories of writes, requests, actions and outcomes, with no planner and no task-level outcome. Table 8 lists the four suites in the order of Table 1.

Fixed and TaskVariant share the five task families of Table 9, and the public-request comparisons run the same families under the public contract (Table 11; Appendix G).

Table 8: ESMB suites. Fixed is the reference contract; each other suite changes one primary dimension. Joint-success predicates and run settings are in Appendix B.2.

Suite	Change from Fixed	Benchmark unit	Size / denominator	Methods reported	Primary quantities
Fixed	None: reference contract with a live planner and label-support cues	Task execution: two annotated requests, one two-action decision	5 tasks $\times$ 3 executions; $n = 15$ per method and planner	`gpt-5.2` (primary): no memory, lexical-preview summary, lexical fragment retrieval, RSM. `gpt-4.1` (robustness): lexical-preview summary, RSM	Joint success (Wilson intervals in Appendix D); grounding
TaskVariant	Task surface restated; same contract, families, and R and U_a structure	As Fixed	5 tasks $\times$ 3 executions; $n = 15$ per method and planner	Lexical-preview summary, RSM (both planners)	As Fixed
Replay	No planner: fixed writes, requests, actions and outcomes, in six separately authored traces	Deterministic trace	6 traces; each metric averaged over the traces that declare it	No memory, lexical-preview summary, lexical fragment retrieval, RSM; RSM ablations	Recall, grounding, recovery, contamination, footprint
LongHorizon	Write–use lag (Δ, G); six-fragment budget; opaque, citation-based planner interface	Generated multi-checkpoint task execution	11 tasks $\times$ 5 executions; $n = 55$ per method and planner	Hashed-vector retrieval, RSM (both planners)	Joint success (citation-based predicate)

Fixed task inventory. Table 10 lists the decision checkpoint of each Fixed task as written in the task fixtures. Every task has two annotated context requests: a first checkpoint with one permitted action, whose authored effect writes a new record (the decisive result in the cross-branch families), and the decision checkpoint, which offers two actions. Both requests have a non-empty R and enter grounding. In the conflicting-evidence control, R omits the contradiction records but U_a includes them, so joint success checks their delivery only through the action clause. In the artifact-grounded update control, the audit records are relevant labels at the decision checkpoint but lie outside R and U_a, so joint success there checks only the delivery of the earlier result and its hypothesis.

TaskVariant. Each TaskVariant task maps onto one row of Table 10. It keeps the family, the branch placement of the decisive and required records, the checkpoint phases and expected kinds, the dependencies that R and U_a encode, and the sizes of both sets, while restating object names, the scientific situation, and action identifiers and titles.

Replay. Replay consists of six fixed traces authored separately from the Fixed tasks. The Replay branch-switching, conflicting-evidence, stale-supersession and artifact-update traces share a family name with Fixed but have their own records, requests and branch layouts. Only the Replay branch-switching and delayed-context traces span two branches, so the Replay stale-supersession trace, unlike its Fixed namesake, is single-branch. Two traces have no Fixed counterpart: repeated-error recovery, which is single-branch, and delayed-context usefulness. Appendix E defines each metric and the traces that declare it.

The LongHorizon generator, its horizon variables and its task mapping are specified in Appendix F.

Table 9: Task families of Fixed and TaskVariant, cross-branch families first. In both suites, the decision checkpoint of each cross-branch family requires a source that the same-branch filter of lexical-preview summary and lexical fragment retrieval excludes, so a method applying that filter cannot satisfy R there; the local controls require only current-branch sources.

Task family	Evidence	Memory-sensitive requirement	Gold action / distractor
Failed-experiment recovery	Cross-branch	Carry the failed result from the original branch into the recovery decision on the alternate branch.	Run the recovery experiment / re-read the alternate-branch note.
Branch switching	Cross-branch	Carry the original branch's null result into the decision to pursue the alternate branch, together with that branch's hypothesis and evidence.	Validate the alternate branch / re-read its note.
Stale supersession	Cross-branch	Connect the superseding hypothesis and its evidence with the null result that contradicts the superseded hypothesis on the other branch.	Validate the superseding hypothesis / re-read its note.
Conflicting evidence	Local control	Keep a stored contradiction and its report visible next to newly observed support on the same branch.	Run the controlled rerun / conclude from the support alone.
Artifact-grounded update	Local control	Bring an audit artifact and its derived evidence to the decision about an earlier gain on the same branch.	Re-evaluate the earlier result / retain it.

Table 10: Fixed decision checkpoints, read from the task fixtures. Decisive records are the result written by the first checkpoint's action (cross-branch families) or the stored records that the task description names (controls).

Task family	Decision checkpoint (branch; phase)	Decisive record(s) (branch)	Required labels R	Support U_a of gold action	Gold action / distractor
Cross-branch families					
Failed-experiment recovery	`ctx_recovery` (`alt`; branch reassessment)	`r_lr_null` (`main`)	`r_lr_null`, `h_batch`, `ev_batch`	$= R$	`run_batch_fix` / `inspect_batch_signal_only`
Branch switching	`ctx_branch_switch` (`alt`; branch reassessment)	`r_main_null` (`main`)	`r_main_null`, `h_alt`, `ev_alt`	$= R$	`validate_alt_branch` / `inspect_alt_note_only`
Stale supersession	`ctx_supersede` (`alt`; branch reassessment)	`r_old_null` (`main`)	`r_old_null`, `h_new`, `ev_curriculum`	$= R$	`run_curriculum_validation` / `inspect_curriculum_note_only`
Local controls					
Conflicting evidence	`ctx_conflict` (`main`; experiment planning)	`ev_conflict`, `art_conflict` (`main`)	`ev_support`, `h_salt`	`ev_support`, `ev_conflict`, `art_conflict`	`run_controlled_rerun` / `conclude_from_support`
Artifact-grounded update	`ctx_re_eval` (`alt`; decision update)	`ev_audit`, `art_audit` (`alt`)	`r_prev_gain`, `h_data`	$= R$	`run_re_evaluation` / `retain_old_result`

B.2 Contracts, Scoring, and Run Settings

Table 11 lists, for every study in the paper, the cues, the planner and its decoding, the context budget, and the number of executions.

What the annotation-cued request contains. The runner sends every memory method the same request, which carries the branch, query, purpose and fragment budget together with four memory-side annotations: the focus IDs (resolved to stored items), the required labels R, the workflow phase and the expected kind. The R passed to memory is the set the scorer checks. RSM uses all four

Table 11: Contracts and run settings of every study. Executions are nested within tasks, except in ScienceAgentBench-102, where each task has one trajectory and is the statistical unit. *Not recorded* means that no run record or pre-specified study protocol states the value.

Study	**Cues**	**Planner; temperature; output cap**	**Context budget**	**Tasks × executions**
Annotation-cued contract (ESMB)				
Fixed, TaskVariant	Memory: focus IDs, R, phase, expected kind. Planner: label-support cues	`gpt-5.2` (primary), `gpt-4.1` (robustness); 0.0; none[c]	3 fragments; no byte cap[c]	5×3 per suite and planner
Replay	Memory: focus IDs, R, phase, expected kind	No planner (fixed actions)	2–4 fragments per request	6 traces × one deterministic pass
LongHorizon	Memory: focus IDs, phase, expected kind; R only at the terminal checkpoints of the branch-switching and stale-supersession families. Planner: opaque fragments, no label-support cues	`gpt-5.2` (primary), `gpt-4.1` (robustness); 0.0[c]; none[c]	6 fragments; no byte cap[c]	11×5 per planner
Public contract (ESMB tasks)				
A-MEM comparison	None	`gpt-5.2`; 0.0; 256 tokens	3 fragments / 4,096 UTF-8 bytes	5×3 per suite
StructMem comparison	None	`gpt-5.2`; 0.0; 256 tokens	3 fragments / 4,096 UTF-8 bytes	5×3 per suite
Joint cue removal	None; no label-support cues	`gpt-5.2`; 0.0; 256 tokens[p]	3 fragments / 4,096 UTF-8 bytes	5×3 per suite
Outside ESMB				
ScienceAgent-Bench-102	None; single-branch repair wrapper	Model requested as `gpt-5.2` through a third-party relay (Responses API); 0.0; 2,000 tokens[p]. The served model's identity is not independently established	6,000-token memory cap; 2,000-token feedback cap	102×1, up to 3 repairs each
External pilots	Protocol-specific wrappers (Appendix H.2)	One-shot ScienceAgentBench pilot: `gpt-5.2` per its task manifest, with transport and temperature not recorded. Other pilots: not recorded	Protocol-specific	4×1 per protocol
Induction study (downstream)	Adapter input excludes evaluator labels and gold objects, relations and actions	Held-out split identifier, adapter model, prompt and schema version, decoding, retry and cache settings, downstream planner and budgets: not recorded		20 episodes per write condition

[c] Read from the harness code (runner and LongHorizon run script) and not checked against a run record. [p] Set by the study's pre-specified protocol.

annotations (Appendix C.1); the text baselines use at most the expected kind, as a ranking signal. Relevant labels accompany the request for scoring only, and no memory method reads them. At the three cross-branch Fixed decision checkpoints, the focus IDs include the decisive earlier result together with the original-branch and current-branch hypotheses (`r_lr_null`, `h_lr` and `h_batch` at `ctx_recovery`). Public requests clear these annotations (Appendix G); LongHorizon keeps focus IDs, phase and expected kind, and R at two families' terminal checkpoints, but renders opaque fragments (Appendix F).

Joint success. For a Fixed or TaskVariant task, let T be the source labels returned at the decision checkpoint, R that checkpoint's required labels, U_a the authored support labels of the intended action a, and A the indicator that a is executed there. Then

$$J = \mathbf{1}[R \subseteq T]\ \mathbf{1}[U_a \subseteq T]\ \mathbf{1}[A].$$

U_a is supplied by the action's authored effect, not inferred from the planner's citations. Most tasks have $U_a = R$; the conflicting-evidence control does not ($R = \{\texttt{ev_support}, \texttt{h_salt}\}$, $U_a = \{\texttt{ev_support}, \texttt{ev_conflict}, \texttt{art_conflict}\}$), so the two label clauses cannot in general be merged. The labels listed per action in the planner prompt (Appendix B.3) form a third, separately authored set that joint success does not use. Label checks establish only that a source is present in the returned context; they do not verify that its rendered text is complete. In LongHorizon, joint success requires the gold action together with a planner-cited fragment carrying the decisive label. In the public comparisons the same three clauses apply, with support projected through the fragment lines retained under the byte cap. The benchmark logs the executed action separately, but this paper reports no action-only rate; joint success is the outcome quantity for these suites. Grounding (Appendix A) is computed at both context requests of each Fixed and TaskVariant task, each against its own required labels.

B.3 Planner Interface

We derive the prompt description below from the released harness code, not from logs of the recorded LLM runs; it is consistent with the reported grounding values. The planner never reads the store. At each checkpoint it receives the returned context together with benchmark-fixed decision information. Under the annotation-cued contract of Fixed and TaskVariant, only the returned context and the cues computed from it differ across memory methods, and the prompt includes:

- *Checkpoint description:* the task identifier and description, and the checkpoint identifier, purpose, query and branch.
- *Workflow state:* the identifiers of all records written so far in the task, the completed actions, and the recent observation, i.e., the authored observation text of the previous action.
- *Returned context:* the returned source labels, and each fragment's text with its source labels, candidate identifiers and kinds.
- *Closed action menu:* each allowed action's identifier, title, branch and description, with the context labels the fixture lists as that action's requirements.
- *Label-support cues:* for each action, which of its listed labels are present in or missing from the returned context, and whether it uses a returned cross-branch result; a decision protocol that tells the planner to treat the listed labels as support requirements and to prefer the action with stronger label-grounded support; and, at branch reassessment, a list of the returned cross-branch results and, if some action lists one, an instruction to prefer the actions that do. At other phases, or when no cross-branch result is returned, the prompt states that no extra phase-specific rule applies.

The prompt does not contain the stored records, the focus IDs, the relevant labels, the checkpoint's R as such, U_a, or the gold action. At the three cross-branch decision checkpoints, the gold action lists the decisive result with the original-branch and current-branch hypotheses (for recovery, `r_lr_null`, `h_lr` and `h_batch`, whereas $R = U_a = \{\texttt{r_lr_null}, \texttt{h_batch}, \texttt{ev_batch}\}$), and the distractor lists only the two current-branch records in R. At the two local controls, both actions list exactly R. Decision-checkpoint actions carry generic titles and descriptions that differ only in naming an execution or a review path (in Fixed, "Advance with execution" versus "Advance with review"); the gold action is the execution one in all ten Fixed and TaskVariant tasks, and the action identifiers shown in the menu (e.g., `run_batch_fix` versus `inspect_batch_signal_only`) remain descriptive. If the planner returns an action identifier outside the allowed set, the runner

deterministically substitutes the allowed action whose listed labels overlap most with the returned labels; how often this repair fired is not reported.

Because the recent observation and the written-record identifiers reach every method's planner, every planner at the Fixed recovery checkpoint sees the flat-sweep observation and the label `r_lr_null`, whether or not memory returns that result. Fixed therefore tests prescribed source delivery combined with label-support cues; it does not remove the decisive evidence from the planner's input entirely.

Public and LongHorizon interfaces. Under public requests (Appendix G), the planner receives the request purpose, query and branch, the completed actions and latest observation, the fragments by opaque identifier and text, and the actions by opaque identifier, title and description, with no per-action label requirements. It returns an action and the fragments it cites, under a 4,096-byte context projection and a 256-token output cap; parser and execution failures are retained as failures. In LongHorizon, the prompt contains the task and checkpoint identifiers, description, purpose and query, together with the completed actions and the recent observation. Fragments are rendered opaquely as `C1`–`C6`, with text but no labels, and actions carry only identifiers, titles and descriptions. The planner cites the fragments that support its choice, and joint success is scored on those citations (Appendix F). The action-identifier repair also applies in LongHorizon, where it uses per-action labels hidden from the planner.

B.4 Task Validity: Retrospective Audit

Three raters retrospectively audited all 16 episodes of the Replay, Fixed and TaskVariant suites: six Replay, five Fixed and five TaskVariant. The inventory contains 23 scoring-relevant checkpoints (13 Replay, five Fixed, five TaskVariant), ten action checkpoints (the Fixed and TaskVariant decision checkpoints) and five matched Fixed–TaskVariant pairs. Raters were blinded to method identity, performance, gold actions, labels, types, and relations. The raters, recruited separately, are not among the authors. Judgments assess scientific coherence and solvability, acceptable actions, decisive sources, wording leakage, and variant equivalence. Table 12 reports the figures before adjudication, and no rater was excluded for low agreement. Acceptable-action-set agreement is measured among raters, whereas preferred-action agreement is measured against the benchmark gold. Leakage uses the labels None, Weak, Strong, and Unclear. Raters reviewed the finished tasks and authored none of them.

In the audit record, no sensitivity analysis that removes disputed items reverses the direction of the RSM–lexical-preview-summary comparison; we do not tabulate the per-analysis values, subset definitions, rater forms, eligibility criteria or category anchors. Rater packets omit the memory-side annotations (focus IDs, R, phase and expected kind), so the audit supports face and content validity within this inventory but cannot rule out every leakage route; LongHorizon was not audited.

Table 12: Retrospective audit of the Replay, Fixed and TaskVariant suites, before adjudication. Denominators differ by judgment and are not pooled.

Judgment or agreement metric	**Result**
Majority-valid episodes	$16/16$
Majority-no-wording-leakage checkpoints	$22/23$
Majority-Weak wording-leakage checkpoints	$1/23$
Majority-Strong / Majority-Unclear wording leakage	$0/23$ / $0/23$
Acceptable-action Krippendorff's α	0.87
Exact acceptable-action-set agreement among raters	$9/10$
Preferred-action agreement with benchmark gold	$10/10$
Mean pairwise decisive-source Jaccard	0.84
Majority-equivalent Fixed–TaskVariant pairs	$5/5$

C MEMORY METHODS

This appendix specifies RSM's stored state and context assembly (Appendix C.1), the comparators (Appendix C.2), and a separate study of inducing typed writes from workflow signals (Appendix C.3).

C.1 RSM STATE AND CONTEXT ASSEMBLY

This subsection and Appendix C.2 describe the released harness code; they do not reproduce logs from the recorded runs.

State. An item has an identifier, one of seven kinds (open question, hypothesis, experiment, evidence, result, decision, artifact), a branch, a title, an optional summary, a schema-validated payload specific to its kind, an importance score, a status, and creation and update times linked to the provenance events that created and last modified it. The status is active (the default), superseded, or archived; retrieval uses it to exclude archived items and to list open questions that are still active and unanswered. Every write is first appended to the provenance log. Experiments and artifacts that carry an external key are then updated in place; items of the other kinds are appended as new items. Relations are typed, directed edges of eight types: tests, produces, supports, contradicts, answers, has-artifact, supersedes, and derived-from. Each type constrains the kinds of its endpoints; for example, a contradicts edge runs from an evidence or result item to a hypothesis or decision.

Requests. A request carries a branch, query text, purpose, and fragment budget B. Under the annotation-cued contract it also carries focus IDs, an expected kind, a workflow phase, and the required labels R_q. Public requests keep the branch and clear the other four fields (Table 11).

Context assembly. The implemented sequence is given below. Bracketed names are the configuration switches that Appendix E.3 disables; unmarked steps have no switch.

1. *Base candidates.* From the request branch: up to two of the earliest active open questions, after dropping answered ones; two hypotheses; and the two most recent experiments and results. From any branch: the focus items and their one-hop relation neighbors, and the two most recent provenance events. Without focus IDs the neighborhood is empty.
2. *Ranking.* Candidates are ordered lexicographically by branch match, focus adjacency (a focus item or its one-hop neighbor), relation signal (adjacent to a focus item through a bundle-eligible relation, below), expected-kind match, lexical overlap with the query, importance, and recency. Section priority and a content key break the remaining ties.
3. *Representative reservation* [two-stage representative selection]. The highest-ranked unresolved question, hypothesis, and result from the branch-scoped lists of step 1 are reserved, in that order, until the budget is reached.
4. *Fill* [item-first selection]. Remaining slots are filled with typed items on the request branch in rank order. If at least one item is selected, the fill stops and adds no other-branch items or events; otherwise it falls back to typed candidates from any branch, which exist only in the focus neighborhood, and then to events. With the switch off, remaining slots are filled from the full ranked list, events included.
5. *Supplements* [supplemental retrieval]. Three routes then add records. (a) *Branch handoff*: at a branch-reassessment request whose selection holds no other-branch item, the most recently updated result on any other branch is attached as a neighbor of a selected hypothesis or, failing that, of another selected item. This route reads the phase, not the focus IDs. (b) *Artifact lineage*: when the expected kind is result and the selected fragments show a result or evidence item but no artifact, a same-branch artifact within two derived-from or has-artifact hops is added as a primary item; if the budget is full, it replaces the lowest-ranked item not reserved in step 3, preferring one outside the focus neighborhood. (c) *Planning cue*: at branch reassessment, if an other-branch item has been selected and no open question is visible, an unresolved question from that item's branch is attached to it as a neighbor.
6. *Required-label backfill* [required-label backfill; runs only when supplemental retrieval is enabled]. While a required label is missing from the selection, the unselected candidate whose fragment would carry the most missing labels is promoted to a primary item, appended or replacing an item not reserved in step 3, until coverage is reached or no candidate qualifies.

Supplements are appended to the logged candidate list, but they do not pass through a new common ranking and packing step.

Fragments and neighbors. Each selected primary item is rendered with up to two neighbors: supplements attached in step 5 first, then direct relation neighbors. Direct neighbors are ordered by relation type (has-artifact, derived-from, supports or contradicts, produces, supersedes), then by same-branch membership, focus membership, importance, and recency. Each of these types admits only certain neighbor kinds; for example, a contradicts or supports neighbor must be an evidence or result item. Tests and answers edges are never bundled, and relation neighbors are not branch-filtered. A neighbor shares its primary item's fragment slot. Fragment text has one line per item, `primary: [kind] title :: payload` followed by `neighbor[tag]: [kind] title :: payload`. The payload is the item's summary, or else its payload fields, cut to 140 characters. The tag names the relation types that link the neighbor to the primary item (for example `neighbor[contradicts]`) or the route that attached it (`supplemental_context` or `planning_cue`). Each fragment also carries the source labels of its items, and the public projection cuts rendered text at its byte cap.

Branch scope. RSM's default selection (the branch-scoped lists of step 1 and the selections of steps 3 and 4) is restricted to the request branch. Other-branch records reach the returned context by five routes. (i) Focus-neighborhood candidates (step 1) become primary items only through backfill or when the request branch holds no typed candidate. (ii) The branch-handoff and planning-cue attachments of step 5 are gated by phase. (iii) Backfill (step 6) is keyed on the required labels. (iv) Relation neighbors are attached at rendering without a branch filter. (v) The two most recent provenance events, taken from any branch, are returned only when no typed item is selected, and they carry no source labels. Routes (i)–(iii) use the request annotations (Section 5).

Running example. At `ctx_recovery`, the request is on `alt`, in the branch-reassessment phase, with expected kind result and focus IDs `r_lr_null`, `h_lr`, and `h_batch`. Through the branch lists and the focus neighborhood, the base candidates contain `h_batch` and `ev_batch` on `alt` and `h_lr` and `r_lr_null` on `main`. Steps 3 and 4 select the two `alt` items and keep that selection, so `r_lr_null` is a candidate but not a primary item. It enters the context through the branch-handoff attachment of step 5, which bundles it with `h_batch` beside `ev_batch`.

Backfill and annotations. Backfill reads the request's required labels, which are the labels the joint-success predicate checks (Appendix B.2), so backfill itself depends on the annotations. In the harness implementation, disabling backfill alone changes no returned label set at any of the 20 memory requests issued by the Fixed and TaskVariant tasks (two per task, fixed in advance because each first checkpoint has a single permitted action) or at the 17 Replay requests.

C.2 Comparators

Within each comparison, methods share the stored history, requests, planner, actions, and fragment budget, and differ in stored representation, selection rule, and records per fragment. Under annotation-cued requests, RSM reads the focus IDs, phase, expected kind, and required labels, whereas the text baselines read at most the expected kind, as a ranking tie-break or bonus. Table 13 summarizes each comparator.

Text records and the lexical ranker. Every comparator other than no memory, A-MEM, and StructMem stores one text record per write. The record serializes the write's kind, title, branch, summary, and payload; the lossless serialization used by lossless and repair summary, and by hashed-vector retrieval in the A-MEM panel, adds every other method-visible field of the write, including its relations. The lexical ranker scores a record by its token overlap with the query, after lower-casing and stop-word removal, and breaks ties by expected-kind match and then by write order. None of these comparators calls a summarization model, a learned embedding model, or a neural reranker.

Hard branch filter. Lexical-preview, lossless, repair, and rolling-preview summary, lexical fragment retrieval, raw-event retrieval, and truncated history all apply the same filter; truncated history applies it before taking the most recent records. Public requests keep the request branch, so the filter

Table 13: Memory methods. B is the request's fragment budget. *Same branch*: the hard filter, under which only records on the request branch are candidates whenever that branch holds a record, and all records otherwise.

Method	Candidates	Selection and rendering	Records per fragment	Reported in
No memory	none	returns an empty context	0	Fixed, Replay, staged pilot
Lexical-preview summary	same branch	lexical ranker; top $3B$ records, three per fragment, each as kind, title, and a 100-character preview	3	Fixed, TaskVariant, Replay, cue removal
Lexical fragment retrieval	same branch	lexical ranker; top B records, each in full	1	Fixed, Replay
Hashed-vector retrieval	all records	hashed-vector score (below); top B records, each in full	1	LongHorizon, A-MEM panel, staged pilot, induction
Lossless summary	same branch	as lexical-preview summary, over lossless serializations rendered in full	3	A-MEM and StructMem panels
Repair summary	single branch	lossless summary behind a first fragment holding the latest program and its execution result	2 in the first, then 3	ScienceAgentBench
Rolling-preview summary	same branch	lexical ranker; top $4B$ records restored to write order, four per fragment, 90-character previews	4	staged pilot
Raw-event retrieval	same branch	as lexical fragment retrieval	1	staged pilot
Truncated history	same branch	the B most recent records, unranked	1	staged pilot
RSM	request branch, focus neighborhood, recent events	Appendix C.1	1 primary $+ \leq 2$ neighbors	all

also applies under the public contract; in the single-branch ScienceAgentBench wrapper it has no effect. Hashed-vector retrieval applies no filter.

Hashed-vector retrieval. The query and each record are tokenized as for the lexical ranker and mapped to 64-dimensional vectors by signed hashing: each token adds ± 1 to one coordinate, and the SHA-256 digest of the token selects both the coordinate and the sign. Vectors are then ℓ_2-normalized. A record's score is its cosine similarity with the query, plus 0.15 if it lies on the request branch (a soft branch bonus), 0.10 if its kind matches the expected kind, and 0.05 times its lexical overlap.

Lossless and repair summary. Lossless summary stores every write losslessly but passes the planner only the records it selects. Repair summary is the lossless-summary configuration inside the single-branch ScienceAgentBench wrapper. Its first fragment always carries the latest generated program and its execution result, truncated at head and tail to fit, followed by the adapter's own fragments. Requests allow 64 fragments within a 6,000-token memory cap, and a 2,000-token feedback cap applies.

Records per fragment. A budget of B fragments admits $3B$ records in lexical-preview and lossless summary, $4B$ in rolling-preview summary, B primary items plus up to $2B$ neighbors in RSM, and B records in lexical fragment, hashed-vector, and raw-event retrieval and in truncated history. Under the three-fragment budget these are 9, 12, 3 primary items plus up to 6 neighbors, and 3. Fragment

budgets and the public byte cap (Table 11) therefore do not equalize the records or text volume that methods deliver, or the memory-side compute they use.

Identifiers. Run records use legacy identifiers: `no_memory` (no memory); `summary_memory` (lexical-preview summary and, in other configurations, lossless summary and the repair summary of the ScienceAgentBench manifest); `vector_memory` (lexical fragment retrieval, which uses no vectors); `dense_retrieval_plus_reranker` (hashed-vector retrieval, which has no learned embedding model or neural reranker); `rolling_summary` (rolling-preview summary); `raw_event_log_retrieval` (raw-event retrieval); `full_history_truncation` (truncated history); and `structured_memory` (RSM).

C.3 Inducing Typed Writes from Explicit Workflow Signals

ESMB supplies typed writes through fixtures and authored action effects (Section 8). A separate study tests whether such writes can be induced from explicit workflow signals. A frozen adapter reads 214 chronological planner, tool, metric, and artifact signals from 20 held-out episodes and constructs typed items and relations. Its input excludes evaluator labels, future information, gold objects and relations, and gold actions. Three annotators supply reference annotations over the four core types (hypothesis, experiment, evidence, and result), so the study does not establish extraction quality for the other three kinds. Downstream, 20 executions per condition compare RSM with reference writes, RSM with induced writes, and hashed-vector retrieval (Appendix C.2) over the same raw signals stored losslessly. The adapter's run settings are in Table 11.

Table 14: Inducing typed writes from explicit workflow signals. Extraction metrics are computed over signal, object, and relation instances; each downstream condition has 20 episodes.

Extraction metric	**Value**
Object detection F1	0.93
Type macro-F1	0.92
Relation F1	0.88
Exact provenance-source accuracy	0.98
Unsupported-write rate	2.0%
Downstream condition	**Successful episodes**
RSM with reference writes	18/20
RSM with induced writes	17/20
Hashed-vector retrieval over the raw signals	12/20

Induced writes yield one success fewer than reference writes (-5.0 points) and five more than hashed-vector retrieval ($+25.0$ points; Table 14). The study covers explicit, instrumented signals only; induction from unstructured interaction is untested. Extraction errors are stored and reused by the memory layer like any other write.

D ESMB-Fixed and ESMB-TaskVariant Results

This appendix completes the annotation-cued comparison of Section 6.1. Main Table 4 reports joint-success counts; Table 15 adds, for each planner and suite, Wilson intervals, grounding, the discordant task-trials behind the paired comparison, and the family-level W/T/L. Tasks, required labels, and the joint-success predicate are specified in Appendices B.1 and B.2; paired trial-0 examples appear in Appendices I.1 and I.2.

Executions and per-design outcomes. Each suite contains five task designs, one per family (Table 10), and each method executes each design three times with the planner at temperature 0.0. The trial index is part of the planner's response-cache key, so the three executions are distinct model calls, whose outcomes can differ even at temperature 0.0. Counted per design, RSM has three wins, two ties, and no losses against lexical-preview summary in each model/suite cell.

Table 15: Annotation-cued Fixed and TaskVariant: RSM against lexical-preview summary for both planners. Joint successes are counts over five task designs × three executions, with Wilson 95% intervals (Wilson, 1927). Grounding and discordant task-trials list RSM first and lexical-preview summary second; a discordant task-trial is one that only that method solves. Exact two-sided McNemar p values (McNemar, 1947) are computed from these discordant counts, and family W/T/L compares the two methods per task design. Executions are nested within five task designs, so the intervals and p values are descriptive.

	Joint successes					
Planner	**RSM**	**Lexical-preview summary**	**Grounding**	**Discordant task-trials**	**McNemar p**	**Family W/T/L**
Fixed						
`gpt-5.2` (primary)	12/15 [0.548, 0.930]	6/15 [0.198, 0.643]	1.000 / 0.900	6 / 0	0.031	3/2/0
`gpt-4.1` (robustness)	14/15 [0.702, 0.988]	6/15 [0.198, 0.643]	1.000 / 0.900	8 / 0	0.008	3/2/0
TaskVariant						
`gpt-5.2`	15/15 [0.796, 1.000]	6/15 [0.198, 0.643]	1.000 / 0.900	9 / 0	0.004	3/2/0
`gpt-4.1`	15/15 [0.796, 1.000]	6/15 [0.198, 0.643]	1.000 / 0.900	9 / 0	0.004	3/2/0

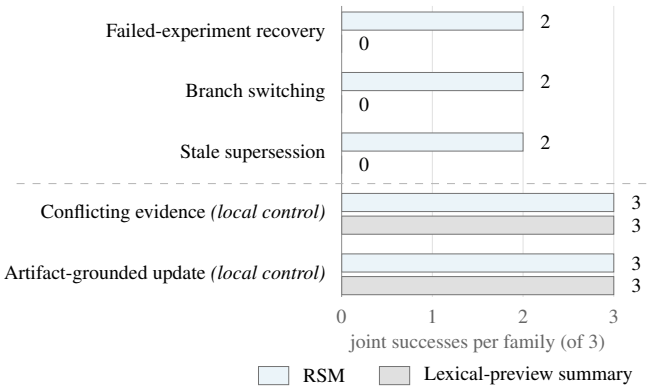


Figure 4: Joint successes per task family on annotation-cued Fixed with `gpt-5.2`, three executions per family; rows sum to 12/15 for RSM and 6/15 for lexical-preview summary. The lexical-preview-summary bars repeat Table 3; RSM's counts are those of Section 6.1. The three cross-branch families move from 0/3 to 2/3, and the two local controls, below the dashed rule, tie at 3/3.

Per-family outcomes. Figure 4 gives the per-family counts for the primary `gpt-5.2` Fixed run. With `gpt-4.1` on Fixed, RSM's three family-level wins are the cross-branch families and its two ties are the local controls at 3/3, so it solves 8/9 cross-branch task-trials combined; the per-family split of those eight is not reported. On TaskVariant, with either planner, RSM reaches 3/3 and lexical-preview summary 0/3 on each cross-branch family, and both tie at 3/3 on each local control.

Source of the paired differences. In the released harness, lexical-preview summary's branch filter excludes, at each cross-branch decision checkpoint, the one required result that lies on the other branch, so the coverage clause of joint success fails there whatever action the planner selects. Its 6/15 is therefore a ceiling, reached in every cell with 0/3 on each cross-branch family and 3/3 on each local control, and its grounding of 0.900 reflects the single missing label at each of the three cross-branch decision checkpoints. RSM also solves every local-control task-trial in every cell, so all discordant task-trials favor RSM and their number equals RSM's cross-branch successes. Lexical-preview summary is the comparator because it has the highest joint success of the three baselines reported for `gpt-5.2` on Fixed (Table 4(a)); the other three cells report RSM and lexical-preview

summary only. In every cell the annotation-cued request supplies focus IDs, workflow phase, and required labels, which RSM uses and the text baselines do not.

Post-failure recovery diagnostic. Applied to Fixed, the event-level recovery metric of Replay (Appendix E) is $0/9$ for every method, RSM included. Each of the nine applicable failure events per method is annotated at a cross-branch decision checkpoint whose action ends the task, so no later context request falls inside its recovery window. The metric therefore cannot register a recovery in Fixed and is not an outcome measure.

E ESMB-REPLAY RESULTS

ESMB-Replay isolates retrieval and context assembly. Its six traces fix the writes, the 17 context requests (each with a budget of two to four fragments) with their focus IDs and relevant and required labels, the planner actions, and the outcome annotations (Appendix B.1). Every method receives the same sequence and differs only in the context it returns. The reported values are deterministic descriptive metrics with no sampling variation. A rerun of the harness on the Replay fixtures reproduces every entry of Tables 16 and 17.

E.1 METRICS AND AGGREGATES

For a request q, let L_q be its relevant labels, R_q its required labels, and T_q the source labels of all returned fragments, attached neighbors included. Each metric is first computed per trace, and the reported value is its mean over the traces that declare the metric, so each trace has equal weight whatever its number of requests or events. Recall and footprint are declared by all six traces, grounding by four, recovery by three, and contamination by the two traces that span two branches.

- *Recall*: the mean of $|T_q \cap L_q|/|L_q|$ over the trace's requests with $L_q \neq \emptyset$.
- *Grounding*: the mean of $|T_q \cap R_q|/|R_q|$ over requests with $R_q \neq \emptyset$, as defined in Appendix A. Because L_q and R_q need not coincide, perfect grounding need not imply perfect recall.
- *Recovery* (the Replay metric): the fraction of the trace's annotated failure events that are followed, within the trace's recovery window (8, 4, and 6 trace steps for repeated-error recovery, branch switching, and delayed-context usefulness), by a context request whose returned labels include a productive label for that request's branch or a gold-decision label. It is an event rate. Because Replay's actions are fixed and identical across methods, it scores only the labels memory returns.
- *Contamination*: over requests with at least one pre-budget candidate on the request branch, the fraction of selected candidates that lie on another branch, pooled within the trace. Selected candidates are the records a method places in fragments as primary content: every returned record for the text baselines, and fragment primaries for RSM. Neighbors attached inside a fragment are not counted. A selected candidate on another branch counts toward contamination even when it is relevant.
- *Footprint*: retrieval-eligible stored units per ingested write, a storage ratio that does not measure context size.

Recovery, contamination, and footprint. Recovery can register only in the repeated-error trace. In the branch-switching trace, no context request falls inside the window that follows the failure event. In the delayed-context trace, the only productive label that memory can return, the final result `r2`, is written after the last context request that falls inside either window; the other productive

Table 16: Deterministic ESMB-Replay metrics, averaged over the traces that declare each metric. Recall and grounding concern returned labels, recovery is an event rate, contamination is branch mismatch among selected candidates (lower is better), and footprint is a storage ratio.

Method	Recall	Grounding	Recovery	Contam.	Footprint
No memory	0.000	0.000	0.000	–	0.000
Lexical-preview summary	0.956	1.000	0.333	0.000	1.000
Lexical fragment retrieval	0.744	0.792	0.000	0.000	1.000
RSM	1.000	1.000	0.333	0.000	1.000

and gold-decision labels name actions and outcomes, which are never stored. A recovery of 0.333 therefore means that a method recovers both failure events of the repeated-error trace and none elsewhere, and 0.333 is the maximum attainable on these fixtures. Lexical-preview summary and RSM reach it. Lexical fragment retrieval does not, because it displaces `r2` at the repeated-error trace's last request (Appendix E.2).

Contamination is zero for the hard-filter text baselines because, whenever the request branch holds a record, all their candidates lie on it. RSM's complete adapter also selects only same-branch primary items in these requests. It does attach two `main` results as neighbors at `alt` requests, `r_main1` at the branch-switching trace's `ctx2` and `r1` at the delayed-context trace's `ctx3`; both are relevant labels at those requests, and as neighbors they are not counted. Footprint is 1.000 for every memory method because each write yields one retrieval-eligible unit, while no memory stores none.

E.2 Exclusion and Displacement in Replay

Replay logs each method's pre-budget candidate list alongside its returned fragments, so a relevant label missing from a returned context can be assigned to one of the two failure modes of Section 2: an *exclusion* when no pre-budget candidate carries it ($r \notin K_t$), and a *displacement* when a candidate carries it but the record is not returned ($r \in K_t \setminus C_t$). The branch-switching request of Appendix I.4 is an exclusion. The two cases below are displacements of lexical fragment retrieval, in which ranking and packing drop a record that was among the candidates (Section 4).

Two displacements. At the stale-supersession trace's `ctx1` (branch `main`, three-fragment budget, relevant labels `h2` and `r1`), the query "focus on the superseding hypothesis" shares words with the old hypothesis `h1`, the superseding hypothesis `h2`, and the experiment `exp1`, but not with the refuting result `r1`, which ranks last of five candidates. Lexical fragment retrieval returns `h1`, `h2`, and `exp1`, so the old hypothesis reaches the context without the result that refutes it. At the same request, RSM returns `h1` as a primary item with `r1` attached through its contradicts edge and `h2` through its supersedes edge, the configuration of Figure 1(B). Bundling itself is not ablated (Appendix E.3). At the artifact-update trace's `ctx2`, lexical fragment retrieval returns the initial result `r1`, the updated result `r2`, and the experiment `exp1`, but not the metrics artifact `art1` or the evidence `ev1` derived from it, which explain the update. Both cases describe stored support missing from the selected context and do not measure task outcomes.

Truncated text. A label counts as returned even when its text is cut. A lexical-preview summary line shows only 100 characters of the serialized record, and an RSM line shows at most 140 characters of the payload. A returned label, and hence an aggregate recall score, does not show that the planner received a complete claim.

E.3 Context-Assembly Ablations

The ablations rerun Replay with one configuration switch of the complete RSM adapter disabled (Appendix C.1, steps 3–6). Some switches cover more than one operation. Supplemental retrieval covers the branch-handoff, artifact-lineage, and planning-cue routes, and required-label backfill runs only when supplemental retrieval is enabled, so the "no supplemental retrieval" variant disables both. Neighbor bundling and individual relation types have no switch and are not ablated, so these results do not measure their effect.

Table 17: RSM context-assembly ablations on ESMB-Replay. Each variant disables one switch of the complete adapter; units follow Table 16.

Variant	Recall	Grounding	Recovery	Contamination
Complete adapter	1.000	1.000	0.333	0.000
No two-stage representative selection	1.000	1.000	0.333	0.000
No item-first selection	1.000	1.000	0.333	0.096
No supplemental retrieval (backfill also off)	0.944	1.000	0.333	0.000

Switches and failure modes. For RSM, K_t in the definitions of exclusion and displacement (Section 2) is the base candidate set of step 1. Supplemental retrieval addresses exclusion, because its routes can add records from outside K_t. The other switches address displacement: required-label backfill promotes a record already in the candidate list whose fragment carries a missing required label, and item-first and two-stage selection decide which candidates occupy the budget.

Results. *No supplemental retrieval* lowers recall from 1.000 to 0.944 and loses three relevant labels, each an exclusion because it is absent from this variant's candidate list. Two are the `main` results attached by the branch-handoff route (`r_main1` at the branch-switching trace's `ctx2`, `r1` at the delayed-context trace's `ctx3`), and one is the artifact `art1` inserted by the artifact-lineage route at the delayed-context trace's `ctx5`. The table cannot separate supplementation from backfill, but backfill alone changes no returned label set (Appendix C.1), so the loss traces to the supplement routes. Without them, RSM's recall is below lexical-preview summary's 0.956. *No item-first selection* changes no returned label set. It raises contamination to 0.096 because unlabeled events fill the slots left after the single `alt` hypothesis at the branch-switching trace's `ctx2` and the delayed-context trace's `ctx3`, and at each of these requests one of them lies on `main` (1 of 8 and 1 of 15 selected candidates in the two traces). *No two-stage representative selection* produces no aggregate change. It alters the returned label set at six of the 17 requests, each time by dropping the reserved open question and, at five of them, adding an experiment record; at those requests both labels lie outside $L_q \cup R_q$. Recovery stays at its attainable maximum of 0.333 in every variant (Appendix E).

F ESMB-LONGHORIZON

LongHorizon lengthens the distance between writing the decisive record and needing it (Table 1). A generator expands each of the three cross-branch families into short, medium, and long variants (Table 18); a conflicting-evidence control and an artifact-grounded-update control, both low-horizon, complete the 11-task suite. Every request has a six-fragment budget. The planner sees opaque fragments without source labels and cites the fragments it relies on; joint success on this suite requires the gold action at the scored checkpoint together with a planner-cited fragment that carries the decisive label. Planner models and decoding settings are listed in Table 11, and Appendix I.5 traces the long recovery variant.

Horizon variables. Figure 5 shows the two variables, both assigned by the generator before any method runs. The temporal lag $\Delta = t_u - t_w$ is counted in checkpoints, where t_w indexes the checkpoint whose action writes the decisive record and t_u the checkpoint whose decision needs it. The long recovery variant writes the record at `cp04` and needs it at `cp10`, so $\Delta = 6$, with five checkpoints in between. G is a per-variant structural-lag label; by the benchmark specification it counts the task-relevant branch and relation transitions between the decisive record and its use, and the harness does not compute it from the provenance graph. The temporal bins are variant labels with no common Δ cutoff; short supersession has $\Delta = 3$, for example, and short recovery $\Delta = 2$. Within each family Δ and G grow together, so the variants do not isolate either lag.

Results. With either planner, RSM obtains 10 more joint successes out of 55 than hashed-vector retrieval, a difference of 18.2 percentage points (Table 19). The harness also defines no memory, rolling-preview summary, raw-event retrieval, and truncated history for this suite; we report outcomes only for RSM and hashed-vector retrieval, and for the headline runs only as suite totals, without a breakdown by temporal bin or by G. Memory-side focus IDs name the decisive record at every

Table 18: LongHorizon positive-task construction: entries are (Δ, G) for each cross-branch family and variant. The two low-horizon control tasks are not part of this mapping.

Family	Short	Medium	Long
Failed-experiment recovery	$(2, 2)$	$(4, 3)$	$(6, 4)$
Branch switching	$(2, 3)$	$(4, 4)$	$(6, 5)$
Stale supersession	$(3, 2)$	$(5, 3)$	$(7, 4)$
Checkpoints per variant	6	8	10

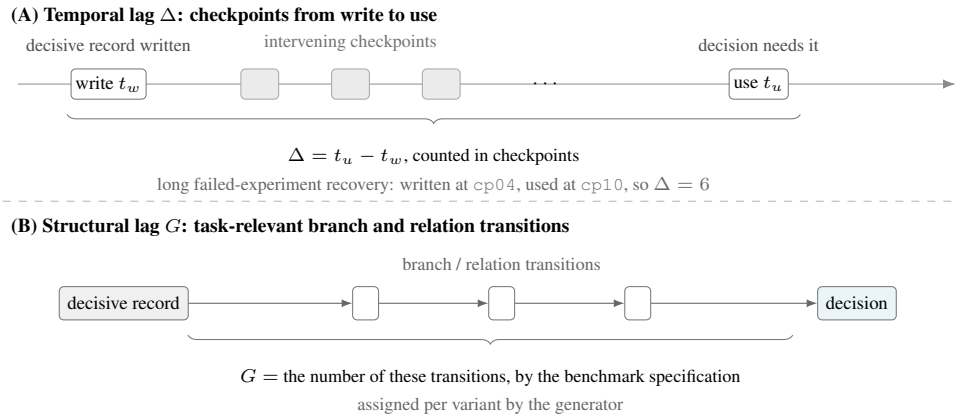


Figure 5: The two horizon variables of ESMB-LongHorizon, both assigned by the generator. (A) Temporal lag $\Delta = t_u - t_w$ in checkpoints; the long recovery variant has $\Delta = 10 - 4 = 6$. (B) Structural lag G, which by the benchmark specification counts the task-relevant branch and relation transitions between the decisive record and its use. Values for every positive task are in Table 18.

Table 19: ESMB-LongHorizon joint successes over 11 tasks (9 positive, 2 controls) with 5 executions per task, so $n = 55$ per method and planner. Outcomes are reported for RSM and hashed-vector retrieval only. Rates carry Wilson 95% intervals (Wilson, 1927); the difference is in percentage points.

		RSM		Hashed-vector retrieval		
Planner	**Role**	**Successes**	**Rate [95% CI]**	**Successes**	**Rate [95% CI]**	**Difference**
`gpt-5.2`	primary	45/55	0.818 [0.697, 0.898]	35/55	0.636 [0.504, 0.751]	+18.2
`gpt-4.1`	robustness	55/55	1.000 [0.935, 1.000]	45/55	0.818 [0.697, 0.898]	+18.2

checkpoint after its write, and hashed-vector retrieval does not use them. The suite thus tests annotation-guided delivery at delayed use; it does not test whether a method can find long-range dependencies without annotations.

G Public-Request Comparisons and Contract Sensitivity

Section 6.2 compares RSM with two published memory systems under public requests. A public request carries the branch, query, fragment budget, and workflow state (completed actions and the latest observation), but no focus IDs, required labels, phase, or expected kind. Public action descriptions omit per-action label requirements, so the planner receives no label-support cues (Appendix B.3). This appendix gives the settings and outcomes of the two comparisons (Appendix G.1) and a joint cue-removal test that applies public requests and the 4,096-byte projection to the annotation-cued comparison between RSM and lexical-preview summary (Appendix G.2).

G.1 A-MEM and StructMem Comparisons

Shared settings. Within each panel, all methods receive the same normalized public event stream, planner, semantic action descriptions, and stopping rule. Planner and projection settings are in Table 11. Each method's native retrieval is projected into at most three fragments and 4,096 UTF-8 bytes before the planner sees it. Lossless summary and hashed-vector retrieval ingest a lossless text serialization of the events (Appendix C.2); in the A-MEM panel, A-MEM ingests the same serialization. Each suite has five task families with three executions per family and method, and parser and execution failures remain in the 15-execution denominators as failures. The projection caps planner-visible context for all methods but does not equalize delivered text or compute; A-MEM and

StructMem also make memory-side model calls, which RSM does not. Joint successes are reported in Table 5(a).

A-MEM panel. The panel runs the official A-MEM implementation (Xu et al., 2025) at revision `ceffb860`, keeping its memory construction, link evolution, and retrieval. Construction and evolution use `gpt-5.2` at temperature 0.7 within a separate 4,096-token output envelope, which is unrelated to the 4,096-byte planner budget. Embeddings use MiniLM (`all-MiniLM-L6-v2`), and native retrieval uses $k = 10$ before the public projection. A-MEM makes 338 memory-side LLM calls in this panel.

RSM ties A-MEM on Fixed ($3/15$ each) and leads on TaskVariant ($3/15$ against $0/15$). Averaging the two suites within each family gives one win, four ties, and no losses for RSM, so the whole difference comes from one family. Lossless summary exceeds RSM in both suites ($6/15$ and $5/15$ against $3/15$ and $3/15$). Hashed-vector retrieval has the most Fixed successes in the panel ($9/15$) and none on TaskVariant. Six TaskVariant executions ended in planner-output parse failures and are counted as failures.

StructMem panel. StructMem (Xu et al., 2026) is adapted from the official code: event extraction, cross-event synthesis, and updating follow the official implementation, with extraction and synthesis prompts written for scientific workflows. Its memory side uses `gpt-4o-mini-2024-07-18` for generation (temperature and top-p 0.1, at most 2,000 output tokens) and `text-embedding-3-small` for 1,536-dimensional embeddings. Native retrieval returns 60 atomic entries and five synthesis summaries before the public projection.

RSM reaches $6/15$ on Fixed and $7/15$ on TaskVariant, against $1/15$ and $2/15$ for StructMem, a gain of 33.3 points in each suite, and $2/15$ and $3/15$ for lossless summary. Removing any one family, in either suite, leaves RSM ahead of both StructMem and lossless summary.

Differences between panels. Both panels use the same `gpt-5.2` planner and the same three-fragment, 4,096-byte projection as the joint cue-removal test, and both run the same tasks with three executions each. Each panel ran under its own frozen protocol. The A-MEM protocol gives every method the same operational action descriptions in place of the test's generic decision wording, and the StructMem protocol adds an adaptive one-to-three fragment ceiling for narrower requests. Apart from the published system and its memory-side models, the panels differ in their method sets (four methods against three), and each panel executed all of its methods, including RSM and lossless summary, in its own run. The same adapters therefore score differently across panels. On Fixed, lossless summary scores $6/15$ in the A-MEM panel and $2/15$ in the StructMem panel, and RSM scores $3/15$ and $6/15$; on TaskVariant the corresponding pairs are $5/15$ and $3/15$, and $3/15$ and $7/15$. We cannot attribute these differences to a cause; because they are comparable in size to the within-panel margins, we interpret each comparison only within its panel.

G.2 Joint Cue Removal

The joint cue-removal test asks how much of the annotation-cued gap between RSM and lexical-preview summary survives when memory-side annotations and planner label-support cues are removed together. Table 20 has two pairs of columns. *Frozen reference* repeats the annotation-cued `gpt-5.2` outcomes of Table 4: $12/15$ against $6/15$ on Fixed and $15/15$ against $6/15$ on TaskVariant. *Strict live* reports new live runs of the same five Fixed and five TaskVariant tasks, with three executions per task and method, under public requests and with label-support cues removed from the planner prompt. Because both cue channels are removed at once, the test measures sensitivity to the information contract, not the effect of memory annotations alone. Both columns use the `gpt-5.2` planner at temperature 0 (Table 11).

Sealed check. The test's frozen protocol also reruns each of 159 checkpoint–adapter pairs (17 Replay requests $\times$ 7 adapter configurations, plus 10 Fixed and 10 TaskVariant checkpoints $\times$ 2 methods) twice, once with the checkpoint's sealed evaluator annotations permuted (multi-element label lists rotated; the gold action, phase and expected kind shifted to other valid values) and once with them replaced by sentinel strings. Each rerun checks that the memory input and output, the rendered prompt, the outbound planner request, and the selected action and citations are unchanged

Table 20: Joint cue removal: joint successes under the annotation-cued reference and the strict live contract. Each suite has five tasks with three executions per method; the combined row sums the two suites only.

	Frozen reference		Strict live	
Suite	**RSM**	**Lexical-preview summary**	**RSM**	**Lexical-preview summary**
Fixed	12/15	6/15	1/15	1/15
TaskVariant	15/15	6/15	2/15	0/15
Combined	27/30	12/30	3/30	1/30

and that no sentinel appears before action commitment. All 318 comparisons pass. A deterministic qualification planner (first action, first fragment) stands in for the live model to produce the selection and citations, and Replay pairs use one synthetic action. The check therefore covers these enumerated runtime channels and does not test for semantic leakage.

The reference gaps are 0.40 on Fixed and 0.60 on TaskVariant, and the strict gaps are 0 and $2/15$. With equal suite weighting,

$$g_{\text{ref}} = \frac{27-12}{30} = 0.50, \qquad g_{\text{strict}} = \frac{3-1}{30} = \frac{2}{30} \approx 0.0667,$$

$$\text{gap retention} = \frac{g_{\text{strict}}}{g_{\text{ref}}} = \frac{2/30}{0.50} \approx 13.3\%.$$

Here 13.3% is the retained fraction of the reference gap, not a 13.3-point advantage. The whole strict separation comes from one paired task. Under the strict contract the two methods differ by two of 30 task-trials, so most of the annotation-cued gap does not survive cue removal; together with the A-MEM panel, where lossless summary exceeds RSM, this is the dependence on the information contract noted in Section 8.

H SCIENCEAGENTBENCH TRANSFER AND EXTERNAL PILOTS

This appendix details the 102-task ScienceAgentBench comparison of Section 6.3 (Appendix H.1) and reports three exploratory pilots on small external task sets (Appendix H.2). Outcomes are not pooled across these studies.

H.1 SCIENCEAGENTBENCH-102 PROTOCOL AND PAIRED OUTCOMES

Protocol. The study uses all 102 tasks of the verified split of ScienceAgentBench (Chen et al., 2024). For each task, the two methods share one initial program, and each method then runs one repair trajectory with up to three repairs. Success means that the final program passes the benchmark's native evaluator. The task is the unit of analysis; repair attempts and the two method trajectories are not independent observations. Both methods run in the same single-branch wrapper with a 6,000-token memory cap, a 2,000-token cap on execution feedback, and no ESMB annotations. Planner settings are listed in Table 11.

Memory in the repair wrapper. After each attempt, the wrapper writes two items: the generated program, as an artifact holding the full program text, and its public execution feedback, as a result linked to that program by `derived-from`. Each new program is linked to its predecessor by `supersedes`, and every item carries the same branch. Before each repair, both methods receive the same request: a fixed repair query on that branch, with no focus IDs, phase, expected kind, or required labels. RSM returns up to the two most recent execution results as primary items, each with the program that produced it attached as a neighbor; this wrapper's RSM adapter renders full item text instead of the 140-character lines of Appendix C.1. Fragments are taken in rank order until their text fills the 6,000-token cap, and the fragment that crosses the cap is cut at its end; if the rendered context, with its fragment headers, still exceeds the cap, the wrapper keeps its head and tail and cuts the middle. Repair summary applies grouped lexical retrieval to the same full records and

Table 21: ScienceAgentBench-102: paired native outcomes, one trajectory per method and task. The discordant cells give the exact McNemar test.

Paired outcome	Repair summary succeeds	Repair summary fails	Total
RSM succeeds	38	22	60
RSM fails	8	34	42
Total	46	56	102

places the newest execution result and its program in the first fragment to preserve latest-attempt context (Appendix C.2); RSM receives no such guaranteed first fragment. With one branch and no annotations, the branch-filter exclusion of Section 4 and RSM's annotation-gated cross-branch routes (Appendix C.1) cannot occur, so the comparison exercises typed selection and program–result bundling in a short repair loop.

Outcomes. RSM solves 60 tasks and repair summary 46, a paired gain of 13.7 points with a 95% task-bootstrap interval of $[3.9, 23.5]$ points. Of the 30 discordant tasks, 22 favor RSM and eight favor repair summary (exact two-sided McNemar $p = 0.0161$; Table 21). Per-domain counts are in Table 5(b); the gain is positive in every domain.

H.2 Exploratory External Pilots

The three pilots below are exploratory. Each uses four tasks and one trial per method, so its counts describe individual tasks and support no general claim. Each uses its own wrapper and is reported separately from the 102-task study; unlike the 102-task wrapper, the pilot wrappers pass their own focus IDs to memory. The ScienceAgentBench pilots use tasks 1, 4, 8, and 24; tasks 1 and 8 are illustrated in Appendix I.6. The comparator in the first two pilots is the lexical-preview summary adapter; the 102-task study of Appendix H.1 uses repair summary.

ScienceAgentBench one-shot generation and self-debugging. These are two separate execution protocols over the same four tasks (Table 22). RSM solves one task under one-shot generation and lexical-preview summary solves none; with self-debugging, both solve three.

CORE-Bench. The CORE-Bench (Siegel et al., 2025) pilot runs four native `codeocean_easy` capsules through a shared evidence reader, and both methods answer every native question correctly (Table 23). A separate memory-only evaluation reuses the resulting artifacts and answers and, after a distractor event, asks two authored provenance or disambiguation questions per capsule. RSM answers six of these eight follow-ups correctly and lexical-preview summary three; on the four artifact-path questions among them, the counts are four and two.

Staged ScienceAgentBench pilot. This pilot runs the same four tasks through a six-stage wrapper: task inspection, branch choice, initial execution, failure interpretation, repair choice, and final execution. It is the only external setting that includes a branch choice. Its memory requests carry required labels and phase at every stage, and, as in LongHorizon, the planner sees opaque fragments. Its cited grounding counts required labels in planner-cited fragments, unlike the grounding of Appendix A, which counts them in the returned context. RSM reaches cited grounding 1.000 but solves only two of the four tasks, the same as no memory and fewer than raw-event retrieval and truncated history, which solve three each (Table 24).

Table 22: Exploratory ScienceAgentBench pilots on tasks 1, 4, 8, and 24: native successes out of four under two separate execution protocols.

Protocol	RSM	Lexical-preview summary
One-shot generation	1/4	0/4
Self-debugging	3/4	3/4

Table 23: CORE-Bench pilot on four capsules and derived memory follow-ups. Native and derived counts come from separate evaluations and are not pooled.

Evaluation	RSM	Lexical-preview summary
Native capsule success	4/4	4/4
Derived follow-up accuracy	6/8	3/8
Derived artifact-path accuracy	4/4	2/4

Table 24: Staged ScienceAgentBench pilot on tasks 1, 4, 8, and 24: native successes out of four and cited grounding averaged over the wrapper's checkpoints.

Memory method	Native success	Cited grounding
No memory	2/4	0.000
Rolling-preview summary	1/4	1.000
Raw-event retrieval	3/4	0.958
Truncated history	3/4	0.896
Hashed-vector retrieval	2/4	0.958
RSM	2/4	1.000

I Worked Examples

The worked examples below cover the four ESMB suites and two ScienceAgentBench tasks from the exploratory pilots (Appendix H.2). Labels are the fixture identifiers. Fixtures, requests, and returned fragments are deterministic. Outcomes are from trial 0 of the `gpt-5.2` primary run for the recovery and supersession examples, suite-level for the other ESMB examples, and from the single pilot trial for the ScienceAgentBench examples.

I.1 ESMB-Fixed: Failed-Experiment Recovery

Table 25 shows the running example of Figure 2 and Section 4, from the writes the memory ingests to the labels the scorer checks. It also lists what each planner sees at the decision.

The label-support cues are computed from the returned labels, so a method that returns `r_lr_null` also changes the cues its planner receives.

I.2 ESMB-Fixed: Stale Supersession

In the stale-supersession task (Table 26), the decisive cross-branch item is again the old branch's null result, `r_old_null`. As in recovery, both planners also see the rerun observation and the label `r_old_null` outside memory.

I.3 ESMB-TaskVariant: Matched Failed-Experiment Recovery

TaskVariant restates each Fixed task on a new surface. Table 27 shows the recovery task and its variant.

Because the decisive record again lies on `main` while the request names `alt`, the lexical baselines' branch filter excludes it on both surfaces. At the decision request, lexical-preview summary returns only the alternate-branch records, and RSM attaches the null result to the alternate hypothesis as in Table 25. The suite tests whether a method's behavior is robust to task surface and adds no new workflows. Three blinded raters judged all five Fixed–TaskVariant pairs equivalent by majority vote (Appendix B.4). Since RSM's grounding is 1.000 in every Fixed and TaskVariant run (Table 15) and $U_a = R$ on the cross-branch tasks, RSM's Fixed failures come from the action clause of J.

Table 25: ESMB-Fixed failed-experiment recovery under the annotation-cued contract. Rows spanning both columns are common to the two methods; split rows give lexical-preview summary (left) and RSM (right). The last row reports trial 0 of the `gpt-5.2` primary run.

Field	Lexical-preview summary	RSM
Goal	Decide whether the recovery branch is justified by the failed main-branch result and the existing batch-size evidence.	
Initial writes	`q_train` (open question, `main`) "Why did training stall?"; `h_lr` (hypothesis, `main`) "Learning-rate hypothesis"; `h_batch` (hypothesis, `alt`) "Batch-size recovery hypothesis"; `ev_batch` (evidence, `alt`) "Gradient-instability note", which `supports` `h_batch`.	
Checkpoint 1	`ctx_lr_failure` on `main`, with one permitted action, `inspect_lr_failure`, which runs a stored analysis script on the sweep metrics. Its authored effect writes the decisive result `r_lr_null` (result, `main`) "Learning-rate sweep stayed flat", which `contradicts` `h_lr`.	
Checkpoint 2 (decision)	`ctx_recovery` on `alt`; query "decide whether the recovery branch is justified by the failed branch"; phase branch reassessment; expected kind result; budget $B = 3$ fragments. Focus IDs: `r_lr_null`, `h_lr`, `h_batch`. Required labels $R = \{$`r_lr_null`, `h_batch`, `ev_batch`$\}$.	
Action set	`run_batch_fix` runs the recovery experiment; `inspect_batch_signal_only` re-reads the stored note. Both end the task. The prompt gives them the generic titles "Advance with execution" and "Advance with review".	
Returned context	One fragment carrying `h_batch` and `ev_batch`. With `alt` records present, the branch filter admits only these two as candidates, so `r_lr_null` is never ranked (exclusion).	One fragment with primary `h_batch`, carrying `ev_batch` and `r_lr_null` as neighbors (Appendix C.1). A second fragment holds `ev_batch`.
Planner inputs outside memory	In the released harness, both prompts show the checkpoint description, the labels observed so far, which include `r_lr_null`, the completed action, and the recent observation "The learning-rate sweep was flat across all candidate values." Both also receive a decision protocol that prefers the action with stronger label support, and label-support cues computed from their own returned labels (Appendix B.3).	
Label-support cues	`run_batch_fix`: 1 of its 3 prompt-listed labels present. `inspect_batch_signal_only`: 2 of 2.	`run_batch_fix`: 2 of 3 present (`h_lr` missing), and flagged as using a visible cross-branch result, which the branch-reassessment guidance says to prefer. `inspect_batch_signal_only`: 2 of 2.
Prompt vs. scoring sets	The prompt lists `r_lr_null`, `h_lr` and `h_batch` as the label requirements of `run_batch_fix`, and `h_batch` and `ev_batch` for `inspect_batch_signal_only`. The authored support set of `run_batch_fix` is $U_a = \{$`r_lr_null`, `h_batch`, `ev_batch`$\} = R$, which omits `h_lr`, so the scored set differs from the prompt's list.	
Evaluation	$J = \mathbf{1}[R \subseteq T]\,\mathbf{1}[U_a \subseteq T]\,\mathbf{1}[A]$, where T is the set of labels returned at `ctx_recovery`, $a =$ `run_batch_fix`, and A indicates that a is executed (Appendix B.2). Because `r_lr_null` $\notin T$ for lexical-preview summary, its J is 0 whichever action is selected.	
Trial-0 outcome	Selects `inspect_batch_signal_only`; $J = 0$.	Selects `run_batch_fix`; $J = 1$.

I.4 ESMB-Replay: Exclusion in the Branch-Switching Trace

Replay fixes the writes, requests, actions and outcomes, so only what an adapter returns varies. Table 28 shows one annotated request of the Replay branch-switching trace, where the pre-budget candidate sets separate exclusion from displacement.

Appendix E.2 gives two displacement cases for comparison.

Table 26: ESMB-Fixed stale supersession under the annotation-cued contract, in the layout of Table 25. The last row reports trial 0 of the `gpt-5.2` primary run.

Field	Lexical-preview summary	RSM
Goal	Decide whether the superseding curriculum branch is justified by the stale main-branch failure and the supersession relation.	
Initial writes	`h_old` (hypothesis, `main`) "Old optimizer hypothesis"; `h_new` (hypothesis, `alt`) "Curriculum hypothesis", which `supersedes` `h_old`; `ev_curriculum` (evidence, `alt`) "Curriculum branch cue", which `supports` `h_new`.	
Checkpoint 1	`ctx_old_null` on `main`, with one permitted action, `inspect_old_null`, which summarizes the stored optimizer-only rerun. Its authored effect writes the decisive result `r_old_null` (result, `main`) "Optimizer-only rerun stayed flat", which `contradicts` `h_old`.	
Checkpoint 2 (decision)	`ctx_supersede` on `alt`; phase branch reassessment; expected kind result; $B = 3$ fragments. Focus IDs: `r_old_null`, `h_old`, `h_new`. $R = \{$`r_old_null`, `h_new`, `ev_curriculum`$\}$.	
Action set	`run_curriculum_validation` runs the validation, with $U_a = R$ (the prompt instead lists `r_old_null`, `h_old` and `h_new`); `inspect_curriculum_note_only` re-reads the stored note. Both end the task.	
Returned context	One fragment carrying `h_new` and `ev_curriculum`; the branch filter excludes `r_old_null` before ranking.	One fragment with primary `h_new`, carrying `ev_curriculum` and `r_old_null` as neighbors; a second fragment holds `ev_curriculum`.
Evaluation	J as in Table 25, at `ctx_supersede` with $a =$ `run_curriculum_validation`.	
Trial-0 outcome	Selects `inspect_curriculum_note_only`; $J = 0$.	Selects `run_curriculum_validation`; $J = 1$.

Table 27: ESMB-TaskVariant: the failed-experiment recovery task restated on a new surface. Object names (e.g., the decisive result becomes `r_dropout_null`), the scientific situation and the action identifiers change; the decisive record's role and branch placement, the branch placement of the alternate evidence, the phase, and the sizes of R and U_a do not.

Element	Fixed	TaskVariant
Situation	Training stalled; learning-rate versus batch-size explanations	A run collapsed after a dropout-scale change; dropout versus microbatch explanations
Main branch	`q_train`, `h_lr`	`q_dropout`, `h_dropout`
Alternate hypothesis and evidence	`h_batch`; `ev_batch`, which `supports` `h_batch`	`h_microbatch`; `ev_microbatch`, which `supports` `h_microbatch`
Decisive write	`r_lr_null` on `main`, written after `ctx_lr_failure`	`r_dropout_null` on `main`, written after `ctx_dropout_probe`
Decision checkpoint	`ctx_recovery` on `alt`	`ctx_microbatch_recovery` on `alt`
Phase, kind, budget	branch reassessment, result, $B = 3$	branch reassessment, result, $B = 3$
R $(= U_a)$	`r_lr_null`, `h_batch`, `ev_batch`	`r_dropout_null`, `h_microbatch`, `ev_microbatch`
Gold action	`run_batch_fix`	`launch_microbatch_recovery`
Distractor action	`inspect_batch_signal_only`	`inspect_microbatch_signal_only`

I.5 ESMB-LongHorizon: Failed-Experiment Recovery, Long Variant

LongHorizon separates the decisive write from its use by intervening checkpoints and hides source labels from the planner (Appendix F). Table 29 shows the long failed-experiment recovery variant, with temporal lag $\Delta = 6$ and generator-assigned structural lag $G = 4$ (Table 18).

Table 28: ESMB-Replay, request `ctx2` of the Replay branch-switching trace. There is no planner and no task outcome; the graded quantity is which labels are returned. Candidate counts are pre-budget, so a method with a single candidate lost `r_main1` before ranking rather than to the fragment budget.

Field	Example
Writes before the request	`q1` (open question, `main`) "Which ablation explains the gain?"; `h_main` (hypothesis, `main`) "Regularization hypothesis"; `exp_main` (experiment, `main`), which `tests` `h_main`; `r_main1` (result, `main`) "Regularization branch was null", which `contradicts` `h_main`; `h_alt` (hypothesis, `alt`) "Data-order hypothesis".
Request	`ctx2` on `alt`; $B = 3$ fragments; phase branch reassessment; expected kind hypothesis. Focus IDs: `h_alt`. Relevant labels $L = \{\texttt{h_alt}, \texttt{r_main1}\}$; required labels $R = \{\texttt{h_alt}\}$.
Returned context	No memory returns nothing. Lexical-preview summary and lexical fragment retrieval each have a single pre-budget candidate and return `h_alt` alone, because with one `alt` record stored the branch filter leaves the four `main` records out of the candidate set. RSM's candidate set has four entries, including `r_main1` added by the phase-gated branch-reassessment attachment, and it returns `h_alt` with `r_main1` attached as a neighbor.
Scoring	Recall at this request is $\lvert T \cap L\rvert/\lvert L\rvert$: $1/2$ for both text baselines and $2/2$ for RSM. Every method except no memory returns R, so grounding is unaffected. Trajectory recall is 0.833, 0.833 and 1.000 respectively; suite aggregates are in Table 16.

Table 29: ESMB-LongHorizon, terminal checkpoint `cp10` of the long failed-experiment recovery variant. The planner sees opaque fragments and is scored on its selected action together with the labels carried by the fragments it cites, while the memory request still resolves focus IDs.

Field	Example
Trajectory	Along the gold action path, twelve writes across `main`, `alt` and an `analysis` branch: `q_objective` and `h_main` on `main` and `h_alt` on `alt` initially; one setup record at each of `cp01`–`cp03`; the decisive result `r_main_failed`, which `contradicts` `h_main`, and its artifact on `main` at `cp04`; and one intervening artifact at each of `cp05`–`cp08`. The reassessment checkpoint `cp09` writes nothing. With use at `cp10`, $\Delta = t_u - t_w = 10 - 4 = 6$ checkpoints; the generator assigns $G = 4$.
Terminal request	`cp10` on branch `recovery`, which holds no writes; phase decision update; expected kind decision; $B = 6$ fragments. Focus IDs: `q_objective` and `r_main_failed`. The request carries no required labels, and the gold action `run_alt_recovery` and the grounding requirement on `r_main_failed` stay hidden from the planner.
Returned context	Hashed-vector retrieval ranks all twelve records across branches and returns `q_objective`, the four intervening artifacts, and `cp01_state`, an item of kind decision written at `cp01`. `r_main_failed` is a candidate but is displaced, because the artifacts score higher in hashed-vector similarity to the query, and `cp01_state` matches the expected kind. RSM returns four fragments from the one-hop neighborhood of the focus IDs; `r_main_failed` appears in three (as a primary item, as a neighbor of its artifact, and as a neighbor of `h_main` through its `contradicts` edge), and the fourth holds `q_objective`.
Planner interface	Fragments are rendered as `C1`, `C2`, … without source labels. The four actions carry titles and descriptions only, with no label requirements and no label-support cues.
Evaluation	Success requires `run_alt_recovery` at `cp10` and a planner-cited fragment carrying `r_main_failed`, so returning the record without citing it does not suffice.
Scope	The focus IDs name the decisive record, so RSM reaches it through the focus neighborhood whatever the lag, whereas hashed-vector retrieval ranks it against the intervening writes by similarity to the query (Appendix F). Suite-level joint successes with `gpt-5.2` are $45/55$ for RSM and $35/55$ for hashed-vector retrieval.

I.6 ScienceAgentBench Tasks 1 and 8: ClinTox and DKPES

In ScienceAgentBench (Chen et al., 2024), task 1 asks for a molecular prediction program that writes a specified output file, and task 8 requires a feature-selection analysis and a corresponding figure,

Table 30: ScienceAgentBench tasks 1 and 8, from the exploratory four-task self-debugging pilot (Appendix H.2), not from the 102-task study. Native success is the benchmark evaluator's verdict on the saved artifact; there is no ESMB label predicate.

Field	Task 1 (ClinTox)	Task 8 (DKPES)
Domain	Computational chemistry	Bioinformatics
Task	Train a multitask model on ClinTox to predict drug toxicity and FDA approval status.	Identify DKPES chemical features predictive of signal inhibition by backward feature selection with logistic regression.
Inputs	Training and test CSV files with molecular SMILES and the `CT_TOX` and `FDA_APPROVED` labels, together with the task instruction and dataset description.	DKPES training and test CSV files with signal-inhibition values and chemical descriptors, together with the task instruction and dataset description.
Output contract	Save test-set predictions, including SMILES and positive-label probabilities, to `pred_results/clintox_test_pred.csv`.	Binarize signal inhibition at an appropriate threshold, run the selection, and save the accuracy-versus-feature-count plot to `pred_results/dkpes_feature_selection_analysis_pred.png`.
Repair context	The generated program and its execution feedback; repairs must keep the two prediction targets, the molecular identifiers and the output location.	The generated program and its execution feedback; repairs must keep the binarization requirement, the backward-selection procedure and the accuracy-versus-feature-count relation.
Evaluation	The native program-output evaluator scores the saved predictions after execution.	The native figure evaluator scores the saved plot with `gpt-4o` visual judgment; this model belongs to the benchmark's evaluator, not to the planner.
Pilot outcome	Lexical-preview summary and RSM both fail in the recorded self-debugging trial.	Lexical-preview summary and RSM both pass in the recorded self-debugging trial.

which the benchmark's native evaluator judges visually. Table 30 shows both specifications and pilot outcomes.

J Reproducibility, Compute, and Licenses

This appendix covers the language models in the evaluation loop, compute, the terms of the external assets, and the scope of the intended release. Per-study run settings, including planner model, decoding, and context budget, are tabulated once in Table 11 (Appendix B.2).

Evaluated language models. Each planner selects actions from the context a memory method returns and is held fixed across methods within a comparison. The A-MEM and StructMem comparators call their own memory-side generation and embedding models, which are listed with those comparisons (Appendix G.1). ScienceAgentBench's native evaluator scores figure outputs with a `gpt-4o` visual judge, as in the feature-selection task of Appendix I.6; this judge is part of the benchmark's scoring.

Compute. No model is trained or fine-tuned. ESMB runs consist of Python orchestration, stored-artifact task execution, file I/O, and cached-response management around calls to hosted inference services; the memory methods run on commodity CPU-class compute with storage for traces. External tasks additionally execute the native task programs, which can fit task-specific predictive models, so their compute depends on the task. A-MEM and StructMem add memory-side model calls, reported with those comparisons (Appendix G.1). We do not report total inference and task-execution cost per method on a common basis.

Licenses and terms for existing assets. Table 31 lists the external assets, their license or terms, and their treatment in the intended release.

Table 31: External assets used in this study, their license or terms, and their treatment in the intended release.

Asset	License or terms	Treatment in the release
ScienceAgentBench benchmark code, task data, and native evaluator (Chen et al., 2024)	code under the MIT License; most task data under CC BY 4.0, with some tasks retaining their source repositories' licenses; the maintainers ask that the unpacked benchmark files not be redistributed	cite the upstream benchmark and include only the wrapper code, task manifests, and derived results; do not redistribute upstream task data
CORE-Bench benchmark code and manifests (Siegel et al., 2025)	upstream repository under the MIT License and benchmark-access terms from the benchmark maintainers	cite the upstream benchmark and include only derived manifests, result summaries, and the documentation needed for this study
Official A-MEM implementation (Xu et al., 2025), pinned revision	upstream repository under the MIT License	cite the upstream repository and pinned revision; upstream code is not redistributed
Official StructMem code (Xu et al., 2026), pinned revision, adapted	terms of the upstream repository	cite the upstream repository and pinned revision; upstream code is not redistributed
MiniLM sentence embeddings (`all-MiniLM-L6-v2`), used by A-MEM	terms of the upstream model repository	used through the A-MEM code; no weights are redistributed
`text-embedding-3-small` embeddings, used by StructMem	service access terms	accessed through an inference service; nothing is redistributed
Hosted language models: planners, memory-side models, and evaluator judges	service access terms	no model weights or run records are redistributed; only evaluation summaries are released

Release scope and documentation. The intended release comprises the benchmark code, task fixtures, method implementation, and evaluation summaries, documented by the ESMB specification (Appendix B) and the memory-method descriptions (Appendix C). Run records, caches, and the adapter and planner configurations of the LLM-planner suites are not part of the release, so the reported planner outcomes are not independently reproducible from it. ESMB-Replay is deterministic and has no planner, so its reported metrics follow from the released fixtures. Upstream assets are handled as in Table 31. A retrospective three-rater audit of the base-suite episodes is reported in Appendix B.4.