WHAT DOES CONTEXT COMPRESSION COST AN AGENT? INTERACTION COSTS UNREVEALED BY TASK-COMPLETION METRICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Task completion is the standard metric for evaluating context compression, yet it is an incomplete measure: compression can substantially increase an agent's interaction cost — the reacquisition of state it dropped — without any detected change in completion. We frame completion-only evaluation as a lossy projection of the full evaluation outcome and introduce a controlled runtime protocol that measures this reacquisition cost in a bounded-horizon agent. Across three models and two task regimes, retrieval tool calls increase in every one of six model–regime comparisons, five of them after Holm correction, while execution calls change little. At the pre-specified $5\times$ point no completion change is detected in any cell, whereas the retrieval signal already appears: on GPT-5.6 Sol completion is unchanged ($[-5, +6]$ pp; MDE 8.4 pp) while retrieval rises by 24.2 calls. At more aggressive compression, completion degrades as reacquisition consumes the horizon. Oracle restoration and retention interventions probe the mechanism: restoring the queryable dropped state removes most of the added tool cost, while replacing D content with fabricated irrelevant state raises retrieval without a detected completion change. Random retention does not differ detectably from an offline hindsight oracle, suggesting state validity and presence matter more than fine-grained selection in this regime. The effect is not universal: fact-preserving extractive and abstractive operators avoid the surge, while the dropping operator produces it, and the surge reappears in τ-bench; the ALFWorld probe dropped no context and is uninformative about compression. Context compression can therefore incur hidden interaction cost when execution-relevant state must be reacquired, and completion alone can miss that cost.

1 INTRODUCTION

Long-horizon agents accumulate trajectories that quickly exceed any context window, so modern runtimes compress — via summarization, sliding windows, or retrieval filters — as a matter of course. Existing evaluations ask whether task performance survives compression. We ask a complementary question: *what runtime cost can accumulate before that performance changes?* Task completion is an incomplete measure of the runtime cost of context compression — it does not by itself reveal the interaction cost incurred when the agent must reacquire compressed-away state. Compression is costly when dropped execution-relevant state must be reacquired through additional interaction under a bounded budget; completion may show no detectable change while that burden grows.

Prior work on context compression asks what to keep, whether information survives, whether it is used, and which strategy wins on aggregate capability and efficiency (Kang et al., 2026; Kontonis et al., 2026; Zhu et al., 2026; Trukhina & Vashkelis, 2026; Kwon, 2026; Gasté, 2026; Lodha et al., 2026; Yehudai et al., 2025; Liu et al., 2025). Related behavioral probes exist: Min et al. (2026) measure blocked actions and repeated exploration after compression in a refetchable environment, and Fan et al. (2026) study an explicit recovery mechanism for elided coding-agent context. We instead quantify the spontaneous re-issuance of ordinary environment reads as a distinct retrieval cost C_R over the full bounded agent loop. We ask a fifth question: *what does it cost the agent to re-acquire what was dropped, and is that cost visible to standard metrics?*

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.

We introduce a controlled measurement protocol for this reacquisition cost. The setup fixes a single interaction horizon (24 turns), varies compression severity via a sliding-window operator, spans two task regimes that differ in how much execution-relevant state must be discovered at runtime, and *manipulates the availability of dropped state* through controlled oracle restoration; we decompose the agent's tool calls into *retrieval* (re-acquiring external state) versus *execution* (performing the task). We run the protocol across three models — DeepSeek, Qwen, and GPT-5.6 Sol — to separate what holds across models from what is model-specific (DeepSeek at a full severity sweep, Qwen and GPT-5.6 Sol at a fixed $5\times$ ratio; GPT-5.5 cross-generation replication in Appendix A.2).

Our contributions are:

1. **Evaluation non-identifiability under context compression.** Task completion is not an identifying projection of interaction cost: we characterize this at the measurement level as a non-identifiability result and realize it under controlled intervention; the contribution is a measurement protocol and a controlled realization, not a new theory of non-identifiability. GPT-5.6 Sol pays a large reacquisition cost ($+24.2$ calls) with no detected completion change.
2. **A mechanistic decomposition of the hidden cost.** We decompose the reacquisition burden along a recoverability axis — externally queryable task state (D) versus history-dependent state (R) — and show that retrieval calls increase in all six model–regime comparisons, with injecting D^* attenuating the added tool cost but not consistently recovering completion.
3. **Controlled probes of state retention and validity.** Retention interventions separate how much state is retained, which state is retained, and whether retained content is valid. Fine-grained selection among real task-relevant atoms has limited marginal value at the digest budgets we test, which retain most of it (Random vs. Hindsight: no detected difference), whereas replacing task-relevant D content with fabricated irrelevant state sharply increases retrieval. The content effect replicates on a second GPT generation at a loose digest budget (Appendix A.5); operator probes characterize where the surge is absent, and an external environment reproduces it weakly.

2 Related Work

Prior work on context compression spans four categories: *compression methods* (what to keep (Kang et al., 2026; Kontonis et al., 2026; Xu et al., 2026; Zhu et al., 2026; Jiang et al., 2023)), *recoverability evaluation* (whether information survives or can be recovered (Trukhina & Vashkelis, 2026; Kwon, 2026; Wang et al., 2026; Zerhoudi et al., 2026; Zhang, 2026)), *long-context utilization* (whether surviving information is used (Gasté, 2026)), and *aggregate efficiency evaluation* (completion alongside tokens, latency, and runtime (Lodha et al., 2026); success alone is insufficient and must be paired with cost (Yehudai et al., 2025; Liu et al., 2025; Han et al., 2026)). Each line evaluates what compression preserves, recovers, or achieves. To our knowledge, none provides a diagnostic for the interaction cost that completion-centric evaluation does not expose — where the added interaction goes, and whether completion tracks it.

A parallel line asks what the *right metric* for an agent is. Completion alone is a blunt instrument: it compresses process-level variance (the "completion fallacy" (Hou et al., 2026)), obscures graded differences among failing agents (Khanal et al., 2026; Ma et al., 2024), and reveals horizon-dependent trade-offs (Sah et al., 2026); the cost side is priced through tool-calling accuracy taxes (Zhang et al., 2026), cost-optimal planning gaps (Liu et al., 2025), and cost–performance studies of compound agent designs (Bogdanov et al., 2026). Reacquisition cost is recognized only operationally (Factory Research, 2025; bolnet, 2026), without causal attribution. To our knowledge, none defines a black-box, model-agnostic measure of how much interaction budget an agent spends to reacquire state that compression removed, nor manipulates that state's availability causally.

Closest contrasts. CostBench (Liu et al., 2025) and Tool-Use Tax (Zhang et al., 2026) study cost or accuracy effects under intact context; Control Under Compression (Hou & Yang, 2026; Zheng et al., 2026) measures reliability under compression; Token Reduction $\neq$ Cost (Weinberger & Hozez, 2026) shows that token savings need not equal billing savings. To our knowledge, none directly manipulates dropped state and measures the resulting reacquisition cost. *Lost in Compaction* (Gasté, 2026) attributes compaction failures to attention dilution despite information remaining in context;

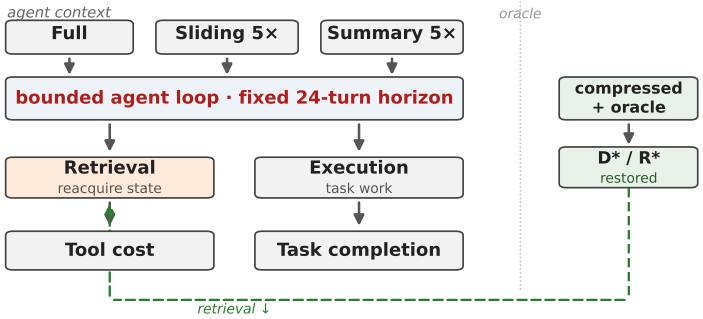

Figure 1: Measurement protocol. Context conditions (Full; Sliding $5\times$; Summary $5\times$) feed a bounded agent loop under a fixed 24-turn horizon. Missing execution-relevant state adds *retrieval* (re-querying the environment) on top of *execution* (task work), and the loop yields completion and tool cost jointly: completion can stay flat while the interaction cost rises (Figs. 2–4). A parallel branch injects D^* (queryable) or R^* (history-only) state, with different effects for the two classes. Digest budget B is a retention parameter (Fig. 5).

our sliding intervention instead makes dropped state genuinely absent and measures the resulting reacquisition. *Compress the Context, Keep the Commitments* (Trukhina & Vashkelis, 2026) and *Reclaim Evaluation* (Kwon, 2026) study recoverability as a property of representations or single-shot recovery, whereas we measure cascading recovery inside an agent loop under a bounded interaction budget.

3 Method

Figure 1 (measurement protocol overview): an agent context feeds either a full / fact-preserving context or a compressed (sliding) context; missing execution state diverges into *retrieval* (reacquisition) versus *execute* (task work), which together consume the 24-turn horizon and determine completion / tool cost. A parallel oracle branch (compressed + $D^*/R^* \rightarrow$ ground-truth state injected $\rightarrow$ retrieval falls) shows recoverability is a manipulated variable, not a passive observation.

3.1 Evaluation outcomes and completion as a lossy projection

We define the evaluation object this paper measures. For a context-management condition m we record a triple of measured outcomes.

Definition 1 (Evaluation outcome). $Y(m) = (Q, C_R, C_E)$, where Q is task completion, C_R retrieval tool calls, and C_E execution tool calls, under a fixed interaction horizon. We interpret C_R as state re-acquisition effort and C_E as task work; $C = C_R + C_E$ is the *tool-level interaction cost* — the agent's budgeted currency, not a claim about wall-clock cost.

Completion-centric evaluation applies a projection $\pi_Q(Y) = Q$ to this triple. Let $\sim_Q$ denote that the pre-specified paired test (paired Wilcoxon signed-rank; §3.4) does not detect a completion difference between two conditions. A completion-only evaluation can preserve cost-aware ordering only if $m_1 \sim_Q m_2$ implies the two conditions also do not differ materially in interaction cost.

Remark 1 (Non-identifiability of completion-only evaluation). The completion-only projection π_Q discards the interaction-cost dimensions of Y. Distinct outcome states with different interaction costs can therefore map to the same completion value. Under our protocol, this non-identifiability is realized empirically by conditions whose completion differences are not detected by the pre-specified paired test while their retrieval costs differ substantially.

Empirical counterexample. Full vs. Sliding ($5\times$) on GPT-5.6 Sol under High-IR: completion $77\% \rightarrow 77\%$ (paired $p = 1.00$; 95% CI $[-5, +6]$ pp, containing zero; MDE 8.4 pp), while retrieval $18.4 \rightarrow$

42.6 (paired $p = .002$; Cohen's $d = 2.56$; 95% CI $[+18.5, +29.6]$, excluding zero). A completion-only evaluator would be indifferent at the point estimate, despite a $2.3\times$ increase in retrieval cost.

3.2 AGENT AND TASK ENVIRONMENT

We study a minimal tool-using agent on a deterministic project-planning environment (**IRBench**): **10 tasks** over a small set of **resources** (capacity one), with hidden execution constraints revealed only through interaction. The agent has four tools. Three read environment state — `get_task_info`, `check_dependency`, `query_resource` — and count as *retrieval* calls (C_R); `execute` changes state and counts as an *execution* call (C_E). The split is fixed by tool identity, so both are computable from the environment's own call log. The three read tools are the only channel to the task graph, and the dropping operator removes prior observations of it, so recovering a dropped observation requires re-requesting state the agent already held. A supplementary check on a later DeepSeek checkpoint measures exactly that: the trace-level diagnostic Reacquisition Share, $\alpha_R = N_{\text{reacq}}/(N_{\text{reacq}} + N_{\text{first}})$, is 0.584 under Sliding (27.5 re-acquisitions against 19.6 first reads per run) and 0 under Full (Appendix A.11).

The environment is **deterministic**: a seed fully determines the world instance, so any condition can be replayed on the identical task set. Two task regimes vary the density of hidden, execution-relevant facts: **High-IR** contains execution-revealed state that is costly to lose; **Low-IR** reveals one failure condition, one ordering constraint, and no holds. The task graph and resource state can be queried again at any time; a previously revealed failure condition or fail-until count exists only in the execution history.

The agent receives a fixed system prompt and completes all ten tasks within a **fixed interaction horizon of 24 turns**. A turn is one model call plus every tool call returned in it, all executed and counted individually, so the horizon bounds model calls rather than tool calls: a turn may carry several, letting C_R grow by tens of calls while the turn count stays pinned near the ceiling. Completion counts tasks finished when the horizon ends.

3.3 CONTEXT CONDITIONS: OPERATORS AND ORACLE INTERVENTIONS

We hold the task instance, system prompt, tool schema, and model fixed, varying **only** the context delivered to the model (applied online, before each response). We compare **Full**, **Sliding**, extractive fact-preserving **Summary**, and **Oracle**-restored contexts. Full retains all history; the compressed variants are matched to each other in token budget. Sliding is *deletion* (not summarization), so dropped state is genuinely absent — testing "absent $\rightarrow$ reacquire" rather than "present but ignored". At ratio r it is applied online at every model call, targeting $1/r$ of the current history (Appendix A.11). Summary replaces dropped early turns with observed state facts, discarding only the reasoning text; it extracts them deterministically by rule, with no model in the loop, and only the abstractive variant uses a model — the same one under test. Oracle injection is a trailing message in a fixed position, re-injected at every call from turn 1, leaving the system prompt untouched.

The oracle supplies environment ground truth for the state sliding can drop, split into two recoverability classes: D^*, externally queryable state (the task graph), recoverable in principle with more tool calls; and R^*, history-dependent state, not recoverable from any single public query. They define the manipulated variable: the availability of this state.

3.4 PROTOCOL AND ANALYSIS

Each condition uses the same **10 paired seeds** (10 task instances per seed); retention interventions use **20**. Statistics are per-seed paired: Wilcoxon signed-rank tests and bootstrap 95% intervals (exact tests and the design's resolution in Appendix A.13, Table 11). We pre-specify the primary comparisons (Full vs. Sliding on task completion, tool calls, and retrieval calls). For multiple-comparison control, the primary family is the six retrieval comparisons (three models $\times$ two regimes); Holm–Bonferroni at $\alpha = 0.05$ leaves five of six significant under the pre-specified paired test (adjusted $p \leq .047$; only Qwen High, adj. $p = .088$). Remaining pairwise tests are exploratory. Models: **DeepSeek**, **Qwen**, and **GPT-5.6 Sol**. All were evaluated without an explicit extended-thinking mode, at temperature 0.3 without a fixed API seed, to keep the observable interaction protocol as comparable as possible across providers.

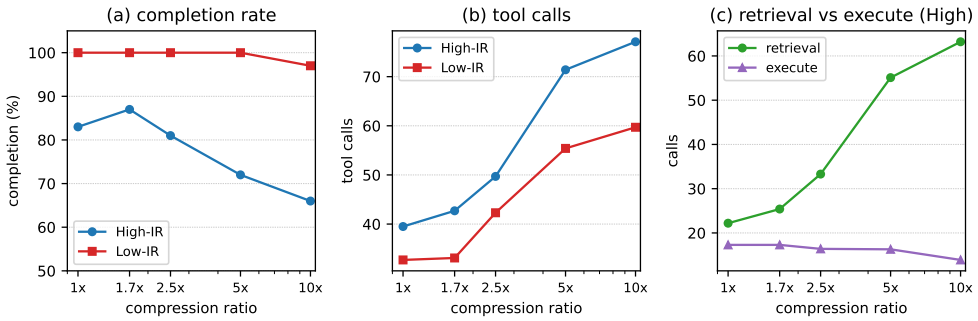

Figure 2: Severity sweep on High-IR across compression ratios (log axis): completion (left), tool calls (center), and the retrieval–execution split (right).

3.5 Retention interventions as controlled probes

The oracle result is consistent with restoring queryable dropped state reducing reacquisition cost. This raises a question about *granularity*: at what level does retained state matter to the agent's interaction cost — how much is retained, which atoms are retained, and whether the retained content is valid and task-relevant? We address it with a family of retention interventions, applied online at a fixed compaction point (turn 10, t_c), each taking the observed trajectory up to t_c, extracting the set of observed state atoms, and re-rendering a compressed prefix under a digest budget $B \in \{265, 100, 50\}$ tokens. We vary retention amount, selection policy, and content validity; the full policy matrix is in the appendix.

4 Experiments

We follow the protocol of §3.4. Unless stated otherwise, "completion" is the fraction of the ten required tasks completed within the fixed 24-turn horizon; "tools" is the per-run tool-call count, decomposed into retrieval and execution calls.

4.1 Result 1: Completion misses interaction cost

4.1.1 Severity sweep (Figure 2; Table 2, appendix)

Varying compression severity with the sliding-window operator ($r \in \{1, 1.7, 2.5, 5, 10\}$) on High-IR:

Completion declines only at the most aggressive ratios ($5\times$: -11pp, $p = .25$; $10\times$: -17pp, $p = .016$, the only significant completion change in our study). Ten paired seeds on a binary outcome give the test limited power, so we read the $5\times$ cell as a failure to detect a change, not as evidence that completion is unaffected. Retrieval, by contrast, is already significantly elevated at $5\times$ ($p = .002$): the cost signal is detected at milder compression than completion is. The decomposition shows *where* the extra interaction goes — between $1\times$ and $10\times$ retrieval rises $22.2 \rightarrow 63.2$ ($+41.0$, $p < .01$) while **execute calls do not increase** (they fall $17.3 \rightarrow 13.9$). Under Low-IR, which reveals far fewer hidden facts, the same operator is near-costless in completion (100% until $10\times$).

4.1.2 Cross-model generality (Figure 3; Table 3, appendix)

We repeat the Full vs. Sliding ($5\times$) comparison on Qwen and GPT-5.6 Sol (High and Low); the full matrix is in Appendix A.2. The $5\times$ comparison point was pre-specified before the severity sweep. Across all six model–regime comparisons, retrieval calls increase (five of six significant under the pre-specified paired test after Holm correction), while no completion change is detected in any cell (all $p \geq 0.125$): DeepSeek's degradation is suggestive, Qwen's completion response is below resolution, and none is detected for GPT-5.6 Sol ($77\% \rightarrow 77\%$, $p = 1.00$) while its tool cost rises by $+68\%$ (almost entirely retrieval; execute $+1.2$, n.s.). The retrieval response is consistent across models; its conversion into reduced completion is not.

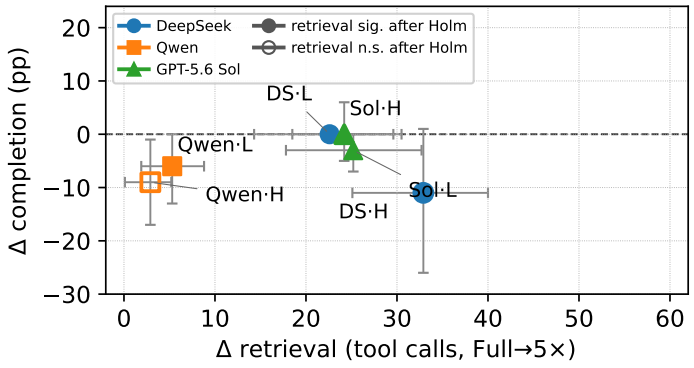

Figure 3: Two-dimensional phase map: the six cells occupy two distinct axes — high retrieval growth with completion loss (DeepSeek High; n.s.), high retrieval growth with no detected completion change (GPT-5.6 Sol, both regimes; DeepSeek Low is at ceiling with zero observed paired variance), and little retrieval growth with a mild drop (Qwen, near the design's resolution in both regimes). The per-cell tool decomposition (Full vs. $5\times$, retrieval vs. execution) is in Appendix A.2.

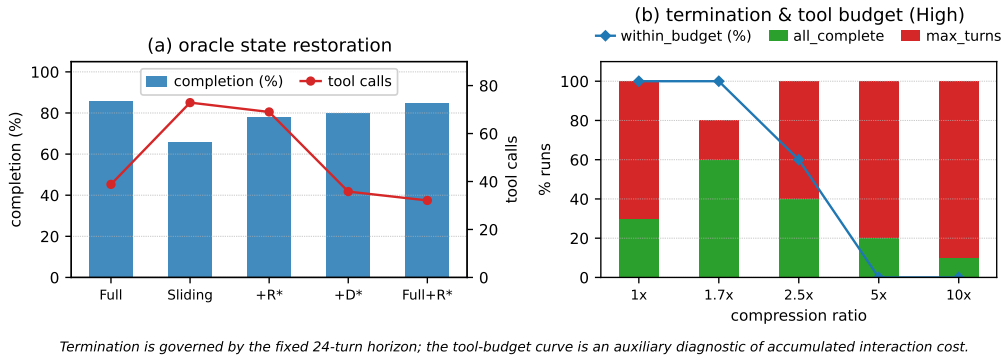

Figure 4: Oracle intervention: adding back D^* removes most of the compensatory tool calls; both recover much of this batch's completion gap. Panel (b): termination and tool-budget diagnostics under the fixed horizon.

4.2 Result 2: Why does the cost appear? Recoverability and retention interventions

4.2.1 Restoring queryable dropped state substantially attenuates the added cost (Figure 4; Table 4, appendix)

In this DeepSeek batch, restoring D^* removes most of the compensatory tool calls (72.9 $\rightarrow$ 35.8) and recovers most of the completion gap (66% $\rightarrow$ 80%); injecting R^* helps completion (+12pp) but leaves tool calls high (69.0). The asymmetry is consistent with what is still missing: D is queryable, so while it is absent the agent keeps paying to re-query for it, whereas R^* is history-only and no amount of querying recovers what was not retained. The intervention is consistent with recoverability: adding back queryable state attenuates the extra tool calls, while adding back history-only state leaves them high here (Appendix A.3).

4.2.2 Selection null: which atoms are retained matters little at these budgets

Random selection matches the offline hindsight oracle (-22.1% vs -22.4%; paired difference $+0.10$, 95% CI $[-2.3, +2.3]$), while the recoverability-prioritized TypeAware does not improve on Random. The budget sets the scope of that null: at $B = 265$ the TypeAware digest already holds essentially all history-dependent state and most queryable state ($\text{cov}_R = 1.00$, $\text{cov}_D = 0.86$; Appendix A.5), leaving little to choose between — a ceiling on what selection can buy *at budgets loose*

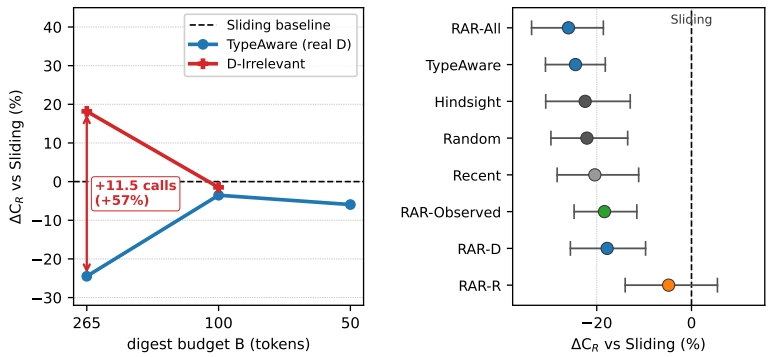

Figure 5: Retention interventions at a fixed compaction point (turn 10); ΔC_R vs. Sliding throughout. *Left* (content validity): across digest budgets $B \in \{265, 100, 50\}$ (decreasing left to right; D-Irrelevant at 265 and 100), only the load-bearing contrast is shown — TypeAware (real D) vs. D-Irrelevant (fabricated D); the gap is largest at $B = 265$ (Table 1) and closes as the budget shrinks. *Right* (selection null): the eight selection policies at $B = 265$ (RAR = recoverability-aware retention) with 95% bootstrap CIs; Random and Hindsight do not differ detectably, so at this budget fine-grained selection does not move cost. Coverage dose-response: Appendix A.5.

Table 1: D-Irrelevant control at $B = 265$: retrieval 20.4 $\rightarrow$ 31.9 ($+11.5$, $+57\%$, $p < .001$). Execution and completion show no detected change (Appendix A.5).

	real D	irrelevant D
C_R (retrieval)	20.4	31.9

enough to retain most atoms, not evidence that selection never matters: tighter budgets replicate the null. R-only is the structural exception, because R is compact enough to retain fully at any budget.

4.2.3 Content intervention: the D-Irrelevant control

The selection null leaves open whether the digest's *content* matters at all. The D-Irrelevant control holds the digest format, position, budget, and retained R content fixed and replaces the D atoms with fabricated out-of-universe state.

All 20 seeds show higher retrieval, and the effect is entirely in retrieval. D-Irrelevant also exceeds the no-digest Sliding baseline ($+18\%$, $p = .002$). The effect is budget-gated and replicates at the loose budget in the earlier generation ($+7.3$, $p = .006$; Appendix A.5).

4.3 Result 3: Boundaries and external probes

The retrieval signature is conditional on the compression operator. We probe that axis with matched $5\times$ operators, and the environment axis with Sliding in two further environments.

4.3.1 Operator contrast (Figure 6, left; Table 7, appendix)

The same $5\times$ budget can be spent by three operators — sliding (drop old turns), a fact-preserving extractive summary, or a fact-preserving abstractive summary (prompted to retain every observed state fact) — each compared against its own batch's Full condition so that between-batch level differences do not masquerade as an operator effect. Sliding raises retrieval by $+32.9$ calls versus -0.2 (extractive) and -3.7 (abstractive); only dropping state triggers the surge. The retrieval-cost response tracks whether execution-relevant state remains usable — reproducing ACON-style "compression need not hurt" only for operators that preserve state facts. Lossy summarizers that drop facts remain outside our scope. A prompt-ablation control leaves it unchanged (Appendix A.11).

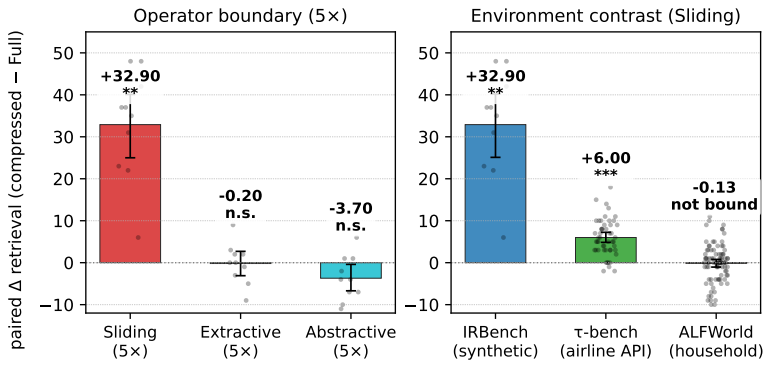

Figure 6: The retrieval signature is conditional on the compression operator. *Left:* matched $5\times$ operators on DeepSeek High, each against its own batch's Full — Sliding surges, extractive/abstractive do not. *Right:* Sliding across three environments: the surge appears in IRBench and reproduces in τ-bench; in ALFWorld the budget was never reached, so no context was dropped and that bar is uninformative. Environments use their own agents, so the right panel is not model-controlled. Stars denote a two-sided sign-flip permutation test on the paired deltas, Holm-corrected within each panel ($^{**}p < .01$, $^{***}p < .001$); n.s. denotes not significant after correction.

4.3.2 External environment probe (Figure 6, right; Table 8, appendix)

Neither environment exposes an IRBench-style task-graph query tool, so actions are classified by a deterministic rule (API identity for τ-bench; leading verb for ALFWorld) (Appendix A.11). ALFWorld yields no contrast: its operator never bound, so no context was dropped (Δ(retrieval-like) $= -0.13$); τ-bench reproduces the signature ($+6.00$, 95% CI $[+4.80, +7.20]$, 10/10 tasks positive, sign test $p = .002$), with repeated requests rising from 0.02 to 0.40 per run and execution-like actions showing no detected change (-0.12, $p = .36$; call-level decomposition in Appendix A.8). Reward falls from 56% to 46%; secondary, because inference is analysis-unit sensitive ($p = .13$ over 50 pairs; task level $p = .022$ when the six tied tasks are scored for Full, $p = .625$ without; Table 8). Sliding's magnitude varies across environments in the direction expected if reacquisition difficulty matters — consistent with, but not identifying, a recoverability-dependent account. One caveat: each environment is probed with its own harness and agent (DeepSeek for IRBench and ALFWorld, Sol for τ-bench), so environment and model are not separated; we read the τ-bench arm as external replication, not a model-controlled estimate, and the ALFWorld arm supplies none. We report that arm rather than omit it so the attempted cross-environment probe is on the record; its budget never bound, so it bears on the account neither way. A within-environment control shares that limit: re-issuing a call whose result has left the window is $11.7\times$ more frequent than re-issuing one still present (0.32 against 0.027), and across the visibility boundary, the rate rises from 0.3% at the window edge to 38.7% one turn past it (Appendix A.9).

5 Discussion

5.1 What completion misses

Our protocol decomposes compression's interaction cost into state reacquisition (retrieval) and task work (execution). The headline observation is that **task completion is an incomplete measure of this cost**: retrieval increases are significant in five of six cells under the pre-specified paired test after Holm correction, while no completion change is detected in any cell. Completion registers the cost only once reacquisition consumes enough of the horizon to limit task progress (DeepSeek under high compression). Used alone it misprices the cost both ways: understating it on robust models, which spend interaction budget to recover state while completion holds, and overstating it on fragile ones, which can lose a task without a corresponding surge in reacquisition calls.

5.2 What the reacquisition response tracks

The Sliding–Summary contrast separates retention from reacquisition: preserving observed state facts avoids the retrieval surge, while dropping them does not. This complements attention-centric accounts of compaction by separating the distinct cost of genuinely absent state.

The oracle intervention is consistent with the recoverability account. D is queryable, so when lost the agent can only recover it by re-querying the environment, and re-supplying it attenuates the retrieval burden. History-only restoration can reduce retrieval cost, but its effect is smaller and less stable (Appendix A.4). We call this pattern a re-query loop; we do not claim to observe the agent's internal decision process.

The retention interventions refine *where* a retention decision bites, forming a hierarchy: *state absence* (strong — dropping state inflates reacquisition), *valid content* (strong — irrelevant content raises cost 57%; see Limitation 4 for an alternative reading), *fine selection* (weak — bounded near zero at the budgets we test), and *effect visibility* (budget-gated). A retention decision is consequential at the level of validity and presence, not fine-grained ranking, in this regime.

5.3 Boundaries, runtime implication, and limitations

For a runtime designer, the actionable quantity is not "which compression ratio is safe" but which execution-relevant state, if dropped, would the agent spend its budget re-acquiring. Runtime should track that quantity rather than treat the compression ratio as the primary safety variable: in our setting, retaining compact history-dependent state and valid task-state content matters more than fine-grained ranking among valid D atoms.

The retrieval surge appears under dropping operators in both environments where the operator bound, and is larger where dropped state is harder to re-acquire — consistent with, but not identifying, a recoverability-dependent account (Appendix A.7).

6 Limitations

1. **Environment and model coverage.** The core mechanism is measured in IRBench (synthetic, two regimes), with a positive τ-bench replication and an ALFWorld arm whose operator never bound (hence uninformative about compression); we evaluate three model families (DeepSeek, Qwen, GPT-5.6 Sol, plus an earlier-generation GPT-5.5 replication). Specific cost magnitudes need not transfer to open-ended tasks; future work should extend the probe to partially-observable, long-horizon environments (e.g., WebArena (Zhou et al., 2023), SWE-bench (Jimenez et al., 2023)). The value is the *protocol* and the *decomposition*, which are environment-agnostic.
2. **Bounded horizon and cost proxy.** Real agents operate under a bounded interaction budget, so we evaluate under a 24-turn horizon; a metric insensitive to a large interaction-cost increase *within* that budget is a structural limitation. We measure the budgeted interaction currency — turns and tool calls — rather than provider-dependent wall-clock latency, billable tokens, or dollar cost. Because tokenization and accounting differ across providers, we do not treat token counts as a cross-model currency and do not convert tool calls into token or monetary cost. The relay endpoint used for the GPT arms also prepends a fixed prompt prefix to every request (Appendix A.11), so its token totals would not isolate context size.
3. **Operator scope.** The fact-preserving summaries are our designed controls, not generic summarizers: the abstractive variant is prompted to retain every state fact, so near-losslessness at $5\times$ is not evidence that arbitrary LLM summarizers (or token-level compressors, KV eviction) are lossless.
4. **D-Irrelevant fabrication.** The control replaces D atoms with fabricated out-of-universe entities; the added retrieval could reflect verification of inconsistent state content as well as semantic irrelevance per se. Both readings support the load-bearing claim, but the finer decomposition is left to future work.
5. **Checkpoint availability.** The main-table runs used an earlier DeepSeek checkpoint that is no longer served and therefore cannot be replayed exactly; the retrieval direction and magnitude are, however, reproduced on the currently served checkpoint (Appendix A.11).

Reproducibility Statement

We describe the experimental environments, compression operators, model configurations, evaluation protocol, statistical tests, and retention interventions in Sections 3–4. Appendix A.11 documents the environment verification checks, the pre-specified comparisons, the scale of each experiment, and the operator constants; the remaining appendices report per-condition and seed-level results and qualitative trajectories. The source code and experiment configurations are provided as supplementary material for anonymous review.

AI use statement

We used generative AI tools for the following tasks with required disclosure: providing feedback on research methodology and experimental design; designing and implementing the supplementary experiments reported in Appendix A.11 and Appendix A.2 (the prompt-ablation control and the Qwen-High robustness extension) together with the entity-level re-query diagnostic; assisting with translation; interpreting results and re-checking statistical calculations and tabulated results; and drafting the formal restatement of the visibility-boundary observable in Appendix A.9, whose observable, control analysis, and stated properties were derived from measurements the authors had already specified and run. We have not used generative AI tools to generate synthetic datasets, to propose or refine hypotheses, or to formulate mathematical claims or proofs beyond the formal restatement named above: the core research questions, the main measurement protocol, and the primary analysis plan are the authors' own, and the remaining tasks with required disclosure (cleaning or reformatting datasets, qualitative and 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 and other artifacts, drafting and editing parts of the manuscript for readability, and identifying relevant literature and formatting references. AI-assisted tools were also used to identify inconsistencies between the manuscript, source code, experimental logs, and reported results.

Generative AI tools were not used to collect any experimental data, primary or auxiliary: all runs reported here were executed by the authors. We have reviewed all AI-assisted work: every AI-assisted analysis was re-computed and verified by the authors against the underlying source code, experimental records, logs, and the archived per-run results, and every AI-suggested change to the manuscript was checked against the code and data before adoption. The authors made the final decisions regarding experimental design, statistical analyses, interpretations, and claims, and take full responsibility for the final content of this work, including all text, analyses, and artifacts produced with the aid of generative AI.

References

Igor Bogdanov, Chung-Horng Lung, Thomas Kunz, Jie Gao, Adrian Taylor, and Marzia Zaman. Context, reasoning, and hierarchy: A cost-performance study of compound LLM agent design in an adversarial POMDP, 2026. arXiv:2605.16205.

bolnet. context-clock: The open-source benchmark for agent token efficiency, 2026. GitHub repository, accessed 2026-09-03. `https://github.com/bolnet/context-clock`.

Factory Research. Evaluating context compression for AI agents, 2025. Technical report, Factory Research, 2025-12-16. `https://factory.ai/news/evaluating-compression`.

Run-Ze Fan, Zihao Zhang, Simin Ma, Yebowen Hu, Shouju Wang, Kaiqiang Song, Fei Liu, Hamed Zamani, and Xiaoyang Wang. An empirical study of harness design for coding agents, 2026. arXiv:2609.20804.

Olivier Gasté. Lost in compaction: Measuring information loss in LLM context summaries, 2026. Zenodo record 20273815, 2026-05-18. `https://zenodo.org/records/20273815`.

Qingnuan Han, Boli Fang, Mingzhi Hou, and Claire Liu. RideWay: Benchmarking efficient task completion for tool-using language agents, 2026. arXiv:2609.17985.

Dengzhe Hou, Lingyu Jiang, Dengjie Li, Zirui Li, Fangzhou Lin, and Kazunori D. Yamada. WMF-AM: Probing LLM working memory via depth-parameterized cumulative state tracking, 2026. arXiv:2603.27343.

Yinghan Hou and Zongyou Yang. Control under compression: Reliability frontiers for tool-using agents, 2026. arXiv:2608.01056.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models, 2023. EMNLP 2023. arXiv:2310.05736.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues?, 2023. ICLR 2024. arXiv:2310.06770.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2510.00615.

Aaditya Khanal, Yangyang Tao, and Junxiu Zhou. Beyond pass@1: A reliability science framework for long-horizon LLM agents, 2026. arXiv:2603.29231.

Vasilis Kontonis, Yuchen Zeng, Shivam Garg, Lingjiao Chen, Hao Tang, Ziyan Wang, Ahmed Awadallah, Eric Horvitz, John Langford, and Dimitris Papailiopoulos. MEMENTO: Teaching LLMs to manage their own context, 2026. Microsoft Research. arXiv:2604.09852.

Alex Kwon. Reclaim evaluation: A lossy memory is worse than an empty one, 2026. arXiv:2606.25449.

Jiayu Liu, Cheng Qian, Zhaochen Su, Qing Zong, Shijue Huang, Bingxiang He, and Yi R. Fung. CostBench: Evaluating multi-turn cost-optimal planning and adaptation in dynamic environments for LLM tool-use agents, 2025. arXiv:2511.02734.

Abhilasha Lodha, Mahsa Pahlavikhah Varnosfaderani, Abir Chakraborty, and Abhinav Mithal. Less context, better agents: Efficient context engineering for long-horizon tool-using LLM agents, 2026. Microsoft Research. arXiv:2606.10209.

Chang Ma, Junlei Zhang, Zhihao Zhu, and Cheng Yang. AgentBoard: An analytical evaluation board of multi-turn LLM agents, 2024. arXiv:2401.13178.

Guanghui Min, Liang Wu, Mayank Darbari, Chen Chen, and Liangjie Hong. Toward reliable context compression for long-horizon agents: An empirical study of execution instability, 2026. arXiv:2608.06503.

Tanmay Sah, Vishal Srivastava, Dolly Sah, and Kayden Jordan. The verifier tax: Horizon dependent safety success tradeoffs in tool using LLM agents, 2026. arXiv:2603.19328.

Natalia Trukhina and Vadim Vashkelis. Compress the context, keep the commitments: A formal framework for verifiable LLM context compression, 2026. arXiv:2605.17304.

Zhiqi Wang, Yichi Zhang, Dongwon Lee, and Yuchen Yang. Lost in compaction: Evaluating side-constraint loss under context compaction, 2026. arXiv:2608.11242.

Sarel Weinberger and Amir Hozez. Token reduction is not cost reduction, 2026. arXiv:2607.12161.

Binyan Xu, Haitao Li, and Kehuan Zhang. LLM agents are latent context managers: Eliciting self-managed context via state proprioception, 2026. arXiv:2606.30005.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of LLM-based agents, 2025. Findings of ACL 2026. arXiv:2503.16416.

Saber Zerhoudi, Jelena Mitrovic, and Michael Granitzer. The compaction cliff in long-running AI agent memory, 2026. arXiv:2608.22752; CIKM 2026.

Guangzhe Zhang. Correct now, insufficient later: Auditing update sufficiency in context compression, 2026. arXiv:2609.20045.

Kaituo Zhang, Zhen Xiong, Mingyu Zhong, Zhimeng Jiang, Zhouyuan Yuan, Zhecheng Li, and Ying Lin. Are tools all we need? unveiling the tool-use tax in LLM agents, 2026. arXiv:2605.00136.

Mingwei Zheng, David O'Brien, Siwei Cui, Pardis Pashakhanloo, Rajdeep Mukherjee, Myeongsoo Kim, and Sachit Kuhar. CORVUS: Context optimization and reduction via underlying synchronization for LLM coding agents, 2026. arXiv:2607.22711.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents, 2023. arXiv:2307.13854.

Jialiang Zhu, Gongrui Zhang, Xiaolong Ma, Lin Xu, et al. RE-TRAC: REcursive TRAjectory compression for deep search agents, 2026. arXiv:2602.02486.

A Appendices

A.1 Full severity sweep

Table 2 reports the sweep behind Figure 2: completion, total tools, the retrieval–execution split, and the mean turn count at each compression ratio. Turn counts stay between 22.8 and 23.6 in every condition — the horizon is effectively saturated throughout — while retrieval more than doubles.

Table 2: Severity sweep, DeepSeek High (companion to Figure 2).

ratio	completion	tools	retrieval	execute	turns
1× (Full)	83%	39.5	22.2	17.3	22.9
1.7×	87%	42.7	25.4	17.3	23.3
2.5×	81%	49.7	33.3	16.4	22.8
5×	72%	71.4	55.1	16.3	23.1
10×	66%	77.1	63.2	13.9	23.6

Low-IR separates the cost from the horizon. The same sweep under Low-IR, where far less state is hidden at runtime, behaves differently on outcomes and identically on cost: at $10\times$ retrieval rises from 20.7 to 46.0 calls ($+25.3$; 10 of 10 seeds; Wilcoxon $p = .004$) while turns move by $+0.2$ ($16.8 \rightarrow 17.0$) and completion moves in a single run ($100\% \rightarrow 97\%$). Cost therefore grows with compression in a regime where the horizon is not near its cap — which is the control for reading the High-IR completion loss as something other than turn exhaustion, since at $10\times$ under High-IR completion does fall ($83\% \rightarrow 66\%$; $p = .016$).

The completion boundary is empirical, not predicted. Completion does not degrade in step with the ratio: under High-IR it is not detected at $5\times$ and appears only at $10\times$, while under Low-IR the same $10\times$ ratio leaves it essentially intact. We report this as a property of our setting rather than as a rule. The sweep spans five ratios in one regime pair under one operator, which is not a design on which a predictive model of when completion degrades could be fitted and validated; we do not propose such a model and leave it to future work.

A.2 Cross-model results

Table 3 gives the six-cell matrix at $5\times$ behind Figure 3, and Figure 7 shows the retrieval–execution decomposition of each cell.

Post-hoc robustness extension on the weakest cell. Because Qwen High was the only cell whose pre-specified contrast did not reach significance, we collected an additional ten paired seeds for that cell alone. The point estimate was essentially unchanged: $+2.7$ calls in the added seeds and $+2.8$ pooled over twenty paired seeds, against $+2.9$ in the original ten. The added batch did not reach significance on its own (Wilcoxon $p = .33$; six of ten seeds positive), while the pooled twenty-seed contrast is significant under the pre-specified paired Wilcoxon test ($p = .034$) but not under the exact sign test ($14/20$, $p = .115$). Because the extension was undertaken after inspecting the original result, we do not fold it into the pre-specified primary-family inference: the family-level

Table 3: Three-model matrix at $5\times$ (companion to Figure 3), plus the earlier-generation replication. †: still significant after Holm–Bonferroni across the six-comparison retrieval family (the parenthetical p is uncorrected). The last two rows are the GPT-5.5 replication, which is *not* part of the primary family; it is a cross-generation check whose inference uses the six comparisons of its own generation — its two cells together with the DeepSeek and Qwen cells.

Model	regime	comp. @ $5\times$	$\Delta(p)$	tools	retr. $\Delta(p)$	exe
DeepSeek	High	83→72	$-11(.25)$	39→71	$+32.9(.002)^{\dagger}$	-1.0
Qwen	High	75→66	$-9(.13)$	34→37	$+2.9(.088)$	≈ 0
GPT-5.6 Sol	High	77→77	0 (1.00)	37→62	$+24.2(.002)^{\dagger}$	$+1.2$
DeepSeek	Low	100→100	0 (1.0)	33→55	$+22.6(.004)^{\dagger}$	$+0.1$
Qwen	Low	100→94	$-6(.25)$	28→36	$+5.3(.023)^{\dagger}$	$+2.7$
GPT-5.6 Sol	Low	100→97	$-3(.50)$	23→50	$+25.2(.002)^{\dagger}$	$+1.8$
GPT-5.5	High	80→85	$+5(1.0)$	39→82	$+42.9(.002)$	≈ 0
GPT-5.5	Low	100→100	0 (1.0)	26→49	$+22.4(.002)$	$+0.7$

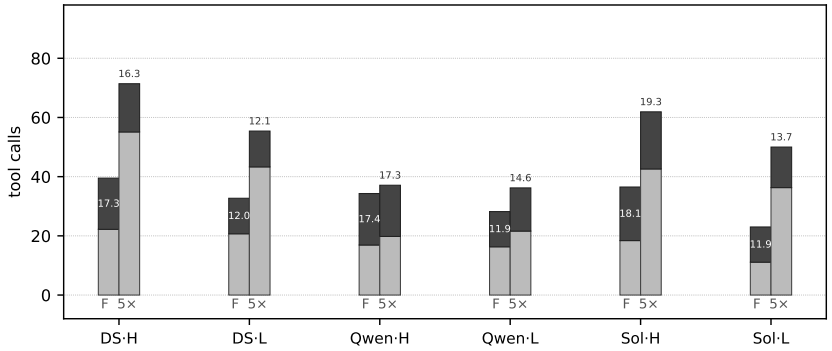

Figure 7: Tool decomposition per model–regime cell, Full vs. $5\times$ (companion to Figure 3). Retrieval (light) accounts for most of the added interaction; execution (dark) stays near-flat across all six conditions.

statement remains five of six, and the extension is reported only as a check on the stability of the point estimate. No seeds were excluded, re-weighted, or re-selected, and the extended data are archived alongside the original run.

A.3 Oracle intervention

These levels are batch-specific. The table below comes from a separate batch from the matrix of Table 3: its Full and Sliding completion levels are 86% and 66%, against 83% and 72% there. Every quantity in it is a within-batch contrast against this batch's own Full and Sliding arms; the levels are not comparable across batches, and we neither combine nor rank them.

Table 4 lists all five oracle conditions behind Figure 4, with the change in tool calls relative to Sliding. These runs were logged with a total tool count rather than the retrieval–execution split, so we report the oracle effect on tool calls, as plotted in Figure 4(a). The GPT-5.6 Sol replication in Section A.4 carries the split. Injection uses environment ground truth for the relevant recoverability class and is re-supplied at every model call from turn 1, so it is a superset of the state any single run observed and dropped, not a reconstruction of it. The split is available for the Sliding conditions, where the added interaction is uniformly retrieval (Table 2: $+32.9$ retrieval against -1.0 execution at $5\times$).

The sanity arm is not inert in this batch. Restoring R^* on top of Full already lowers the tool count from 38.8 to 32.1 (-6.70; 8 of 10 runs lower; Wilcoxon $p = .019$), with completion unchanged (86% against 85%). Part of the Sliding$+R^*$ effect is therefore a standing effect of supplying R^* at every call rather than a recovery of dropped state, and we do not attribute the whole contrast to

recovery. On the GPT-5.6 Sol batch the same arm does not show this ($14.2 \rightarrow 16.9$ retrieval calls; $p = .20$), so the caveat applies to the DeepSeek table above.

Table 4: Oracle intervention, $5\times$ High (DeepSeek; companion to Figure 4).

condition	completion	tools	Δ tools vs sliding
Full	86%	38.8	—
Sliding	66%	72.9	—
Sliding $+R^*$	78%	69.0	-3.9
Sliding $+D^*$	80%	35.8	-37.1
Full $+R^*$ (sanity)	85%	32.1	—

A.4 Oracle state-availability intervention on GPT-5.6 Sol

To probe whether the added retrieval cost under sliding compression is associated with unavailable task state, we reran the oracle intervention on GPT-5.6 Sol with the same five-arm design as the earlier DeepSeek study (Full, Sliding, Sliding$+R^*$, Sliding$+D^*$, and Full$+R^*$ as a retrieval-cost sanity condition; 10 paired seeds, 50 runs). Unlike the earlier runs, these carry the retrieval–execution split. All 50 rows passed a post-hoc integrity audit: unique run identifiers, $C_R + C_E =$ tools in every row, and no retained error rows. This arm was collected as a separate batch from the six-cell matrix of Table 3, so absolute levels are not comparable across the two: at the same nominal setting ($5\times$, High) Full completion is 89% here against 77% there, and retrieval is 14.2 against 18.4 at Full and 54.7 against 42.6 at $5\times$. Every claim below is a within-batch paired contrast.

Sliding raised retrieval calls from 14.2 ± 4.6 to 54.7 ± 9.0 ($+40.5$; 10/10 paired increases; Wilcoxon and sign test $p = .002$), while completion fell from 89% to 77%. Making the task graph directly queryable via D^* reduced retrieval to 28.7 ± 13.7 (-26.0 vs. Sliding; 10/10 paired decreases; $p = .002$) — 64.2% of the Full-to-Sliding retrieval gap, but still 14.5 calls above Full. D^* therefore substantially attenuates rather than eliminates the burden, and unevenly: only 3 of 10 D^* runs reached the low-retrieval floor of the Full condition ($C_R = 10$), and task-level outcome did not improve (76% vs. 77% for Sliding, paired $p = 1.0$). Restoring history-only state via R^* produced a smaller reduction, to 42.7 ± 11.6 (-12.0; Wilcoxon $p = .014$, sign test $p = .109$); the effect is concentrated in a few seeds and in one seed shard, and none of the three shards is individually significant, so we do not read it as a restoration mechanism. Neither intervention returned completion to Full (81% for R^*, 76% for D^*). The Full$+R^*$ arm showed no significant aggregate retrieval change ($+2.7$, $p = .195$), but it lowered completion by 13pp and exhausted the turn budget more often (6/10 vs. 3/10), so we report it as a retrieval-cost check rather than an invariant control.

Two design constraints bound the interpretation. First, D^* supplies exactly the content returned by the three retrieval tools that define C_R, so part of its effect is definitional rather than evidential; the informative quantity is the unchanged task-level outcome. Second, the injected D^* string is $\approx 2.7\times$ longer than the R^* string (mean 667 vs. 244 characters) and this batch contains no length-matched placebo or Full$+D^*$ arm, so prompt volume is not separated from state content. Relative to the earlier DeepSeek oracle (Table 4), the direction replicates — both injections reduce retrieval, D^* more — while the relative size of the R^* effect is larger here ($11.7/25.0$ vs. $3.9/37.1$ of the D^* effect on total tool calls); this between-generation difference is not statistically distinguishable (Mann–Whitney $p = .384$). Taken with the earlier runs, these results support an association between state availability and retrieval burden under compression, but they do not isolate state content from prompt length or tool-specific redundancy.

A.5 Retention policies, coverage, and content validity

Selection matrix. ΔC_R vs. the Sliding baseline at $B = 265$ (20 paired seeds). The budget gate is resolved only to a few calls: at $B = 100$ the paired interval for TypeAware vs. D-Irrelevant is $[-2.3, +3.4]$ calls, so the absence of a content effect at that budget is *bounded* rather than established:

Retained coverage. At $B = 265$, the recoverability-prioritized TypeAware digest retains essentially all R-type state ($\mathrm{cov}_R = 1.00$, $\mathrm{cov}_D = 0.86$), while the unranked RAR-All digest retains all D atoms

Table 5: Selection policy matrix at $B = 265$ (DeepSeek High, 20 paired seeds); RAR denotes recoverability-aware retention.

Policy	ΔC_R vs. Sliding
RAR-All (unranked, observation order)	-26.0%
TypeAware (R-first)	-24.5%
Hindsight (offline oracle)	-22.4%
Random (seeded)	-22.1%
Recent (most-recent atoms)	-20.4%
RAR-Observed (online re-access proxy)	-18.4%
RAR-D (D-only)	-17.8%
RAR-R (R-only)	-4.8%

but collapses R coverage ($\text{cov}_R = 0.30$) — a coverage advantage that does not translate into lower cost (Table 5) or into a higher completion rate. The converse holds too: RAR-Observed's coverage profile ($\text{cov}_D = 0.99$, $\text{cov}_R = 0.27$) is nearly identical to Hindsight's, yet it lands 4 points away (-18.4% vs. -22.4%), and D-Irrelevant ($\text{cov}_D = 0$) *raises* retrieval by $+18.2\%$. What separates these digests is whether the D content they retain is valid, not how much of it they carry.

Tighter budgets. The Sliding baseline itself is measured only at $B = 265$, so the contrasts below are within-budget. The selection null is not an artifact of the loosest budget. Hindsight sits $+0.60$ retrieval calls above Random at $B = 100$ (paired Wilcoxon $p = .76$) and -2.75 calls below it at $B = 50$ ($p = .098$), so the contrast stays null as the digest tightens. The $B = 50$ point is weak in a second sense: the Random arm's cov_D has already fallen to 0.14 (range 0.08–0.15 across both arms, against 0.31 at $B = 100$ and 0.89 at $B = 265$) as its digest shrinks from 221 to 85 to 45 tokens, so the digest no longer holds the D atoms that the ranking is supposed to allocate. $B = 100$, where the null still holds and coverage is not yet at the floor, is the informative tighter point.

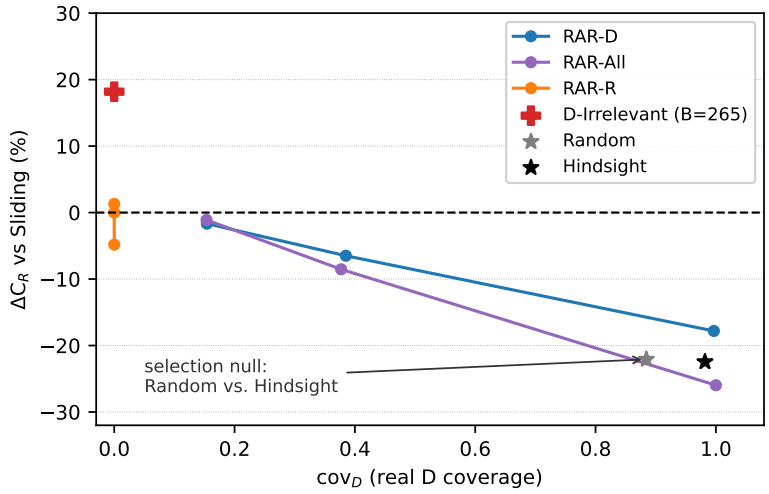

Figure 8: Coverage dose-response (companion to Figure 5). ΔC_R vs. Sliding as a function of real-D coverage cov_D across digest budgets for RAR-D / RAR-All / RAR-R. D-Irrelevant at $B = 265$ sits at $\text{cov}_D = 0$ yet *raises* retrieval ($+18\%$) — content, not coverage, is load-bearing. Random and Hindsight do not differ detectably at $B = 265$ (selection null).

D-Irrelevant seed-level effect. Per-seed paired retrieval increase (D-Irrelevant minus TypeAware, $B = 265$): all 20 seeds positive, range $+3$ to $+23$, mean $+11.5$. Alongside it, neither execution calls nor completion moves detectably in this control (17.3 → 16.4, $p = .086$; $75\% \rightarrow 81\%$, $p = .41$).

Cost-aware dominance. We say m_1 dominates m_2 if $Q(m_1) \geq Q(m_2)$ and $C(m_1) \leq C(m_2)$ with at least one inequality strict. The cleanest instance is within a single batch and seed set: the TypeAware retention digest at $B = 265$ reaches higher completion than Sliding (75.0% vs. 74.5%) at lower tool cost (37.6 vs. 44.5, 20 paired seeds), a gap completion-only evaluation cannot quantify.

Table 6: D-Irrelevant seed-level effect, $B = 265$ (ΔC_R, per seed).

seed	0	1	2	3	4	5	6	7	8	9
ΔC_R	$+16$	$+10$	$+11$	$+9$	$+22$	$+4$	$+9$	$+7$	$+23$	$+8$
seed	10	11	12	13	14	15	16	17	18	19
ΔC_R	$+13$	$+15$	$+12$	$+3$	$+9$	$+11$	$+17$	$+8$	$+5$	$+18$

The completion half of this pair is itself far below what this design resolves; we read the instance as a cost improvement at no detected completion cost, which is the asymmetry the dominance relation is meant to expose.

Earlier-generation (GPT-5.5) replication. At $B = 265$, real-D digest reduces retrieval vs. sliding (-9.5, $p = 0.002$) and D-Irrelevant raises it back ($+7.3$, $p = 0.006$), while no completion change is detected (78% vs. 74%, $p = 0.125$). At $B = 100$ no content effect is detectable, and the poisoning intensity (irrelevant $>$ sliding) seen in DeepSeek does not reproduce.

A.6 Operator contrast

Table 7 gives the three matched $5\times$ operators behind the left panel of Figure 6, each differenced against its own batch's Full condition. Only the dropping operator shows a large retrieval effect.

The operators are dose-matched on context. Mean context per model call falls by about half in every batch: $3711 \to 1842$ tokens for Sliding, $3683 \to 1815$ for Extractive, and $3863 \to 1979$ for Abstractive. The contrast between them is therefore in what they preserve, not in how much they remove.

Table 7: Operator contrast at $5\times$, DeepSeek High (companion to Figure 6, left panel). Three separate 10-seed batches; each delta is against the paired Full from that operator's own batch.

Operator	Δ comp. (pp)	ΔC_R	ΔC_E	Interpretation
Sliding	-11	$+32.9$	-1.0	retrieval surge
Extractive	$+9$	-0.2	$+0.4$	near-lossless
Abstractive	$+3$	-3.7	$+1.5$	near-lossless

A.7 Environment probes

Table 8 summarizes the recoverability geometry of the three environments behind the right panel of Figure 6, together with each environment's probe setup.

The ALFWorld probe is not a resolved null. The operator here is a fixed $4{,}000$-token context budget, and the probe's mean context per model call is 1157 tokens in the heaviest run against a median of 908 across runs. The archived trajectories carry per-run totals rather than per-call sizes, so the peak context cannot be bounded from them. What the archive does show is that the budget was never observed to bind and no turn was observed to be dropped, and that the two conditions differ by only $+0.20$ turns and $+208$ tokens per run (both $p \geq .75$). The $n = 90$ paired comparison and its 95% interval $[-1.06, +0.81]$ calls per run — which would exclude an increase larger than 0.8 calls, about $1/40$ of the IRBench High effect — describe a comparison between two conditions that dropped no context, and so bound compression damage no more than any null would. This arm is therefore uninformative about compression rather than evidence that re-observable state is costless, and we no longer read it as a boundary. The mechanism it was meant to test — that state which can be re-observed directly in the environment does not need to be carried — remains untested here.

τ-bench uses the official harness with a user simulator (GPT-5.6 Sol agent; DeepSeek user simulator), 10 tasks $\times$ 5 repeats $\times$ Full/Sliding; ALFWorld samples 18 tasks across all six task types, 5 paired repeats each (180 runs; 90 pairs).

Cross-generation replication. The τ-bench probe also ran on the earlier generation (GPT-5.5) under the same protocol and harness, and the retrieval signature reproduces: $\Delta C_R = +4.28$ calls

Table 8: Recoverability geometry across environments (companion to Figure 6, right panel). The τ-bench reward difference is reported as secondary: over the 50 repetition-level pairs it is not significant (39 ties; Wilcoxon $p = .13$), and at the task level one task favors Sliding against three for Full with six tied, giving $p = .022$ when the tied tasks are scored for Full and $p = .625$ if they are dropped. Inference therefore depends on the analysis unit. The ALFWorld row compares two conditions that dropped no context — its operator never bound (Section A.7) — so it is uninformative rather than a null about compression.

Environment	state recoverability	ΔC_R	Outcome change
IRBench	D queryable, R history-only	$+32.9$	ΔQ: model/regime dep.
τ-bench	persistent DB + user	$+6.00$	Δreward $= -10$pp (n.s.; unit-sensitive)
ALFWorld	re-observable in-world	-0.13	uninformative (probe never bound)

per run (95% CI $[+2.44, +6.02]$; 10 of 10 tasks positive; paired Wilcoxon $p < .0001$), execution calls flat (-0.56), repeated requests rising ($+0.34$ per run), and reward falling numerically ($44\% \rightarrow 32\%$; $p = .11$). The Sol effect is numerically larger ($+6.00$) and the two intervals overlap; since the batches are separate, we read this as the same direction on a second generation rather than as an effect-size comparison. No token counts are reported for either batch, for the reason given in Section A.11.

A.8 The τ-bench retrieval channel at the call level

The $+6.00$ retrieval increase of Table 8 is an aggregate over calls. We decompose it and report the qualifications it hides. Throughout, an *atom* is a $(\text{tool}, \text{entity})$ pair given by the call target and restricted to the five read tools that count toward C_R; the opportunity pool at a step is the set of atoms already observed earlier in that run, so first calls are excluded by construction.

Two read tools carry the increase. `get_user_details` rises from 50 to 191 calls and `get_reservation_details` from 130 to 257; the five read tools together go from 309 to 609, while the write and transfer tools go from 65 to 59. Sliding spends its extra calls re-reading records it has already seen.

tool	Full	Sliding	Δ
`get_user_details`	50	191	$+141$
`get_reservation_details`	130	257	$+127$
`search_direct_flight`	105	120	$+15$
`search_onestop_flight`	14	24	$+10$
`list_all_airports`	10	17	$+7$
C_R subtotal	309	609	$+300$
write and transfer tools	65	59	-6
total	374	668	$+294$

What is missing is the payload, not the identifier. 329 of the 333 re-queries (98.8%) repeat the previous call for the same atom byte-for-byte, and in 314 (94.3%) the identifier itself is still present in the retained turns. The agent has lost the returned record, not the key to it.

Two evidence lines, reported separately. The similarity criterion of Section A.11 flags 20 repeated requests across the 50 sliding runs, whereas atom-level re-query counts 333: a factor of 16.6. The text criterion records that a request was paraphrased, not that state was re-acquired, and the two counts are neither pooled nor combined.

Visibility here is reconstructed. These trajectories record only the retained and total message counts per step. Replaying the operator reproduces the recorded retained count at all 1076 steps that carry a compression record (976 of which drop at least one turn) with no mismatch, which fixes *how many* turns were retained; *which* turns is inferred from the operator rather than read off a logged context list, so these counts carry a weaker guarantee than the IRBench entity-level check of Section A.11, where visibility is read off the context the compressor returns.

Full is a constructive control. Full never drops a message, so its absent pool is empty by construction rather than by measurement; it is not evidence that the absent rate would be low without compression. The falsifiable Full quantity is its redundant re-read rate, 41 of 950 opportunities (4.3%), against 2.7% for re-reads of still-retained results under Sliding.

Horizon endogeneity. Nine of the 50 sliding runs exhaust the 30-step cap and no full run does. Those nine contribute 71 of the 333 re-queries (21%) and their own rate is lower (0.20) than that of the uncapped runs (0.38), so the contrast is not an artifact of runs that run out of horizon.

The agent side of this arm never emits a reasoning field by construction, so none of these counts can be conditioned on extended thinking; the same caveat applies to the aggregate τ-bench numbers.

A.9 The visibility-boundary reacquisition response

Definition. We restate the observable here so that the profile is self-contained. Fix a run under a compression operator and keep only the calls to tools that count toward C_R. Group them by *atom*: the pair formed by the tool and an entity read off the call arguments, namely the value of the first of `user_id`, `reservation_id`, `payment_id`, origin, airport, destination and `flight_number` that is present and non-empty, stripped and lower-cased, rendered as $key = value$ for origin, airport and destination. A call carrying none of the seven forms the atom $(tool, \text{empty})$, so all such calls in a run share one atom. The *opportunity pool* at a step is the set of atoms already observed earlier in that run, so an atom's first observation never enters the pool. If an atom's most recent observation lies a turns back and the operator holds $r = \max(1, \mathrm{round}(0.2\,k))$ turns at step k — the τ-bench rule, which floors the retained window at a single turn pair rather than at the two turns of the in-house operator of Section A.11 — its *visibility lag* is $\tau = a - r$, positive exactly when the atom has left the retained context. Write $H(\tau)$ for the fraction of lag-τ pool opportunities whose atom is the target of the current call, and h_{vis} for the same fraction over atoms with $\tau \leq 0$. Everything below is computed on the GPT-5.6 Sol batch; the profile is a property of that setting, and we do not claim that every operator reproduces it.

Re-query is conditioned on invisibility. Under Sliding, 333 of 1047 opportunities to re-issue a call whose result has left the window are taken (0.32), against 24 of 883 when the result is still retained (0.027) — an $11.7\times$ contrast. The direction holds in 5 of 5 repetitions and 10 of 10 tasks, and 47 of the 50 sliding runs contribute at least one such call. The two arms differ in how concentrated they are: on the absence side the largest single run contributes 5.4% of the 333 calls, whereas the 24 visible-side calls are not spread at all — 19 of them come from a single task, and all but two of the 9 calls that fall two turns inside the edge are `search_direct_flight` searches, six of them for the same origin, a search the agent repeats while re-checking flight options. Removing the two tasks that contribute them leaves 265 of 695 absence-side opportunities taken against 0 of 604 visible-side ones; removing either alone raises the contrast to $51.2\times$ and $13.4\times$. The reported $11.7\times$ is therefore conservative relative to these leave-one-out comparisons.

The response is gated by the visibility boundary. We bin each opportunity by the lag to the retention edge, age $-$ retained turns. Reacquisition is sharply gated by the visibility boundary rather than explained by information age alone. After an atom leaves the visible context, reacquisition probability spikes for the next one to two turns and then decays over subsequent turns without reaching the Full rate: 0.3% while the result is still retained at the edge ($1/341$), 38.7% one turn after it drops out ($132/341$), 44.3% two turns out ($108/244$), and then 33.1%, 22.8%, 16.9%, 13.2% and 8.3% at three, four, five, six to eight, and nine to twelve turns past the edge. This is an empirical response profile, not a fitted hazard law. Because the retained-turn count grows with the step index, this axis holds neither the step nor the raw age fixed. The decay is not carried by a few trajectories: all ten tasks and all five seeds contribute at the first three lags, and 43 of the 50 sliding runs contribute at the first lag.

age − retained turns	pool	re-queries	rate
≤ -3	127	13	10.2%
-2	169	9	5.3%
-1	246	1	0.4%
0 (retained, at the edge)	341	1	0.3%
$+1$ (just dropped)	341	132	38.7%
$+2$	244	108	44.3%
$+3$	127	42	33.1%
$+4$	92	21	22.8%
$+5$	65	11	16.9%
$+6$ to $+8$	106	14	13.2%
$+9$ to $+12$	60	5	8.3%

The last row is a closed band: twelve further opportunities with lag beyond $+12$ are omitted (none of them taken), so the absence column sums to 1035 of the 1047 opportunities quoted above. The age-binned table below partitions all 1047.

Inside the edge the rate is not uniform: it is lowest in the two cells adjacent to it (0.4%, 0.3%) and highest at the freshest lag (≤ -3: $13/127 = 10.2\%$). These inside-edge rates are not the baseline the crossing is measured against: every one of the 13 calls in the ≤ -3 cell, and 7 of the 9 two turns inside, is a `search_direct_flight` re-search concentrated in the same two tasks that supply the visible-side calls above, and eleven of those thirteen share one origin, so the visible side is carried by repeated same-origin searches rather than by reacquisition of dropped state.

Information age alone does not account for the response. If re-query were driven by how stale a state is, its rate should rise with the age of the last observation whether or not the state was ever dropped. In the uncompressed Full arm, where no observation is absent by construction, the redundant re-read rate stays low at every age (1.3%, 8.0%, 3.1%, 7.0% and 1.4% over the five bands below), and the same holds under Sliding while the record is still inside the window: 3.3% and 1.8% at ages two and three to four, against 52.4% and 44.5% in the arm that has crossed the boundary over the same bands.

Age-binned control.

age of last observation	Full (controls)	Sliding (still visible)	Sliding (crossed)
1	3/229	11/384	—
2	15/187	8/244	43/82
3 to 4	8/259	4/223	149/335
5 to 8	14/201	1/32	119/407
9 and beyond	1/74	—	22/223

This rules out a simple age-only account and supports a boundary-gated interpretation; it is not a demonstration that the boundary is the only causal variable, since the age and lag axes are not independent and the control cells rest on few contributing runs. We read the profile as boundary-conditioned: reacquisition is far less frequent while a state remains visible than once it leaves the window, and on the visible side the rate stays at a few percent across ages; it spikes as soon as the state leaves the window, and then decays without returning to the Full rate within the range observed (it is still 13.2% at six to eight turns and 8.3% at nine to twelve, against 4.3% for Full). What a compression operator controls is how much state it converts into this transient retrieval demand, not how old the state it keeps is.

What the profile does and does not separate. An atom that is absent at a step is, by construction, also an atom that has not been re-read since it left the window, so the profile's denominator is the same whether its bins are indexed by lag or by elapsed duration. Re-binning the 333 re-queries over that duration reproduces the same decline — 42.4% over the first two turns (255 of 602 opportunities), then 25.9%, 16.1% and 7.2% over turns three to four, five to eight, and nine or more, with a median of two turns — which is a consistency check on the tabulation rather than a test of mechanism. A genuine temporal decay and an unobserved spread across atoms in how likely they are to be needed both yield a falling aggregate rate on this data, and we do not separate them here. The shape above is reported as a description of the measured response.

Empirical property: boundary gating. Over the range observed, the profile empirically exhibits three features. It stays far below the crossed side while the atom remains visible ($h_{\text{vis}} = 2.7\%$ overall, and 0.3% in the cell at the edge itself). It jumps on crossing, to 38.7% one turn past the edge — an order of magnitude above h_{vis}, and $3.8\times$ the largest rate seen inside the edge. And it decays monotonically beyond its peak ($h(+2) > h(+3) > h(+4) > h(+5) > h(+6\text{–}8) > h(+9\text{–}12)$). We state these as properties of this measured profile, not as laws that an operator must satisfy.

A.10 Qualitative trajectories

Case 1: τ-bench repeated-identifier (task 40, repeat 4, sliding). The user supplies their identifier at step 0; sliding compression truncates the earliest turns, so the message carrying it has dropped out of the retained window (six of the twenty-four messages sent so far at the first re-ask below, eight of thirty-four at the second) and the agent asks for it again:

Turn	Conversation (abridged)
0	**U:** My user ID is `sophia_silva_7557` ... I don't have the reservation ID.
11	**A:** Please provide your **user ID** and the reservation ID for the canceled flight.
11	**U:** My user ID is `sophia_silva_7557`, and I'm a Gold member ...
16	**A:** Please provide your **user ID** and the reservation ID for the canceled flight.
16	**U:** I don't have the reservation ID, but my user ID is `sophia_silva_7557`.
22	**A:** Please provide your **user ID** and the reservation ID for the canceled flight.

The user re-supplies the identifier each time it is requested, and the agent's own `get_user_details` calls at steps 12 and 17 return the profile successfully; the re-asks nonetheless continue, because the earliest turns — among them the ones carrying the identifier — are exactly what sliding compression drops. Reacquisition thus consumes turns and the run exhausts the 30-step cap without completing the task (reward $= 0$). From step 11 onward the agent is in fact asking for the *reservation* identifier as well, which the user never provides. The full condition on the same task and repeat completes without this loop.

Case 2: IRBench history-dependent (R^*) loss. Under sliding compression, a previously revealed failure condition or fail-until count (recoverable only from execution history, not from any public query) is dropped; the agent re-queries the task graph without recovering it, and the associated task is abandoned or exhausts turns — the behavior the oracle arm probes by re-injecting R^* (Table 4).

Case 3: Abstractive vs. sliding. The abstractive operator represents the same observed state facts in a digest, so the agent does not need to re-acquire them; the sliding operator deletes them, producing the retrieval surge in Figure 6 (left panel).

A.11 Setup and reproducibility

Environment verification. All IRBench numbers come from a single deterministic environment in which a seed fully determines the world instance, so every condition is replayed on an identical task set. Each instance holds ten tasks over three capacity-one resources; the High-IR regime reveals three failure conditions, four hidden ordering constraints, and two resource holds through execution, while Low-IR reveals one of each and no holds. Before collecting results we ran six self-checks on the environment: (i) *determinism* — the same seed generates an identical world three times; (ii) *replay* — a fixed eight-step tool sequence reproduces an identical snapshot; (iii) *class separation* — the three retrieval tools never leak history-dependent state; (iv) *mechanism* — a fully informed executor completes every instance within the derived tool budget; (v) *solvability* — 100% of instances are solvable across 60 seeds $\times$ 2 regimes; and (vi) *regime contrast* — a memory-limited policy separates the two regimes as intended.

Pre-specification. The primary comparison (Full vs. Sliding at $5\times$), its paired test, and the six-comparison retrieval family for Holm correction were fixed before the severity sweep was run; the

sweep's per-cell outcomes did not determine them. The $5\times$ point is therefore a pre-specified operating point and not a best-case selection.

Scales. The main-text IRBench experiments use ten paired seeds (ten task instances per seed); the retention interventions use twenty; ALFWorld and τ-bench use five repeats each. Each run is a single stochastic trajectory (no fixed API seed); conditions are paired by task instance, not by a shared random realization. The severity sweep is five ratios $\times$ ten seeds per regime (50 runs per regime, 100 total). Runs are resumable: each is appended atomically to a CSV, and a re-run skips seeds that already completed.

Compression operators. Context is rebuilt before every model call. Token counts are estimated as characters$/3.5$ plus four tokens per message, with tool-call arguments included, so budgets are computed without any provider tokenizer. At ratio r the sliding operator targets $1/r$ of the current full history and keeps the most recent whole turns that fit, retaining at least two turns and always the system prompt and initial user message. The digest for the extractive summary is produced deterministically from the structured tool observations — facts are parsed by rule, with no model call — so it is reproducible from the call log; only the abstractive variant uses a model, and the tokens it consumes on the compression side are not counted in the agent's trajectory metrics, which measure interaction cost rather than total memory-management cost.

Agent system prompt. The agent receives a fixed prompt that lists all task and resource identifiers, instructs it to query every task's metadata first, to keep track of every constraint it observes, and to avoid retrying a blocked task. The prompt is retained verbatim in every context condition, including Sliding, whereas the observations it produces are dropped. Because the prompt is always present while its earlier outputs are not, an agent whose history has been truncated re-executes the instruction as a fresh sweep: in nine of the ten Sliding runs a single later turn re-reads at least eight task records, and in six of them all ten, whereas no Full run ever repeats a metadata read; of the 27.5 re-acquired read calls of Table 9, 76% are `get_task_info` metadata reads. The measured read-call increase in Sliding is therefore shaped in part by this prompt–operator interaction.

Prompt-ablation control. We tested whether those instructions drive the surge, by removing both explicit carriers — the strategy step that instructs an exhaustive metadata scan, and the initial message's request to inspect the tasks — and rerunning the Full–Sliding contrast on ten paired seeds with four arms. The contrast persisted: $\Delta C_R = +26.0$ calls under ablation against $+27.0$ under the original prompt, with all ten seeds positive under each (exact sign test $p = .002$ both times); the paired interaction is -1.0 calls (95% CI $[-12.2, +10.2]$). The ablation did not depress the baseline: the Full arm completes 83% of its runs under the ablated prompt against 76% under the original, and the Sliding arm 79% against 80% (ten seeds each), so the persisting contrast is not an artifact of a weakened condition. The exhaustive character of the sweep is unchanged: in all four arms the agent issues all ten metadata reads within a single turn, and the repeated-sweep rate after compression is 1.0 per run under ablation against 0.9 under the original prompt. The full-sweep behavior therefore persists after removing the explicit sweep instruction, showing that the retrieval surge is not solely induced by that prompt instruction; it is consistent with context loss driving the re-acquisition, since no Full run repeats a metadata read. The `get_task_info` share of re-acquired reads is 70% here against 76% in the supplementary diagnostic; those are two separate runs on the same checkpoint and we do not pool them. The remaining magnitude is still specific to this task and prompt setup, and should not be generalized unchanged across prompting strategies.

These runs were collected on the currently served DeepSeek checkpoint, so the original-prompt arm above is also a directional reproduction of the main-table contrast on that checkpoint ($+27.0$ against $+32.9$ on the earlier one): same direction, all ten seeds positive, about 18% smaller in magnitude. We report it as a current-checkpoint robustness check rather than as an independent replication of the primary cell, and the batch's completion levels are not comparable with the main tables.

Entity-level re-query check. This paragraph reports a supplementary run that instruments the entity level (Table 9). It reuses the High-IR regime, Sliding fraction 0.2 ($5\times$), the 24-turn horizon and the same ten seeds, and adds a Full control on those seeds (20 runs). We key an *atom* by (tool, entity) as given by the *call target* rather than by the returned result, so that an atom is fixed by what was asked for; visibility is read off the context list that the compressor actually returns for that turn, not inferred from a window assumption. Each read call is *first* (the atom was never observed earlier in the run), *re-acquired* (observed earlier and absent from the context sent that turn), or *still visible*

(observed earlier and still present); calls repeating an atom already called in the *same* turn, whose result the model cannot have seen, are excluded from the Reacquisition Share, and none occurred here. Its criterion — α_R = re-acquired / (re-acquired + first) ≥ 0.5 — was fixed before the run.

It is met at 0.584 (27.5 re-acquisitions against 19.6 first reads per run; per-seed shares range from .30 to .71, mean .56). Both controls behave as required. Under Full the count is 0.0 in all ten seeds, and the metadata read is never repeated at all: every Full run issues exactly ten `get_task_info` calls and no more, although task status changes just as much in that arm. Recomputing every re-read against what that atom last showed, by deterministic replay of the archived call sequence, 72% of the 210 re-acquired metadata reads returned a `completed` flag unchanged since the previous read of that task; across all 275 re-acquired reads, 15% are `query_resource` calls, whose answer is static over a run and carries no status field at all. Of the 30.2-call increase in read calls over Full, 27.5 (91%) are re-acquisitions; execute calls do not increase (they fall, from 22.9 to 19.3), so the inflation is confined to the read channel. Re-acquisitions begin as early as turn 5 and span turns 5–24 (median 12), and no counted re-acquisition has a smaller gap than the three turns that the operator's two-turn retention floor permits — consistent with the window contents being measured rather than assumed.

Four limits. (i) The share is not compression-neutral: it is depressed by the mandatory first reads (19.6 per run), and it tracks termination, reaching 0.500 in the four Sliding runs that complete all tasks and 0.615 in the six that exhaust the horizon. (ii) In 16–31% of the counted re-acquisitions — depending on whether one, two, or three preceding turns are scanned, as recomputed by `requery_overlap_count.py` — the same entity had also been read by a *different* read tool within the immediately preceding turns, so an observation of that entity survived in the window even though the atom's own observation did not. The count therefore measures absence of that atom's own observation, not of every fact about it. (iii) A re-read performed precisely because the agent no longer knows whether a value changed is by construction indistinguishable from a content read, so a status-checking account cannot be excluded outright; it is bounded, however, by the Full arm, where such reads are rare (0.8 per run, and never a repeated metadata read). (iv) These runs use a later DeepSeek checkpoint (`deepseek-flash`) than the one behind every main-table number; that earlier checkpoint is no longer served, so the main experiments cannot be re-run. Completion in this run is therefore not comparable with the main tables in any direction (Full $8.2/10$, Sliding $8.8/10$ here) and is reported only to show that the read-call composition above is measured on runs of comparable length. The check is indicative evidence about the mechanism, not a replication of a main-table cell.

Table 9: Entity-level read-call composition, Full vs. Sliding $5\times$ (supplementary run on a later DeepSeek checkpoint). An *atom* is a (tool, entity) pair given by the call target. *re-acquired* counts read calls whose atom had been observed earlier in the run and was absent from the context sent to the model that turn; *still visible* counts reads of atoms that were still present. A *whole sweep* is a turn issuing at least eight `get_task_info` reads, including the initial sweep every run performs at its first model call; the Reacquisition Share pools *re-acquired* against *re-acquired* + *first*, so its denominator is smaller than the read-call column. Indicative only: the earlier checkpoint behind the main tables is no longer served, so those runs cannot be repeated, and these numbers are not merged with them.

condition	read calls	first	re-acquired	still visible	whole sweeps
Full	17.3	16.5	0.0	0.8	1.0
Sliding $5\times$	47.5	19.6	27.5	0.4	2.2

Turn and tool accounting. A turn is one model call. Every tool call returned in that call is executed and counted individually, so C_R and C_E are per-call counts, and the 24-turn horizon bounds model calls only: a run that exhausts the horizon has still been free to issue many more tool calls than turns.

Models. DeepSeek (deepseek-v4-flash), Qwen (qwen3.7-plus), and GPT-5.6 Sol were each accessed through an OpenAI-compatible endpoint at temperature 0.3 with a 4096-token response cap and no streaming; extended-thinking modes were disabled wherever the API exposes a switch, since a preliminary check showed the thinking variant performing worse across the board. The GPT arms are served by a relay endpoint; we verified at batch start and end that the served model identifier equals the requested one, and distributed runs across three relay credentials (the retrieval effect

is directionally consistent across all three; per-endpoint attribution is not identifiable because task assignment is nested within endpoint). We do not compare token counts across models: tokenization and accounting differ across providers, and the relay additionally prepends a fixed prompt prefix to every request, so those counts do not measure context size. The earlier-generation GPT-5.5 runs (Appendix A.2) were collected on the same relay before that model was withdrawn from it.

τ-bench. We use the official harness unmodified apart from context assembly, in the `airline` domain on the `test` split, with ten fixed task indices reused across five repetitions (100 runs; 50 pairs nested within the ten tasks, which are the independent units used by the reported sign test). The agent is GPT-5.6 Sol and the user simulator is DeepSeek, both served as above. Each step executes only the first tool call returned, and the episode ends at 30 steps or when the environment signals completion. Reward is computed by the harness from the final database state hash, not by the agent's own report. The airline API's read tools — `get_reservation_details`, `get_user_details`, `list_all_airports`, `search_direct_flight`, `search_onestop_flight` — count toward C_R; its write tools toward C_E; and `think` and `calculate` toward neither. We count a request as repeated when its text matches an earlier request in the same run at a similarity of at least 0.75.

ALFWorld. We sample 18 tasks stratified across all six task types (three each), so that no single type dominates the probe, and run Full and Sliding on five paired repeats (one stochastic trajectory per cell; no API seed) (180 runs). Episodes are capped at 50 turns. Because the environment exposes no query tool, actions are classified by their leading verb: `look`, `examine`, and `inventory` are retrieval-like; `take`, `open`, `close`, `put`, `clean`, `heat`, `cool`, and `use` are execution-like; `go` is navigation and is counted separately. Unlike IRBench, the operator is applied here as a fixed 4000-token context budget rather than a ratio, because the two environments are not token-matched. Absolute completion in the probe is low (17 of 180 runs), and as Section A.7 reports, that budget was never reached, so we read this arm as uninformative rather than as evidence of a costless operator.

Termination. We mark a run *nudged* when the agent stopped calling tools without completing and the harness prompted it to continue (at most three times); the remaining labels are the agent reporting completion and the horizon being reached. Termination is dominated by the fixed horizon rather than by the agent stopping early. On DeepSeek High, the number of runs of ten that end with all tasks complete falls from 3 at $1\times$ to 2 at $5\times$ and 1 at $10\times$, with the remainder reaching the horizon or being nudged by the harness (Table 10); on Low-IR every ratio except $10\times$ finishes all ten runs. In τ-bench, 9 of 50 sliding runs and none of the 50 full runs reach the 30-step cap. All 180 ALFWorld runs terminate within the turn cap, by task success or failure.

Table 10: Termination by compression ratio, DeepSeek High (10 runs per ratio), with the companion Low-regime counts.

	$1\times$	$1.7\times$	$2.5\times$	$5\times$	$10\times$
High: all tasks complete	3	6	4	2	1
High: horizon exhausted, no nudge	7	2	6	8	9
High: ended after a nudge	0	2	0	0	0
Low: all tasks complete	10	10	10	10	9

Auxiliary tool budget. Writing α for the multiplier, a run's tool budget is $\alpha\times$ (number of required tasks + the minimum number of executions needed to complete them), with $\alpha = 2.0$ throughout. `within_budget` — whether a run stayed under that budget — and the overhead ratio against the minimum execution count are reported as diagnostics only; success is task correctness, and the binding runtime constraint is the turn horizon. In the GPT-5.6 Sol cells of the six-cell matrix the diagnostics behave accordingly: `within_budget` falls from $10/10$ runs at $1\times$ to $2/10$ at $5\times$ under High-IR and from $10/10$ to $3/10$ under Low-IR, with the overhead ratio rising from 2.51 to 4.24 and from 2.01 to 4.39 respectively (turns $22.6 \rightarrow 23.7$ and $13.4 \rightarrow 18.1$). The auxiliary budget is thus not the binding constraint in either regime even where reacquisition more than doubles, and we do not read it as a second performance metric.

Deployment checklist. To diagnose this cost in another runtime: (1) log every tool call with its name and arguments, and classify them as state-reacquiring versus task work before looking at outcomes; (2) report task completion alongside the reacquisition count, never the former alone; (3) fix the

interaction budget and report how many runs exhaust it, so that a cost hidden by a flat completion curve is still visible; (4) where the runtime allows it, run one condition with the dropped state restored — if the reacquisition count falls, the cost is recoverability-driven rather than a compression artifact; and (5) check whether the dropped state can be re-observed directly in the environment, since that is the property our own probe was unable to vary.

A.12 SEED-LEVEL ASSOCIATION

For each of the six model–regime cells we compute, per seed, the paired change in retrieval calls (ΔC_R) and completion (ΔQ, pp) at $5\times$, and their Spearman correlation. None of the three High-regime cells reaches significance: the association is negative for DeepSeek ($\rho = -0.40$, $p = .25$) and Qwen ($\rho = -0.32$, $p = .37$), and indistinguishable from zero for GPT-5.6 Sol ($\rho = +0.05$, $p = .89$). In the DeepSeek Low cell the association is undefined (completion invariant across seeds); the Qwen and GPT-5.6 Sol Low associations are weak and non-significant ($\rho = +0.16$, $p = .67$; $\rho = -0.22$, $p = .55$). In the earlier generation the GPT-5.5 High cell reached nominal significance ($\rho = -0.64$, $p = .045$), which the current generation does not reproduce. We report these as exploratory, descriptive within-seed associations, not causal estimates.

A.13 STATISTICAL CALIBRATION AND SENSITIVITY

Design resolution. With ten paired seeds and a binary task-level outcome, we report a normal-approximation minimum detectable effect (MDE) as the design's resolution at $\alpha = .05$ (two-sided) and 80% power, $\mathrm{MDE} = (z_{.975} + z_{.80})\,\mathrm{sd}(\Delta)/\sqrt{n} = 2.8016\,\mathrm{sd}(\Delta)/\sqrt{n}$. For interpretive purposes we report the MDE for comparisons that do not reject the pre-specified test; comparisons that do reject it are reported as detected effects rather than characterized by an MDE. Table 11 gives the value for each cell, together with projections to larger seed counts.

Table 11: Design resolution at $n = 10$ paired seeds (companion to Table 3). Δ and $\mathrm{sd}(\Delta)$ are in each metric's own units (C_R: tool calls; completion: percentage points). MDEs are computed at $\alpha = .05$ (two-sided) and 80% power from the observed $\mathrm{sd}(\Delta)$; the n=20 and n=40 columns are *projections*, not measurements. "N/A": zero observed paired variance (both conditions at 100%), so no MDE is defined. For the Qwen-High retrieval cell the extension of Section A.2 gives an *observed* twenty-seed resolution of 3.4 calls (sd 5.4), against the projection of 2.7 obtained from the original ten-seed sd.

Cell	metric	Δ	$\mathrm{sd}(\Delta)$	$\mathrm{MDE}_{n=10}$	$\mathrm{MDE}_{n=20}$	$\mathrm{MDE}_{n=40}$
DeepSeek High	C_R	$+32.9$	13.0	11.5	8.1	5.8
DeepSeek High	comp.	-11.0	22.8	20.2	14.3	10.1
DeepSeek Low	C_R	$+22.6$	13.8	12.3	8.7	6.1
DeepSeek Low	comp.	$+0.0$	0.0	N/A	N/A	N/A
Qwen High	C_R	$+2.9$	4.3	3.8	2.7	1.9
Qwen High	comp.	-9.0	13.7	12.1	8.6	6.1
Qwen Low	C_R	$+5.3$	5.8	5.1	3.6	2.6
Qwen Low	comp.	-6.0	10.7	9.5	6.7	4.8
GPT-5.6 Sol High	C_R	$+24.2$	9.4	8.4	5.9	4.2
GPT-5.6 Sol High	comp.	$+0.0$	9.4	8.4	5.9	4.2
GPT-5.6 Sol Low	C_R	$+25.2$	12.6	11.2	7.9	5.6
GPT-5.6 Sol Low	comp.	-3.0	6.7	6.0	4.2	3.0

Two consequences affect how the results should be read. (i) The single non-significant retrieval comparison (Qwen High, $\Delta = +2.9$) has an observed displacement below its own resolution (3.8 calls): the experiment does not resolve whether a smaller retrieval effect is present there, and we do not read it as the absence of a response. (ii) The DeepSeek-Low and GPT-5.5-Low completion cells have zero observed paired variance (both conditions at 100%), so no MDE is defined; they are ceiling-censored and neither support nor contradict a completion effect. The GPT-5.6 Sol High completion interval ($[-5, +6]$ pp) is the tightest among the High-regime cells.

A third reading follows from the same table. The completion channel is coarse in absolute terms: in every cell where its MDE is defined the design cannot detect a change smaller than 6 pp, whereas in the three cells with the largest retrieval increases (DeepSeek High, GPT-5.6 Sol High and Low) the

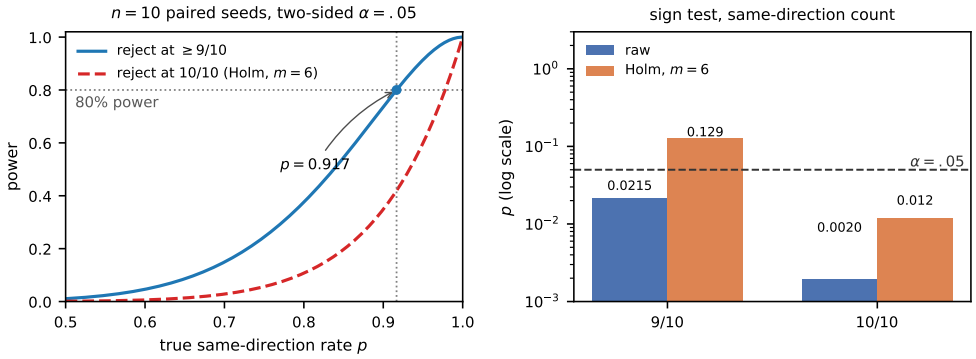

Figure 9: Power of the paired sign test at $n = 10$ (left; two-sided $\alpha = .05$, computed exactly) and the effect of Holm correction within the six-comparison family (right).

same runs resolve 24 to 33 additional calls against a completion change of at most 11 pp that is not detected. Completion is what a deployment monitors, so this asymmetry is itself part of the result: a completion null is weak evidence about interaction cost, and we read it against the retrieval signal measured in the same runs — which is how the Results section reports it.

Exact-test sensitivity. The count of significant retrieval comparisons is sensitive to the choice of paired test at this sample size: five of six under the pre-specified paired test, versus four of six under an exact sign test (Table 12). An exact permutation test on the signed-rank statistic agrees with the pre-specified Wilcoxon test in every cell, so the discrepancy is confined to the sign test and not to the rank family we pre-specified.

Table 12: Sensitivity of the six-comparison retrieval family to the paired test ($n = 10$ seeds; companion to Table 3). "exact sign" is a two-sided exact sign test on the non-zero paired differences; "exact perm" enumerates all 2^{10} sign assignments of the signed-rank statistic; Holm correction is applied within the six-comparison family.

Cell	ΔC_R	Wilcoxon p	exact perm p	exact sign p	Holm p
DeepSeek High	+32.9	0.0020	0.0020	0.0020	0.0117
DeepSeek Low	+22.6	0.0039	0.0039	0.0039	0.0117
Qwen High	+2.9	0.0879	0.0879	0.1094	0.0879
Qwen Low	+5.3	0.0234	0.0234	0.1797	0.0469
GPT-5.6 Sol High	+24.2	0.0020	0.0020	0.0020	0.0117
GPT-5.6 Sol Low	+25.2	0.0020	0.0020	0.0020	0.0117

Power at this sample size. The design's resolution has a second face: not the size of the effect it could detect, but how often it would detect one. A two-sided exact sign test on ten paired units rejects only at $9/10$ same-direction or better, so reaching 80% power against a true same-direction rate requires that rate to be at least 0.917; at that rate a $10/10$ requirement is met only 42% of the time (Figure 9, left panel). Inside the Holm-corrected six-comparison family the operative requirement does become $10/10$, since a $9/10$ result leaves an adjusted $p = .129$ while a $10/10$ result gives $p = .012$. A family of six comparisons at this sample size is therefore decisive only when no cell shows a single reversal, and the cells reported as non-significant above should be read as underpowered rather than as evidence against an effect. The Qwen-High extension in Section A.2 illustrates this limitation empirically: its point estimate stayed stable across twenty paired seeds, while the inferential conclusion remained sensitive to the paired test and to individual observations. Resolution improves as $1/\sqrt{n}$, so doubling the paired sample reduces the smallest detectable effect by $\sqrt{2}$ in every cell (Table 11); we therefore treat a larger paired sample as the priority for any extension of this design.

Interval calibration. The percentile bootstrap behind the reported 95% intervals is anti-conservative at $n = 10$. Taking the observed paired differences as the population and resampling ten at a time, its measurable coverage on the Qwen-High completion cell is 0.90 against a nominal 0.95.

Where an exact alternative exists we therefore also report the exact sign-flip interval, obtained by enumerating all 2^{10} sign assignments: on that cell it is $[-9, +9]$ pp and contains zero, consistent with the pre-specified Wilcoxon test, whereas the percentile interval for the same cell is $[-17, -1]$ pp — an exclusion of zero that reflects the under-coverage rather than evidence against the null. We report the percentile intervals for comparability with the pre-specified analysis, and read them with this caveat.