WHAT COMPRESSION PRESERVES: INSTRUCTION SURVIVAL IN LONG-HORIZON LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

When a long-running LLM agent fills its context window, production frameworks compress its history and continue. This compaction becomes the agent's effective memory of its own session, but it is unclear which standing instructions survive the rewrite. Across 1,175 evaluable trajectories on τ-bench retail and AppWorld, we vary three compression trigger thresholds, two compression methods (truncation and LLM summarization), and three reminder frequencies against a no-compression baseline, mechanically scoring every planted instruction. We find selective forgetting based on whether later trajectory content re-cues a requirement: on τ-bench, two task-external output requirements stated in the first user message drop from 82% and 87% without compression to 22% and 32% across compressed conditions, whereas rules coupled to the ongoing trajectory stay at 89–100% when pooled over all conditions. Among compressed trajectories that complete the task, 63% and 55% still violate these two requirements, so task success alone does not measure memory faithfulness. The pattern persists under changes to compaction settings and in a 40-task pilot with a second agent model family. The most aggressive compression also reduces task success by up to 24.7 (τ-bench) and 33.6 (AppWorld) percentage points and may increase input-token use: under uncached full-schema billing, compressed τ-bench conditions use 13–42% more input tokens than baseline; caching the schema prefix largely removes truncation's excess but not summarization's. Scheduling a single mid-conversation reminder improves compressed τ-bench adherence by 5.6 percentage points, an effect that does not replicate on AppWorld. We release the evaluation harness, constraint catalogue, and trajectories.

1 INTRODUCTION

Long-horizon LLM agents accumulate their history in the context window, and when it fills, production frameworks compress it and continue. For most deployed agents, that compaction step is the only memory of their own session: whatever the summary or the retained tail carries forward is all the agent still knows about its own past, including the standing instructions it was given. Consider a retail-support agent whose initial user message specifies a response wrapper, a completion marker, and a prohibited word. After a dozen turns, the framework compacts the history and removes that instruction block with the oldest messages. The agent completes the task and avoids the prohibited word, but omits both output conventions. A downstream parser can then fail despite successful task completion. Across compressed cells, adherence to these two requirements, stated in the first user message, falls to 22% and 32%, from no-compression baselines of 82% and 87%.

Compaction is standard infrastructure. OpenAI's Responses API exposes a server-side `/responses/compact` endpoint, Anthropic ships compaction with context-editing controls, Google's ADK bundles context summarization, and Claude Code-style agents auto-compact near the context limit (OpenAI, 2025; Anthropic, 2026b;c; Google, 2026; Anthropic, 2026a; Rajasekaran et al., 2025). All are designed to carry the instructions in force across that boundary; whether they do, and for which instructions, has not been measured. Agent memory systems are usually evaluated on what they recall: facts, preferences, and prior sessions (Wu et al., 2024; Chhikara et al., 2025). Within-session compaction is a different mechanism from these cross-session memory stores, but it acts as a form of session memory, and we evaluate what it *forgets*, using standing instructions as a mechanically checkable probe.

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 ask three questions. Which instructions survive compression and which are lost? How does compression affect task success and input-token use? Does repeating an instruction mid-conversation protect it?

Our central finding is that compression failure tracks whether later task interactions re-cue a requirement, not simply which compression algorithm is used. Task-external output requirements whose only cue sits in an early user message are easily erased; rules coupled to ongoing task behaviour are much more stable. The split survives a post-hoc sweep of the compaction target and cooldown and is reproduced in a 40-task pilot with a second agent model family. Consequently, compaction should be evaluated on instruction adherence and task success, alongside input-token use under the serving protocol.

Prior work answers different questions, chiefly how to compress, through token pruning, learned soft prompts, or trajectory pruning (Jiang et al., 2023; 2024; Pan et al., 2024; Chevalier et al., 2023; Ge et al., 2024; Xiao et al., 2025). CDCT (Baxi, 2025) shows that single-turn prompt compression hurts constraint compliance 2.9$\times$ more than semantic content, and concurrent agent benchmarks document agents violating instructions while completing tasks (Li et al., 2025; Ye et al., 2026; Sharma, 2026). No study varies all three factors (when compression fires, how it compresses, and how often instructions are restated) on multi-turn agent benchmarks while tracking which instructions break.

We run that experiment. On τ-bench retail (Yao et al., 2025) and AppWorld (Trivedi et al., 2024), we cross three trigger thresholds (compression fires at 50%, 70%, or 90% of a fixed token budget), two methods (truncation and LLM summarization), and three reminder frequencies (an instruction stated once, restated once mid-trajectory, or restated periodically), plus a no-compression reference: 1,175 evaluable trajectories from one open-weights agent, with every planted instruction scored by a simple program, not an LLM judge. One structural difference matters throughout: τ-bench re-sends roughly 3K billed tokens of typed tool schemas (4.4K with the system prompt) with every model call, whereas AppWorld routes all actions through one `world.execute` call and has no comparable overhead. Later sections call a threshold–method pair a *cell* and a reminder frequency a *band*.

Contributions.

1. Compression destroys task-external output requirements far more often than trajectory-coupled rules: on τ-bench, two output wrappers fall from 82%/87% without compression to 22%/32% across compressed cells, while tool-cued rounding and casing rules remain at 99–100% when pooled over the sweep (Section 4.1). The split holds under a post-hoc sweep of the compaction target and cooldown and is reproduced in a 40-task pilot with Llama-3.3-70B as the agent (Section 4.2).
2. Input-token use is protocol-dependent: under uncached full-schema accounting every compressed τ-bench configuration uses 13–42% more input tokens than no compression; cached-schema accounting removes truncation's excess but leaves summarization 19–35% higher (Section 4.4).
3. The most aggressive compression lowers task success by up to 24.7 (τ-bench) and 33.6 (AppWorld) percentage points, and which method is safer flips between benchmarks: truncation is worst on AppWorld yet competitive on τ-bench, consistent with schema overhead on τ-bench and stateful code execution on AppWorld (Section 4.3; Appendix B).
4. Scheduling one mid-trajectory restatement raises compressed τ-bench adherence by 5.6 percentage points (95% CI $[+0.8, +10.1]$); the lift does not replicate on AppWorld, where adherence is already near its ceiling (Section 4.5).

Scope. The findings are conditional on two benchmarks, one open-weights agent, and threshold-triggered compression; Section 6 revisits the limits.

2 Related Work

Compression methods. LLMLingua, LongLLMLingua, and LLMLingua-2 (Jiang et al., 2023; 2024; Pan et al., 2024) prune tokens by perplexity, position-aware scoring, or distilled token-level decisions; AutoCompressors and ICAE (Chevalier et al., 2023; Ge et al., 2024) learn soft-prompt compressors at fixed segment boundaries; Selective Context (Li et al., 2023) and RECOMP (Xu

et al., 2024) drop or rewrite low-information segments; PCToolkit (Li et al., 2024) unifies five such methods, and AgentDiet (Xiao et al., 2025) prunes expired or redundant trajectory tokens. None of these vary a timing policy.

Learned timing policies. MEM1 (Zhou et al., 2025) compresses every turn under a fixed memory budget, MemPO (Li et al., 2026) optimises memory operations with dedicated credit assignment, and ACON (Kang et al., 2025) optimises natural-language compression guidelines with a two-level threshold ablation; Du (2026) survey related joint-optimisation approaches. These policies deliver strong task performance but entangle when to compress with what to compress.

Production memory systems. MemGPT/Letta (Packer et al., 2023) pages state in and out of context, Mem0 (Chhikara et al., 2025) extracts and consolidates episodic memory across sessions, A-MEM (Xu et al., 2025) maintains atomic memory items with attention-style retrieval, LongMemEval (Wu et al., 2024) stress-tests multi-session recall, and StreamingLLM (Xiao et al., 2023) bounds the window through attention-sink retention. These systems are evaluated on conversation-level recall; whether the within-trajectory compaction they rely on preserves standing instructions is the question left open here.

Constraint adherence. CDCT (Baxi, 2025) finds that constraint compliance degrades 2.9× more than semantic accuracy under single-turn prompt compression. Three concurrent multi-turn agent benchmarks vary the constraint axis but hold compression fixed: ODCV (Li et al., 2025) catalogues outcome-driven constraint violations, CCTU (Ye et al., 2026) stresses tool use under co-occurring constraints, and ContextCov (Sharma, 2026) enforces executable constraints extracted from instruction files at runtime. Lost-in-the-Middle (Liu et al., 2024) and RULER (Hsieh et al., 2024) show context-position effects that interact with timing policy without testing the interaction, and LOCA-bench (Zeng et al., 2026) compares six context-management strategies, including compaction, with fixed timing and binary success only. We contribute a joint measurement of when compaction fires, how it compresses, and which standing instructions survive, on existing multi-turn agent benchmarks. Table 2 (Appendix D) positions the closest prior work along the dimensions varied here, with "timing" defined narrowly as the policy that decides when to compact a multi-turn agent context.

3 SETUP

3.1 SUBSTRATES

τ-bench retail (Yao et al., 2025) simulates a customer-service agent over 115 retail tasks; the agent calls 16 typed retail-tool schemas (return, lookup, and the rest) and converses with an LLM-driven user simulator. Reward is mechanical: `Env.calculate_reward()` returns 0 or 1 against ground-truth state transitions. AppWorld (Trivedi et al., 2024) tasks an agent with executing Markdown-fenced Python that calls a sandboxed REST simulation; tool use is implicit (one `world.execute` channel), and the task evaluator (`world.evaluate()`) checks final database state. We pull 74 τ-bench tasks (random subset of the retail split) and 20 AppWorld tasks (sampled from the train/dev pool), each run with two repeats per cell.

The two *substrates* (our term for the benchmark environments) differ structurally. τ-bench's system prompt plus 16 tool schemas pins an empirical floor near 3,500 tokens that compression cannot reduce further (Appendix B, Figure 5). AppWorld has no tool schemas; its system prompt fits in under 1,000 tokens. Compression therefore has very different headroom on each substrate. We selected these two for their structural contrast (typed-tool API protocol vs. Markdown-fenced code execution); other multi-turn agent benchmarks (Jimenez et al., 2024; OpenAI, 2024; Xie et al., 2024; Mialon et al., 2023; Zhou et al., 2023) fall in adjacent regimes where compression mechanics may differ (Section 6).

3.2 COMPRESSION METHODS

We instantiate two *model-agnostic* methods (we call each compression event a *compaction*), both protecting the system prompt verbatim and operating only on `messages[1:]`.

Truncation drops the oldest non-system message, then drops orphan `tool` messages whose preceding `assistant` tool call has been removed (otherwise the API rejects the request), repeating until token count is below the post-compression target.

Abstractive summarization never summarises tool traffic: tool calls and their results are carried over unchanged. It splits the to-be-compressed region into prose and tool-related messages, sends the prose to a summariser LLM (fixed prompt: goals, constraints, tool calls, pending sub-goals), and reassembles the message list as $[\text{system}, \text{summary}, \text{verbatim tool calls}, \text{recent tail}]$. The most recent four messages are always preserved.

3.3 TRIGGER POLICY

A utilization-based trigger fires when the pre-call estimated token count exceeds $\theta \cdot W$ for trigger threshold $\theta \in \{0.50, 0.70, 0.90\}$ and context window $W = 8{,}000$ tokens. W is an experimenter-imposed trigger denominator, set deliberately below the served model's context limit so that compression engages within our task lengths; it is not a hard serving constraint. The post-compression target is $0.30 \cdot W = 2{,}400$ tokens, with three-turn hysteresis (no two compactions within three agent turns) to prevent thrashing. We instantiate a no-compression reference cell where compaction is disabled and the full, uncompressed history is sent to the model on every call. The served model's context comfortably exceeds the longest trajectory, so baseline histories grow past W (baseline τ-bench prompts average 6.5K tokens per call) without truncation or any window-related failure; both baseline cells recorded zero compaction events (Appendix F). The post-compression target and hysteresis are held fixed in the main sweep and varied post hoc in Section 4.2. Local pre-call estimates undercount τ-bench prompts by 16.7% on average (Appendix F); nominal triggers therefore fire at effective utilizations roughly 17% higher on τ-bench (a nominal θ=0.70 trigger fires near true 0.82). We retain nominal labels throughout for clarity but interpret all τ-bench thresholds accordingly.

3.4 CONSTRAINT CATALOGUE

Each trajectory plants three constraints drawn from a fixed catalogue of seven (Appendix F, Table 3), instantiated and scheduled by an RNG seeded on $(\text{task_id}, \text{substrate}, \text{repeat})$, so within a repeat every method/threshold cell receives the same constraints. Every checker is a pure function over the saved trajectory, with no LLM judging. The catalogue was finalised pre-sweep by filtering items structurally unreachable on our task selection or already failing on the no-compression baseline.

We keep two labels separate. A constraint's *surface tag* describes what its checker tests: format, prohibition, count, or value. Our analysis instead groups constraints by *trajectory support*: whether later task interactions provide another cue for the required behaviour. *Task-external wrappers* are final-output obligations whose only cue may sit in an early user message; C1 and C6 form this group. *Trajectory-coupled rules* are evaluated through ongoing task behaviour, tool values, arguments, or generated code; C2–C5 form this group. This analytical grouping does not claim that C2–C5 share one semantic type or identical mechanism: later tool traffic directly re-cues C2/C3, while C4/C5 are expressed across the evolving trajectory.

The retained constraints are numbered consecutively: C1 wraps the final answer in `<answer>`; C2 rounds numerics; C3 requires snake_case keys; C4 prohibits "unfortunately"; C5 caps tool calls at N; C6 ends the response with `[DONE]`; and C7 addresses the user as `{title}`. Their surface tags are *format* (C1, C2, C3, C6), *prohibition* (C4), *count* (C5), and *value* (C7). C1 and C6 cannot be satisfied in AppWorld, whose trajectories end with `apis.supervisor.complete_task()` rather than a free-text answer; they are excluded from AppWorld rates. C7 is planted on τ-bench only. C7 is task-external but not a wrapper; Section 4.1 reports it separately.

3.5 REDUNDANCY BANDS

Each planted constraint is assigned to one of three redundancy bands that determine how often it is restated as a user-role message:

- **Once**: included only in the first user message.
- **Twice**: first user message plus one reminder midway (turn $\lfloor L/2 \rfloor$ for expected length $L = 16$ on τ-bench, 25 on AppWorld: turns 8 and 12).
- **Periodic**: first message plus a reminder every K turns up to L ($K = 8$, turns 8 and 16, on τ-bench; $K = 5$ on AppWorld).

Bands operate *within the truncatable region* (`messages[1:]`), not the system prompt; they therefore probe the compression-induced redundancy effect, not prompt-side restatement. The three constraints in a trajectory are assigned to the three bands by a per-task RNG; balance is exact when the constraint count is a multiple of three.

3.6 MODELS AND RUN PROTOCOL

The agent is Qwen3.5-27B (Qwen Team, 2026), with temperature 0, using a maximum of 30 (τ-bench) or 50 (AppWorld) ReAct turns. The τ-bench user simulator and summariser are Qwen3-32B; the same two roles use Qwen3.5-9B in 53 trajectories. Pre-call token counts are estimated locally and reconciled against the provider's post-call usage report. The main sweep contains 1,175 evaluable trajectories across both substrates. Reasoning tokens account for 37% of agent output on τ-bench and 63% on AppWorld.

3.7 MEASUREMENT

Task success is binary, from the substrate's native evaluator; constraint adherence is the fraction of planted constraints passing their mechanical checker. Pooled rates (each trajectory weighted equally) and per-task-equal-weighted rates agree within 2.4 pp on every cell (Appendix F), so pooled rates are not driven by a few easy tasks. The primary evidence is descriptive cell rates with bootstrap or Wilson intervals; Appendix F adds exploratory GLMM and FDR diagnostics. With $n \approx 130$ (τ-bench) and $n \approx 36$ (AppWorld) per cell, binomial 95% CIs on pooled rates are roughly ± 8.5 and ± 16 pp; effects above those thresholds are reported as findings, and the design is not a confirmatory hypothesis test.

4 RESULTS

The primary evidence is the cell-rate pattern and uncertainty intervals; we report cluster-bootstrap 95% CIs for constraint-level rates over trajectories (1000 resamples).

4.1 TASK-EXTERNAL WRAPPERS ERODE; TRAJECTORY-COUPLED RULES SURVIVE

Which kinds of planted constraint survive compression? Figure 1 reports per-constraint satisfaction pooled across all cells; the rates split sharply by trajectory support, not by the checker's surface tag. C2–C4 stay close to their no-compression rates on both substrates: rounding (C2: 99%/98%), snake_case keys (C3: 100%/100%), prohibited word (C4: 89%/100%); no C2–C4 delta survives FDR correction on either substrate (Appendix F). The tool-call cap C5 (pooled 92%/78%) does degrade at aggressive thresholds (τ-bench 100%→83–90%; AppWorld 95%→43–53% at θ=0.50). Task-external wrappers are low in the pooled figure: `<answer>` tags (C1) at 30% and the `[DONE]` marker (C6) at 40%. Restricting to compressed cells, the rates are 22% and 32%, down from no-compression baselines of 82% and 87% under the same checkers. Among compressed trajectories that succeed, 63% (90/143) violate C1 and 55% (59/107) violate C6. A human validity audit was not run, so these levels remain mechanical-checker measurements (Section 6).

C7 ("address user as `{title}`") is task-external but not a final-output wrapper: its checker scans all assistant text. It falls from 100% at baseline to 53% across compressed cells, with similar loss for every sampled title (43–51% violated; Table 6), so erosion is not specific to wrappers. Because its grouping was fixed post hoc, we report it separately from the C1/C6 contrast.

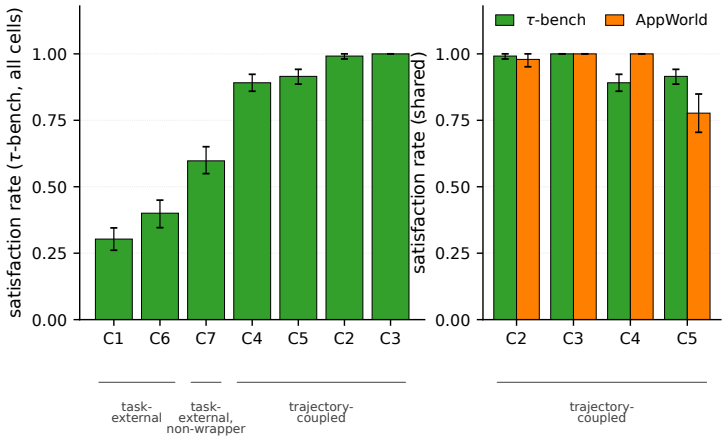


Figure 1: Per-constraint satisfaction pooled over all seven cells, baseline included (per-cell rates: Table 6). Caps are cluster-bootstrap 95% CIs over trajectories. Left: τ-bench, all seven catalogue constraints sorted ascending. Task-external wrappers (C1 `<answer>`, C6 `[DONE]`) sit at 30% and 40%; trajectory-coupled rules (C2–C5) hold at 89–100%; C7 (form of address) is task-external but not a wrapper and is reported separately (see text). Right: shared C2–C5 across both substrates. C1/C6 cannot be satisfied on AppWorld and are excluded there; C7 is τ-bench only.

Compression does not induce rounding, casing, or prohibited-word violations; what falls off is the task-external instruction whose only evidence appears in the final wrapper, the kind of constraint a deployment attaches to enforce a downstream parsing convention. Wrappers like `<answer>` and `[DONE]` are stated in user messages that compression can remove. By contrast, rounding and casing are implicitly re-cued in later tool traffic (numeric returns and tool arguments), so the trajectory itself continues to support the requested behaviour. Figure 3 (Appendix A) shows the split inside one real trajectory.

Pooled adherence. Mean constraint adherence across compressed cells is 68.5% on τ-bench versus 93.6% baseline (a 25.1 pp gap), and 93.1% on AppWorld versus 97.4% baseline (4.3 pp). On τ-bench, roughly one in four planted constraints the baseline would have honoured is dropped under compression.

4.2 The split survives compaction-parameter changes and a second model family

Two post-hoc checks, run after the main sweep on a fixed, seeded 40-task τ-bench subset (one repeat per cell, 360 trajectories; Appendix C), ask whether the split depends on the fixed compaction parameters or on the agent model. First, at $\theta = 0.50$ we vary the post-compression target ($0.30\,W \rightarrow 0.75\,W$) and the cooldown (three turns $\rightarrow$ one or six). Levels move strongly: pooled adherence ranges from 46.7% (cooldown 1, summarization, 24 compactions per trajectory) to 92.5% (target $0.75\,W$, truncation), and C1/C6 rates from 0%/0% to 67%/89%. The split does not: C2 and C3 are at 100% in every arm, and in every arm the C2/C3-minus-C1/C6 gap (+22 to +100 pp) exceeds the main-sweep no-compression gap (+16 pp), though only narrowly with a $0.75\,W$ target and truncation, where C1/C6 nearly recover (67%/89%). Second, replacing the agent with Llama-3.3-70B-Instruct (Meta AI, 2024) (same summariser, $\theta = 0.50$) reproduces the pattern: C1/C6 fall from 89%/87% at baseline to 59%/43% under truncation and 58%/18% under summarization while C2/C3 stay at 100%, a gap of +49 and +62 pp against +12 pp at baseline. Task-success levels and the method ranking are not preserved at this sample size (Llama baseline 23.1%; summarization 28.2% vs. truncation 12.5%, $n \approx 40$ per arm), so the τ-bench method ranking in Section 4.3 should be read as specific to the Qwen agent. These arms use a smaller and differently drawn task sample than the main sweep, so we read them as qualitative replications of the split, not as paired estimates of its size.

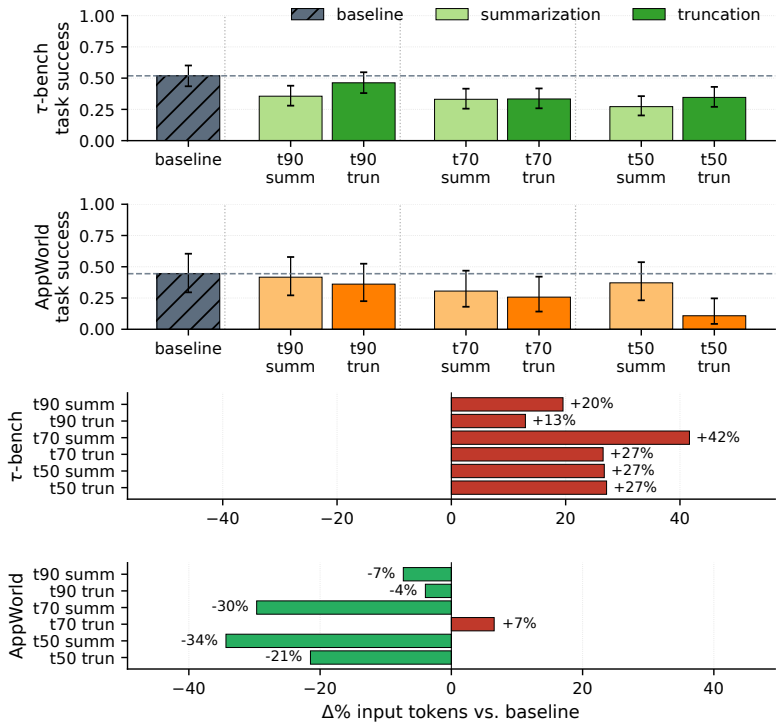


Figure 2: **Top:** pooled task success per cell on each substrate, grouped by trigger threshold (baseline | θ=0.9 | θ=0.7 | θ=0.5). Caps are Wilson 95% CIs; dashed line is the no-compression baseline. **Bottom:** mean agent input tokens per evaluable trajectory under uncached full-schema accounting, expressed as a percent change from baseline. τ-bench uses more input tokens in every compressed cell; AppWorld uses fewer in five of six cells.

Why the substrates diverge. τ-bench compaction frees only 19% of pre-event tokens (protected prompt and schemas), AppWorld's 71% (Appendix B).

4.3 Task success: up to 34 pp lost; method ranking flips

How does compression affect end-to-end task success? Figure 2 (top) reports task success per cell; cells are named by threshold and method, so $t50$*_summ* is the θ=0.50 summarization cell and *trun* abbreviates truncation. At the most aggressive threshold, pooled task success falls on τ-bench from 51.9% baseline to 27.2% at $t50$*_summ* (a 24.7 pp loss); AppWorld drops from 44.4% to 10.8% at $t50$*_trun* (a 33.6 pp loss). Lightest-compression AppWorld cells overlap the baseline within the binomial CI of ± 16 pp at $n \approx 36$. The pooled method ranking differs by substrate: summarization performs better than truncation on AppWorld and worse on τ-bench. Because AppWorld repeats are correlated (ICC 0.73), we resample whole tasks. The task-paired contrast for summarization minus truncation is +15.1 pp on AppWorld (95% CI $[+7.2, +23.9]$, $k = 19$); the pooled cross-substrate interaction is +21.7 pp (CI $[+11.5, +32.4]$). The τ-bench pooled contrast is -6.6 pp (CI $[-13.0, -0.4]$), although its $\theta = 0.50$ slice includes zero. We therefore interpret the method difference at the pooled substrate level, not at every threshold. Per-cell numbers are in Appendix F.

4.4 Input-token use depends on static-prefix caching and compression method

Compression lowers instruction adherence and task success. Does it also reduce input-token use? Under the experiment's uncached full-schema accounting, Figure 2 (bottom) shows that every τ-bench compression cell uses 13–42% more input tokens than the no-compression baseline; the worst cell ($t70$*_summ*) uses 41.7% more input tokens and achieves 18.8 pp lower task success. AppWorld

uses 4–34% fewer tokens on five of six cells; only $t70_trun$ uses more than baseline (+6.5%). This result depends on the accounting assumptions.

A post-hoc simulation using the per-call token counts makes the dependency explicit. Regression over 1,489 baseline calls estimates 3,051 input tokens per call for tool schemas ($R^2 = 0.999$). Under a simulated policy that caches this schema prefix after the first call, truncation's three τ-bench deltas fall to +1.1%, +3.2%, and +3.6% relative to the correspondingly cached baseline, with all task-clustered 95% CIs spanning zero. Summarization remains +19.2%, +35.4%, and +19.0%, with all three CIs excluding zero. Under the more generous assumption that the full 4,356-token static prefix is cached, truncation uses 12–23% fewer input tokens, whereas summarization still uses 10–29% more. Thus truncation's uncached excess is explained by additional ReAct turns repeatedly including the static prefix, while summarization's excess persists after static-prefix reuse (uncached per-cell numbers: Appendix F).

4.5 Redundancy bands: one reminder helps on τ-bench, fails to replicate on AppWorld

If task-external wrappers die because compression removes the user messages that state them, can restating a constraint mid-trajectory buy survival back? The *redundancy hypothesis* says yes: if rarely-restated items erode faster under compression than frequently-restated ones, satisfaction should be monotone in band. Figure 4 (Appendix E) plots satisfaction by band over the compressed trajectories. Over the 789 compressed τ-bench trajectories, the cluster-bootstrap contrast is $+5.6$ pp Once-to-Twice (95% CI $[+0.8, +10.1]$; $+4.8$ pp without C7) and -1.3 pp Twice-to-Periodic (CI $[-6.0, +3.3]$). Bands are intention-to-treat: τ-bench reminders are spliced only when the agent replies to the user at the scheduled turn (60 of 135 baseline trajectories), and the turn-16 Periodic reminder was essentially never reached, so Twice and Periodic differ in schedule, not in delivered treatment. The exploratory GLMM does not resolve a band effect (Table 4).

The lift does not replicate on AppWorld. Pooled AppWorld band rates remain high and non-monotone, leaving little room for a reminder effect: AppWorld's scorable constraints C2–C4 stay at 95–100% in every cell. On τ-bench the lift does not reach the output wrappers (compressed C1 17/18/30%, C6 33/35/28% for Once/Twice/Periodic); Section 4.1 shows which constraints erode regardless of restatement.

5 Discussion

Central argument. Compression failure tracks whether later task interactions re-cue a requirement: task-external output requirements whose cues can disappear with early user messages erode, while trajectory-coupled rules hold (Section 4.1), and the split persists across compaction parameters and a second agent model (Section 4.2). An agent can therefore complete the benchmark task while violating a standing output requirement, so task success alone is not an adequate compaction metric.

Implications for production compaction. Aggressive compression also imposes a task-success cost on both substrates (τ-bench: 24.7 pp; AppWorld: 33.6 pp at $\theta = 0.5$); unlike requirement loss on otherwise successful trajectories, this effect is visible to task-success dashboards. Light compression on AppWorld is statistically consistent with no compression. The pooled method-ranking difference survives task-clustered resampling and is consistent with tool-schema overhead on τ-bench and IPython statefulness on AppWorld, though the data do not isolate their contributions.

Input-token use is protocol-dependent. Under uncached static full-schema prompting, every compressed τ-bench condition uses more input tokens because extra agent turns repeatedly include the static prefix. Cached-prefix accounting changes the pattern by method: truncation's excess vanishes or reverses, while summarization remains higher because its per-call history stays near baseline size as the agent takes additional turns. Token use therefore depends on turn count, static-prefix reuse, and the platform's caching and tool-delivery protocol.

Recommendations. Constraint adherence should be tracked alongside task success, and input-token accounting should include agent-turn count and effective prefix reuse. Task-external requirements

and policy invariants belong in the protected system prompt (untested here; our compactors never modify it); scheduled mid-trajectory restatement, as implemented, did not measurably protect C1/C6. For agent memory systems more broadly, and for compaction in its role as session memory, the result argues for evaluating what is *forgotten*, not only what is recalled: standing instructions are a cheap, mechanically checkable probe of memory faithfulness that recall benchmarks do not exercise.

6 LIMITATIONS

Evaluation coverage. We study three thresholds and two local compression methods on two structurally chosen benchmarks, with a catalogue hand-crafted to be checkable without an LLM judge; real production constraints, such as return-window policies or fare-class restrictions, would require a separate validation protocol. The task-external class is only two wrappers, with C7 the only non-wrapper case. Several checkers pass by default: C3 inspects tool-argument keys, which τ-bench schemas fix as snake_case and AppWorld reduces to one `code` key, so it cannot fail here; C2 passes when the final text has no decimals (22% of τ-bench, 93% of AppWorld instances). Their stability shows that compression does not induce violations, not that the instruction was retained; the active obligations C1, C6, and C7 all erode. Selective tool-output clearing preserves message-type semantics that summarization and truncation destroy, so the gap between task-external and trajectory-coupled requirements may narrow or invert under other compaction operators.

Sample and measurement uncertainty. The analysis uses 924 τ-bench and 251 AppWorld trajectories; AppWorld cell sizes are about 36, giving binomial intervals of roughly ± 16 pp and imprecise per-threshold method comparisons. The task-clustered analysis supports the pooled cross-substrate method difference. τ-bench reminders were delivered only when the agent replied to the user at the scheduled turn, so the band contrast is intention-to-treat. The 22/32% compressed-cell C1/C6 rates are based on mechanical checkers that were not independently hand-audited.

Compression regime. The post-compression target ($0.30 \cdot W = 2{,}400$ tokens) and the three-turn cooldown are fixed, and both bind in the data. The minimum observed gap between compactions is three turns; on τ-bench the target is never reached and the fifth percentile of post-truncation states is 3,546 tokens, while AppWorld reaches the target (median post-compaction size 1,424). A post-hoc sweep on a 40-task subset (Section 4.2) shows that both parameters move adherence and task-success levels by tens of points while leaving the trajectory-support split intact, so the main-sweep levels should not be read as parameter-free. Cached-prefix billing changes token accounting, not model behaviour; larger windows, adaptive targets, provider-native compaction, and tool search require new trajectories.

Model family. All main-sweep trajectories use Qwen3.5-27B as the agent and Qwen3-32B as user simulator (τ-bench) and summariser (Qwen3.5-9B for 53 τ-bench trajectories), so simulator and summariser share the agent's model family. A post-hoc $\theta = 0.50$ pilot with Llama-3.3-70B-Instruct as the agent (Section 4.2) reproduces the split but not the τ-bench method ranking. A frontier-class summariser might change τ-bench summarization specifically; whether the substrate-asymmetric findings reproduce on model families beyond this pilot is open.

7 CONCLUSION

Our strongest supported result is that compression failure tracks whether later task interactions re-cue a requirement: on τ-bench, task-external output requirements fall from 82%/87% without compression to 22%/32% across compressed cells, while trajectory-coupled rules stay at 89–100% pooled on τ-bench (78–100% on AppWorld, where the tool-call cap degrades), and the split persists across changes to the compaction target, cooldown, and agent model. Because this loss can coexist with successful task completion, compression should be evaluated on instruction adherence as well as task success. Input-token use depends on the protocol: prefix caching removes truncation's excess while summarization remains higher. A scheduled mid-trajectory reminder closes part of the τ-bench adherence gap, though not for the output wrappers, without transferring to AppWorld. Read as a form of session memory, threshold-triggered compaction is lossy, and its forgetting is selective, invisible to task-success metrics, and depends on whether later interactions re-cue the requirements.

AI USE STATEMENT

We used generative AI (Anthropic Claude models, including agentic coding tools) throughout this project. Uses in the required-disclosure categories: (a) proposing and refining hypotheses and the conceptual framing, through iterative discussion and multi-agent critique rounds; (b) feedback on research methodology and experimental design, via the same critique rounds, with final design decisions made by the authors; (c) implementing methods, including substantial parts of the evaluation harness, constraint checkers, analysis and data-reformatting scripts; (d) assistance in analysing and interpreting results. We did not use generative AI to generate synthetic datasets, formulate mathematical claims, or write proofs; translation and qualitative or thematic analysis are not applicable. Uses in the recommended-disclosure categories: literature search and discovery, drafting sections of the paper, producing figures via code, and editing for readability. The LLMs inside the experiments (the Qwen3.5-27B agent, the Qwen3-32B user simulator and summariser, and the Llama-3.3-70B-Instruct pilot agent) are objects of study, not authoring tools; the simulator's synthetic user turns are experimental inputs, not ground truth. All reported numbers are produced by deterministic scripts over logged trajectories, and constraint adherence is scored by regex and string checkers with no LLM judge. AI-generated code was reviewed and tested by the authors, and AI-assisted text and citations were checked against sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The anonymized supplementary material includes the evaluation harness, constraint catalogue and checkers, analysis scripts, and logged main-sweep trajectories. Section 3 specifies substrates, compression methods, trigger policy, constraint catalogue, redundancy bands, models, and run protocol; Appendices C and F report the post-hoc arms, per-cell numbers, and diagnostics.

REFERENCES

Anthropic. Automatic context compaction — Claude cookbook. Anthropic Developer Documentation, 2026a. URL https://platform.claude.com/cookbook/tool-use-automatic-context-compaction. Online; accessed 2026-05-05.

Anthropic. Compaction — Claude API documentation. Anthropic Developer Documentation, 2026b. URL https://platform.claude.com/docs/en/build-with-claude/compaction. Online; accessed 2026-05-05.

Anthropic. Context editing — Claude API documentation. Anthropic Developer Documentation, 2026c. URL https://platform.claude.com/docs/en/build-with-claude/context-editing. Online; accessed 2026-05-06.

Rahul Baxi. Separating constraint compliance from semantic accuracy: A novel benchmark for evaluating instruction-following under compression. 2025. URL https://arxiv.org/abs/2512.17920.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *EMNLP*, 2023.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. 2025. URL https://arxiv.org/abs/2504.19413.

Pengfei Du. Memory for autonomous LLM agents: Mechanisms, evaluation, and emerging frontiers. 2026. URL https://arxiv.org/abs/2603.07670.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *ICLR*, 2024.

Google. Agent development kit (ADK): Context management and compaction. Google ADK Documentation, 2026. URL https://adk.dev/. Online; accessed 2026-05-05.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

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

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *ACL*, 2024. URL https://arxiv.org/abs/2310.06839.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *ICLR*, 2024. URL https://arxiv.org/abs/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. 2025. URL https://arxiv.org/abs/2510.00615.

Jinyi Li, Yihuai Lan, Lei Wang, and Hao Wang. PCToolkit: A unified plug-and-play prompt compression toolkit of large language models. 2024. URL https://arxiv.org/abs/2403.17411.

Miles Q. Li, Benjamin C. M. Fung, Martin Weiss, Pulei Xiong, Khalil Al-Hussaeni, and Claude Fachkha. A benchmark for evaluating outcome-driven constraint violations in autonomous AI agents. 2025. URL https://arxiv.org/abs/2512.20798.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. MemPO: Self-memory policy optimization for long-horizon agents. 2026. URL https://arxiv.org/abs/2603.00680.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. In *EMNLP*, 2023. URL https://arxiv.org/abs/2310.06201.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 2024.

Meta AI. Llama 3.3 70B Instruct model card. Hugging Face model card, 2024. URL https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. 2023. URL https://arxiv.org/abs/2311.12983.

OpenAI. Introducing SWE-bench Verified. OpenAI Blog, August 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Online; accessed 2026-05-05.

OpenAI. Responses API: Server-side context compaction. OpenAI Platform Documentation, 2025. URL https://developers.openai.com/api/reference/resources/responses/methods/compact. Online; accessed 2026-05-06.

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

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, et al. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *ACL Findings*, 2024.

Qwen Team. Qwen3.5-27B: Dense multimodal foundation model. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-27B. Online; accessed 2026-05-05. Predecessor series report: Qwen3 Technical Report, arXiv:2505.09388.

Prithvi Rajasekaran, Ethan Dixon, Carly Ryan, Jeremy Hadfield, Rafi Ayub, Hannah Moran, Cal Rueb, and Connor Jennings. Effective context engineering for AI agents. Anthropic Engineering Blog, September 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents. Online; accessed 2026-05-05.

Reshabh K Sharma. ContextCov: Deriving and enforcing executable constraints from agent instruction files. 2026. URL https://arxiv.org/abs/2603.00822.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *ACL*, 2024.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. 2024. URL https://arxiv.org/abs/2410.10813.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. 2023. URL https://arxiv.org/abs/2309.17453.

Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. Reducing cost of LLM agents with trajectory reduction. 2025. URL https://arxiv.org/abs/2509.23586.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2404.07972.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. In *ICLR*, 2024. URL https://arxiv.org/abs/2310.04408.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. 2025. URL https://arxiv.org/abs/2502.12110.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025.

Junjie Ye, Guoqiang Zhang, Wenjie Fu, Zelin Li, Tao Gui, Qi Zhang, and Xuanjing Huang. CCTU: A benchmark for tool use under complex constraints. 2026. URL https://arxiv.org/abs/2603.15309.

Weihao Zeng, Yuzhen Huang, and Junxian He. LOCA-bench: Benchmarking language agents under controllable and extreme context growth. 2026. URL https://arxiv.org/abs/2602.07962.

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. URL https://arxiv.org/abs/2307.13854.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. 2025. URL https://arxiv.org/abs/2506.15841.

A WORKED EXAMPLE

Figure 3 shows, inside one trajectory, the split reported in Section 4.1: the compaction removes the user message that states two task-external wrappers, both are violated, and the trajectory-wide prohibition survives.

Worked example (τ-bench task *test_72*, θ=0.70, truncation; pilot sweep with a Qwen3-32B agent using the same harness, catalogue, and injection).
Planted in the first user message: "Please follow these constraints throughout this conversation: – Wrap your final answer in `<answer>...</answer>` tags. – Never use the word 'unfortunately' in any response. – Always end your final response with the marker `[DONE]`."
Before the turn-9 compaction: 20 messages; the constraint block opens the history.
After: 7 messages; the block is gone and the history resumes mid-task ("The user is trying to modify their order...").
Final output ends in a raw tool call (`...paypal_7729105"}}}`): C1 (`<answer>`, Once band) and C6 (`[DONE]`, Periodic band) are violated; C4 (prohibition, Twice band) is satisfied. The task itself was not solved (reward 0).

Figure 3: One trajectory's view of Figure 1: truncation removes the user message stating two task-external wrappers and both are violated; the trajectory-wide prohibition survives.

B COMPRESSION MECHANICS

The trajectory-support split says *what* erodes; why do the two substrates give compression such different room to act? The explanation is mechanical. τ-bench compaction events recover only 19% of pre-event tokens on average (mean post/pre ratio 81%), with a fifth percentile of 3,546 tokens after truncation because the protected system prompt and tool schemas cannot be removed; AppWorld has no comparable static prefix, and compaction drives utilization from 66% to 19% (mean ratio 29%). The 81% ratio is a property of the protocol, not of either method (per-event mechanics: Figure 5 and Appendix F). Two properties are consistent with the method-ranking difference of Section 4.3: on τ-bench, summarization carries tool traffic over verbatim, so with the schema floor it frees little context (mean post/pre 85% vs. 78% for truncation); on AppWorld, `world.execute` runs in a stateful IPython session whose later cells reference state defined earlier, so truncation drops state-bearing artefacts a later cell may need while summarization keeps the gist. The present data do not identify the relative contribution of these mechanisms.

C POST-HOC ROBUSTNESS ARMS

All arms in Table 1 were run after the main sweep on one fixed τ-bench task subset: 40 retail tasks drawn with a fixed seed, stratified by ground-truth action count, identical across arms, one repeat per cell (360 trajectories). Every arm uses $\theta = 0.50$, $W = 8{,}000$, and Qwen3-32B as summariser and user simulator; only the listed knobs and the agent model change. The main sweep used a larger, differently drawn task sample (74 tasks $\times$ 2 repeats), so cross-run deltas are indicative rather than paired. Per-constraint denominators are 12–24 trajectories because each trajectory plants three of the seven catalogue constraints. The *gap* column is mean(C2, C3) $-$ mean(C1, C6) in percentage points.

D POSITIONING AGAINST PRIOR WORK

Table 2 positions the closest prior work discussed in Section 2 along the dimensions this study varies (compression timing, number of methods compared, and whether instruction constraints are tracked).

E REDUNDANCY-BAND FIGURE

Figure 4 plots the per-band satisfaction rates discussed in Section 4.5; denominators are in Table 7.

Table 1: Post-hoc sensitivity (rows 1–6, Qwen3.5-27B agent) and second-model-family (rows 7–9, Llama-3.3-70B-Instruct agent) arms. Main-sweep reference values: baseline success 51.9%, adherence 93.6%; $t50$*_trun* 34.6% / 59.2%; $t50$*_summ* 27.2% / 63.2%; C1 30%, C6 40%, C2 99%, C3 100% pooled over all τ-bench cells (gap +64.5 pp).

Arm	n	Success	Adherence	Compact./traj	C1	C6	C2	C3	Gap (pp)
Agent Qwen3.5-27B, $\theta = 0.50$; target / cooldown varied									
target $0.75\,W$, cooldown 3, summ.	40	45.0%	75.0%	3.40	8.3%	61.1%	100%	100%	+65.3
target $0.75\,W$, cooldown 3, trunc.	40	65.0%	92.5%	3.02	66.7%	88.9%	100%	100%	+22.2
target $0.30\,W$, cooldown 1, summ.	40	5.0%	46.7%	24.35	5.3%	38.9%	100%	100%	+77.9
target $0.30\,W$, cooldown 1, trunc.	40	32.5%	59.2%	14.50	0.0%	0.0%	100%	100%	+100.0
target $0.30\,W$, cooldown 6, summ.	40	50.0%	72.5%	1.93	23.1%	21.1%	100%	100%	+77.9
target $0.30\,W$, cooldown 6, trunc.	40	47.5%	73.3%	2.12	30.0%	38.1%	100%	100%	+66.0
Agent Llama-3.3-70B-Instruct; main-sweep target $0.30\,W$ and cooldown 3									
baseline (no compression)	39	23.1%	94.9%	0.00	89.5%	86.7%	100%	100%	+11.9
$t50$*_summ*	39	28.2%	73.5%	5.00	57.9%	17.6%	100%	100%	+62.2
$t50$*_trun*	40	12.5%	77.5%	2.35	58.8%	42.9%	100%	100%	+49.2

Table 2: Closest prior work positioned along the dimensions varied here. "Methods" counts compression algorithms compared. The column entries summarise dimensions varied in the cited work, not a quality judgement.

Reference	Timing	Methods	Constraints	Setting
ACON (Kang et al., 2025)	2 levels	1	no	multi-turn
MEM1 (Zhou et al., 2025)	every turn	1 (learned)	no	multi-turn
MemGPT (Packer et al., 2023)	event-driven	1 (paging)	no	multi-session
CDCT (Baxi, 2025)	static (rate sweep)	multiple	yes	single-turn
ODCV (Li et al., 2025)	fixed	—	yes (outcome)	multi-turn
CCTU (Ye et al., 2026)	fixed	—	yes (tool-use)	multi-turn
ContextCov (Sharma, 2026)	runtime (orthogonal)	—	yes (executable)	multi-turn
LOCA-bench (Zeng et al., 2026)	fixed	6	no (binary success)	multi-turn
This work	3 (+baseline)	2	yes (7, mechanical)	multi-turn

F SANITY CHECKS

Table 3: Constraint catalogue. "Both" = scored on both substrates; τ = τ-bench only. *C1/C6 were also planted on AppWorld but cannot be satisfied there and are excluded from every AppWorld rate.

ID	Surface tag	Description	Subs.
C1	format	Wrap final answer in `<answer>...</answer>`	τ^*
C2	format	Round numerics to two decimals	both
C3	format	snake_case JSON keys	both
C4	prohibition	Do not use the word "unfortunately"	both
C5	count	At most N tool calls per trajectory	both
C6	format	End final response with `[DONE]`	τ^*
C7	value	Address user as `{title}` (*Sir/Madam/friend*)	τ

Baseline-no-compression invariant. Both substrates' baseline cells recorded zero compaction events, validating that the trigger is correctly disabled.

Compression monotonicity. Mean compactions per trajectory are monotone in the trigger threshold:

Substrate	$\theta = 0.50$	$\theta = 0.70$	$\theta = 0.90$
τ-bench	4.60	2.85	1.15
AppWorld	3.75	1.28	0.60

Token-count drift. Pre-call estimator vs. post-call OpenRouter usage:

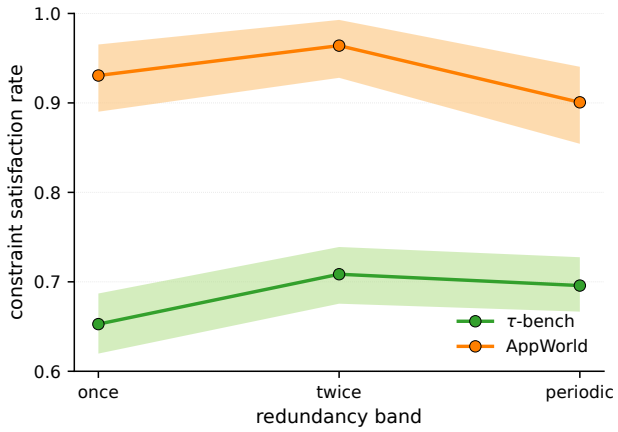


Figure 4: Constraint satisfaction by redundancy band, pooled over the six compressed cells (C7 included; AppWorld C1/C6 excluded as unsatisfiable). Lines are point estimates; shaded regions are 95% cluster-bootstrap CIs over trajectories. Bands operate within the truncatable region only; the system prompt is never modified. Bands are intention-to-treat (Section 4.5). The τ-bench Once-to-Twice lift is the only well-resolved band effect.

Substrate	n samples	mean signed drift	p95 |drift|	passes $\leq 10\%$?
τ-bench	13,504	-16.7%	13.8%	no
AppWorld	4,262	$+3.4\%$	9.2%	yes

Per-task vs. pooled task-success rates. Per-task-equal-weighted rates differ from pooled rates by at most 2.4 pp on τ-bench and 2.3 pp on AppWorld:

	τ-bench		AppWorld	
Cell	pooled	per-task	pooled	per-task
baseline	51.9%	54.2%	44.4%	44.7%
t50_summ	27.2%	28.3%	37.1%	39.5%
t50_trun	34.6%	34.0%	10.8%	10.5%
t70_summ	33.1%	32.4%	30.6%	28.9%
t70_trun	33.3%	33.8%	25.7%	23.7%
t90_summ	35.6%	34.9%	41.7%	39.5%
t90_trun	46.3%	45.3%	36.1%	36.8%

Power-matched cross-substrate sub-sampling. Sub-sampling τ-bench cells to $n = 36$ (matched to AppWorld; single random seed) keeps the baseline highest and *t50_summ* lowest, but middle cells reorder; the maximum absolute difference between full-n and $n = 36$ τ-bench task-success rates is 0.089. The pooled method ordering (truncation above summarization) and the task-external-wrapper loss (C1 77%→22%, C6 88%→39%, baseline→compressed) survive sub-sampling.

Repeats ICC. One-way ANOVA-based intra-class correlation for the two repeats per (task, cell): τ-bench task-success ICC $= 0.468$ (moderate); AppWorld $= 0.732$ (> 0.7). The AppWorld value implies the two repeats are not independent observations: effective n per AppWorld cell is about 21 rather than 36, and constraint-rate CIs are widened accordingly. τ-bench cluster-robust SEs by (task, cell) are sufficient.

AppWorld outcome sensitivity. Table 9 gives best/worst-case task-success bounds for slots without an evaluable outcome. The headline AppWorld truncation result remains a large decline under either bound.

Band denominators. Each τ-bench trajectory contributes one scorable observation per band. AppWorld has 201/163/177 scorable observations (Once/Twice/Periodic), with 22–30 per cell-band after excluding C1/C6, which cannot be satisfied on this substrate. Table 7.

Compression mechanics.

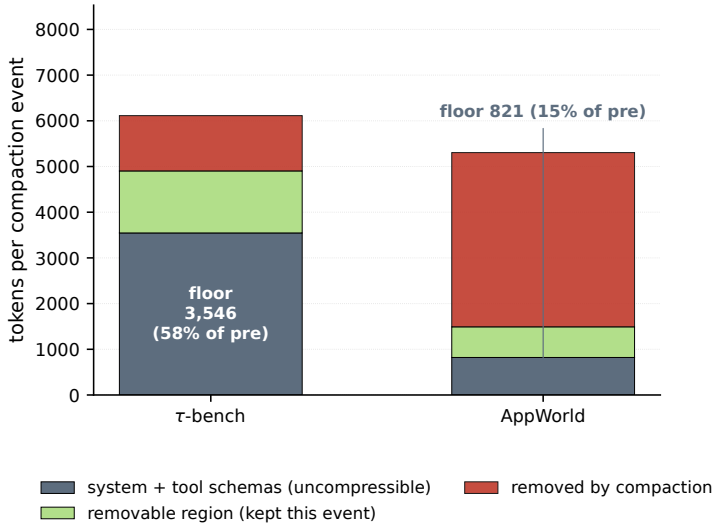


Figure 5: Mean tokens per compaction event, decomposed using a system-prompt-plus-schemas *floor proxy* (5th-percentile post-truncation tokens), *kept above floor*, and *removed by this event*. τ-bench's 3,546-token proxy occupies 58% of a pre-compaction state and 84% of a post-truncation state; AppWorld's 821-token proxy is 15% / 60%.

Per-event mechanics	τ-bench	AppWorld
Compaction events (summ. / trun.)	1,162 / 1,081	159 / 245
Mean tokens pre / post	6,113 / 4,903	5,297 / 1,495
Floor proxy (5th-pctile post-trun.)	3,546	821
Mean compression ratio (post/pre)	81%	29%
Mean utilization pre / post	76% / 61%	66% / 19%

Uncached agent-input accounting: per-cell numbers.

	τ-bench		AppWorld	
Cell	input/traj	Δ	input/traj	Δ
baseline	71.2 K	—	56.4 K	—
$t90_trun$	80.4 K	+13.0%	54.1 K	−4.0%
$t90_summ$	85.1 K	+19.5%	52.2 K	−7.3%
$t70_trun$	90.1 K	+26.6%	60.0 K	+6.5%
$t70_summ$	100.9 K	**+41.7%**	39.6 K	−29.7%
$t50_trun$	90.6 K	+27.2%	44.3 K	−21.5%
$t50_summ$	90.3 K	+26.8%	37.0 K	**−34.3%**

Table 4: Exploratory fixed-effect estimates from the mixed-effects logistic regression on constraint satisfaction. Model: `satisfied ~ method × threshold_c × band + substrate` with random intercepts for task and trajectory. $\texttt{threshold}_c$ is centred and scaled (`(threshold - 0.7)/0.2`); reference levels: method = *none* (baseline), band = *once*, substrate = τ-bench. Estimates are posterior means and 95% credible intervals from BINOMIALBAYESMIXEDGLM; we use this table as a qualitative diagnostic, not a confirmatory model.

Term	$\hat{\beta}$	95% CI	OR	95% CI
Intercept	+0.964	[+0.887, +1.042]	2.623	[2.427, 2.835]
method=summarization	+0.045	[-0.071, +0.161]	1.046	[0.931, 1.175]
method=truncation	-0.203	[-0.315, -0.090]	0.817	[0.729, 0.914]
band=twice	+0.028	[-0.107, +0.164]	1.029	[0.899, 1.178]
band=periodic	+0.095	[-0.040, +0.230]	1.100	[0.961, 1.259]
substrate=appworld	-0.197	[-0.361, -0.033]	0.821	[0.697, 0.968]
method=summarization:band=twice	+0.012	[-0.191, +0.215]	1.012	[0.826, 1.240]
method=truncation:band=twice	+0.112	[-0.085, +0.308]	1.118	[0.919, 1.361]
method=summarization:band=periodic	-0.165	[-0.364, +0.034]	0.848	[0.695, 1.034]
method=truncation:band=periodic	+0.066	[-0.131, +0.263]	1.068	[0.877, 1.301]
thr	+0.838	[+0.750, +0.925]	2.311	[2.118, 2.521]
method=summarization:thr	-0.529	[-0.672, -0.387]	0.589	[0.511, 0.679]
method=truncation:thr	-0.372	[-0.511, -0.233]	0.689	[0.600, 0.792]
thr:band=twice	-0.041	[-0.192, +0.109]	0.960	[0.825, 1.115]
thr:band=periodic	+0.075	[-0.077, +0.228]	1.078	[0.926, 1.256]
method=summarization:thr:band=twice	+0.093	[-0.157, +0.342]	1.097	[0.855, 1.408]
method=truncation:thr:band=twice	+0.014	[-0.227, +0.255]	1.014	[0.797, 1.291]
method=summarization:thr:band=periodic	-0.127	[-0.371, +0.117]	0.881	[0.690, 1.124]
method=truncation:thr:band=periodic	-0.099	[-0.342, +0.143]	0.905	[0.711, 1.154]

Table 5: Exploratory cell × constraint contrasts vs. no-compression baseline, with Benjamini–Hochberg FDR control at $\alpha = 0.05$ over the nominal family of $K = 78$ tests. The family includes 12 AppWorld C1/C6 contrasts, which are omitted here: C1/C6 cannot be satisfied on AppWorld and score 0% in every cell, baseline included. Cell-level satisfaction rates are reported as proportions; the p-value is from a two-sided two-proportion z-test against the baseline rate within the same substrate and does not model repeated tasks or paired constraint assignment. The final column flags contrasts that survive this diagnostic FDR screen.

Substrate	Cell	C#	base	cell	Δ	p	q	FDR
appworld	t50_summ	C2	0.952	1.000	+0.048	0.3231	0.6466	
appworld	t50_summ	C3	1.000	1.000	+0.000	–	1.0000	
appworld	t50_summ	C4	1.000	1.000	+0.000	–	1.0000	
appworld	t50_summ	C5	0.950	0.526	-0.424	0.0025	0.0087	✓
appworld	t50_trun	C2	0.952	1.000	+0.048	0.3115	0.6466	
appworld	t50_trun	C3	1.000	1.000	+0.000	–	1.0000	
appworld	t50_trun	C4	1.000	1.000	+0.000	–	1.0000	
appworld	t50_trun	C5	0.950	0.429	-0.521	0.0003	0.0014	✓
appworld	t70_summ	C2	0.952	0.952	+0.000	1.0000	1.0000	
appworld	t70_summ	C3	1.000	1.000	+0.000	–	1.0000	
appworld	t70_summ	C4	1.000	1.000	+0.000	–	1.0000	
appworld	t70_summ	C5	0.950	0.900	-0.050	0.5483	1.0000	
appworld	t70_trun	C2	0.952	0.950	-0.002	0.9718	1.0000	
appworld	t70_trun	C3	1.000	1.000	+0.000	–	1.0000	
appworld	t70_trun	C4	1.000	1.000	+0.000	–	1.0000	
appworld	t70_trun	C5	0.950	0.684	-0.266	0.0306	0.0995	
appworld	t90_summ	C2	0.952	1.000	+0.048	0.3115	0.6466	
appworld	t90_summ	C3	1.000	1.000	+0.000	–	1.0000	
appworld	t90_summ	C4	1.000	1.000	+0.000	–	1.0000	
appworld	t90_summ	C5	0.950	1.000	+0.050	0.3112	0.6466	
appworld	t90_trun	C2	0.952	1.000	+0.048	0.3231	0.6466	
appworld	t90_trun	C3	1.000	1.000	+0.000	–	1.0000	
appworld	t90_trun	C4	1.000	1.000	+0.000	–	1.0000	
appworld	t90_trun	C5	0.950	0.950	+0.000	1.0000	1.0000	

Table 6: BH-FDR contrasts on τ-bench (continuation of Table 5).

Substrate	Cell	C#	base	cell	Δ	p	q	FDR
tau_bench	t50_summ	C1	0.815	0.097	-0.719	4.8e-16	1.9e-14	✓
tau_bench	t50_summ	C2	1.000	0.979	-0.021	0.2859	0.6466	
tau_bench	t50_summ	C3	1.000	1.000	+0.000	–	1.0000	
tau_bench	t50_summ	C4	0.868	0.887	+0.019	0.7672	1.0000	
tau_bench	t50_summ	C5	1.000	0.902	-0.098	0.0146	0.0496	✓
tau_bench	t50_summ	C6	0.870	0.143	-0.728	1.5e-13	2.4e-12	✓
tau_bench	t50_summ	C7	1.000	0.491	-0.509	2.8e-10	2.2e-09	✓
tau_bench	t50_trun	C1	0.815	0.152	-0.664	2.9e-14	6.7e-13	✓
tau_bench	t50_trun	C2	1.000	0.981	-0.019	0.3150	0.6466	
tau_bench	t50_trun	C3	1.000	1.000	+0.000	–	1.0000	
tau_bench	t50_trun	C4	0.868	0.796	-0.072	0.3220	0.6466	
tau_bench	t50_trun	C5	1.000	0.849	-0.151	0.0021	0.0079	✓
tau_bench	t50_trun	C6	0.870	0.185	-0.685	9.9e-13	9.7e-12	✓
tau_bench	t50_trun	C7	1.000	0.246	-0.754	4.1e-17	3.2e-15	✓
tau_bench	t70_summ	C1	0.815	0.164	-0.651	2.7e-13	3.0e-12	✓
tau_bench	t70_summ	C2	1.000	1.000	+0.000	–	1.0000	
tau_bench	t70_summ	C3	1.000	1.000	+0.000	–	1.0000	
tau_bench	t70_summ	C4	0.868	0.926	+0.058	0.3233	0.6466	
tau_bench	t70_summ	C5	1.000	0.925	-0.075	0.0331	0.1032	
tau_bench	t70_summ	C6	0.870	0.277	-0.594	1.4e-09	8.8e-09	✓
tau_bench	t70_summ	C7	1.000	0.648	-0.352	5.9e-07	3.5e-06	✓
tau_bench	t70_trun	C1	0.815	0.179	-0.636	2.7e-13	3.0e-12	✓
tau_bench	t70_trun	C2	1.000	1.000	+0.000	–	1.0000	
tau_bench	t70_trun	C3	1.000	1.000	+0.000	–	1.0000	
tau_bench	t70_trun	C4	0.868	0.926	+0.058	0.3233	0.6466	
tau_bench	t70_trun	C5	1.000	0.827	-0.173	0.0009	0.0037	✓
tau_bench	t70_trun	C6	0.870	0.218	-0.652	8.4e-12	7.3e-11	✓
tau_bench	t70_trun	C7	1.000	0.339	-0.661	3.4e-14	6.7e-13	✓
tau_bench	t90_summ	C1	0.815	0.284	-0.532	8.5e-10	6.1e-09	✓
tau_bench	t90_summ	C2	1.000	0.980	-0.020	0.3057	0.6466	
tau_bench	t90_summ	C3	1.000	1.000	+0.000	–	1.0000	
tau_bench	t90_summ	C4	0.868	0.887	+0.019	0.7672	1.0000	
tau_bench	t90_summ	C5	1.000	0.964	-0.036	0.1465	0.4232	
tau_bench	t90_summ	C6	0.870	0.547	-0.323	0.0002	0.0010	✓
tau_bench	t90_summ	C7	1.000	0.695	-0.305	4.1e-06	2.2e-05	✓
tau_bench	t90_trun	C1	0.815	0.422	-0.394	4.1e-06	2.2e-05	✓
tau_bench	t90_trun	C2	1.000	1.000	+0.000	–	1.0000	
tau_bench	t90_trun	C3	1.000	1.000	+0.000	–	1.0000	
tau_bench	t90_trun	C4	0.868	0.946	+0.079	0.1559	0.4342	
tau_bench	t90_trun	C5	1.000	0.927	-0.073	0.0365	0.1095	
tau_bench	t90_trun	C6	0.870	0.527	-0.343	9.7e-05	0.0004	✓
tau_bench	t90_trun	C7	1.000	0.745	-0.255	3.5e-05	0.0002	✓

Table 7: Band denominators. **Top:** substrate × band totals: trajectories contributing a scorable constraint observation in that band, and total constraint observations (one per trajectory and band). AppWorld counts exclude constraints C1/C6, which cannot be satisfied there. **Bottom:** per-cell breakdown for the appendix.

Substrate	Band	#trajectories	#observations
appworld	once	201	201
appworld	twice	163	163
appworld	periodic	177	177
tau_bench	once	924	924
tau_bench	twice	924	924
tau_bench	periodic	924	924

Table 8: Per-cell band breakdown (appendix detail for Table 7; AppWorld excludes C1/C6).

Substrate	Cell	Band	#trajectories	#observations
appworld	baseline	once	28	28
appworld	baseline	twice	24	24
appworld	baseline	periodic	26	26
appworld	t50_summ	once	28	28
appworld	t50_summ	twice	22	22
appworld	t50_summ	periodic	25	25
appworld	t50_trun	once	30	30
appworld	t50_trun	twice	24	24
appworld	t50_trun	periodic	26	26
appworld	t70_summ	once	29	29
appworld	t70_summ	twice	24	24
appworld	t70_summ	periodic	25	25
appworld	t70_trun	once	28	28
appworld	t70_trun	twice	22	22
appworld	t70_trun	periodic	25	25
appworld	t90_summ	once	29	29
appworld	t90_summ	twice	24	24
appworld	t90_summ	periodic	25	25
appworld	t90_trun	once	29	29
appworld	t90_trun	twice	23	23
appworld	t90_trun	periodic	25	25
tau_bench	baseline	once	135	135
tau_bench	baseline	twice	135	135
tau_bench	baseline	periodic	135	135
tau_bench	t50_summ	once	125	125
tau_bench	t50_summ	twice	125	125
tau_bench	t50_summ	periodic	125	125
tau_bench	t50_trun	once	133	133
tau_bench	t50_trun	twice	133	133
tau_bench	t50_trun	periodic	133	133
tau_bench	t70_summ	once	130	130
tau_bench	t70_summ	twice	130	130
tau_bench	t70_summ	periodic	130	130
tau_bench	t70_trun	once	132	132
tau_bench	t70_trun	twice	132	132
tau_bench	t70_trun	periodic	132	132
tau_bench	t90_summ	once	135	135
tau_bench	t90_summ	twice	135	135
tau_bench	t90_summ	periodic	135	135
tau_bench	t90_trun	once	134	134
tau_bench	t90_trun	twice	134	134
tau_bench	t90_trun	periodic	134	134

Table 9: AppWorld task-success rates among completed trajectories, with sensitivity bounds under best/worst-case outcomes for remaining slots.

Cell	Expected	Clean	Err.	Unatt.	Rate (clean)	Bounds (excl. unatt.)	Bounds (incl. unatt.)
baseline	40	36	2	2	0.444	[0.421, 0.474]	[0.400, 0.500]
t50_summ	40	35	4	1	0.371	[0.333, 0.436]	[0.325, 0.450]
t50_trun	40	37	2	1	0.108	[0.103, 0.154]	[0.100, 0.175]
t70_summ	40	36	2	2	0.306	[0.289, 0.342]	[0.275, 0.375]
t70_trun	40	35	4	1	0.257	[0.231, 0.333]	[0.225, 0.350]
t90_summ	40	36	2	2	0.417	[0.395, 0.447]	[0.375, 0.475]
t90_trun	40	36	3	1	0.361	[0.333, 0.410]	[0.325, 0.425]