Harness Engineering for Multi-Agent Systems: State, Memory, and Tool Dispatch in Agentic Commerce

Anonymous ACL submission

Abstract

Reliable multi-agent systems repeatedly depend on the same decisions about state composition, memory at phase boundaries, and tool dispatch, yet these are usually treated as framework mechanics rather than empirical design variables. We treat the engineering substrate beneath the agents — the *harness* — as the object of study and factor it into three layers: **L1 State** (typed channel composition and checkpoint scope), **L2 Memory** (working / episodic / semantic tiers with explicit forget and recall operators), and **L3 Tools** (uniform dispatch over LLMs, pretrained models, APIs, and human interrupts).

We instantiate this in COMMERCEHARNESS, a 31-node agentic commerce system, and evaluate it under both probe isolation and full-graph deployment. Our central finding is that L2 memory policy is *substrate-dependent*: an intent-keyed bridge (A6) ties full retention (A1) on a templated probe but strictly dominates A1 by $+10.7$ to $+25.9$ pp on the real graph across all five phase-2 model families, because the compact bridge avoids the multi-agent noise full retention must carry forward. We further show that L1 composition is a genuine storage–latency trade-off (nested saves $\sim$21 kB/scenario at $\sim 1.6\times$ stub-latency cost, with real-graph latency CI crossing zero), and that structured-output dispatch lifts downstream success from 15.6% to 86.7% as a commitment effect rather than a parsing fix. The result recasts harness design as an empirical systems problem rather than framework glue.

1 Introduction

Modern agent frameworks make it easy to wire LLMs, tools, and human interrupts into multi-agent workflows. What remains hard is deciding how state should be composed, what survives phase boundaries, and how heterogeneous tools should be dispatched so the resulting system is reliable enough to ship. These choices sit one level *above* the framework primitives and one level *below* model capabilities, yet they often determine whether an agent system behaves like a product or a demo.

The harness-engineering gap. "Harness engineering" as a named practice has recently gained traction in industry (OpenAI, 2026) and in research (Meng et al., 2026; Pan et al., 2026; Staufer et al., 2026), but the empirical question remains under-reported: which composition, memory, and dispatch choices actually matter once a system leaves the demo setting? Current agent-framework literature focuses on either (a) capability and tool-agent behaviour benchmarks (Liu et al., 2023; Yao et al., 2022a, 2024; Barres et al., 2025), (b) memory for long-horizon or multi-session reasoning (Packer et al., 2023; Xu et al., 2025; Maharana et al., 2024), or (c) prompting and context-engineering techniques (Chroma Research, 2025; Anthropic, 2025). A fourth axis — the *harness* that hosts agent capabilities, manages their memory, and dispatches their tools — is under-studied despite consuming much of the production effort. Treating memory as if it existed independently of the harness is especially misleading: in deployed systems, the write–manage–read loop is inseparable from how state is composed across phases and how tool calls are scheduled.

A three-layer architecture. We argue the harness factors into three layers, each with its own formal primitives and its own empirically testable trade-offs:

- **L1 State Composition** — how per-agent state schemas combine into a single typed-channel workflow state.
- **L2 Memory** — how information is retained, forgotten, and retrieved across phase transitions and across sessions, spanning working, episodic, and semantic tiers.

- **L3 Tools** — how heterogeneous callees (LLMs, pretrained non-LLM ranking models, external APIs, human interrupts) are dispatched behind a single interface.

Each layer rests on the one below: memory operators are applied to state channels; tool calls consume and mutate memory-resident context.

Why commerce. Commerce is a particularly demanding instantiation because all three layers are pushed into their non-trivial regime simultaneously. At L1, a multi-stakeholder workflow (cardholder, N merchant digital twins, payment processor, human) produces state schemas with genuine structural collisions—not accidental naming overlaps but semantically conflicting reducer logic across independent agents (§4.4). At L2, the payment-commit boundary is irreversible by design: τ fires at transaction, not user-turn, and the natural episodic store is a transaction *ledger* rather than a conversation log—which is precisely what motivates the structured–fuzzy bridge ρ. At L3, a pretrained ranking model cannot be replaced by an LLM call without changing the decision semantics, and HITL (Human in the Loop) at payment is a legal constraint, not a UX affordance. No layer degenerates to a trivial case.

Our setting. We instantiate this architecture in a multi-stakeholder commerce domain: a cardholder agent coordinating with simulated merchant agents (LLM-based digital twins of real merchant behaviour), scoring them with a pretrained transaction-sequence ranking model, applying coupons, invoking HITL payment approval, and persisting state for cross-session follow-ups. The workflow is inherently multi-phase (discovery $\rightarrow$ transaction $\rightarrow$ post-purchase), with hard transition boundaries at which memory management decisions become load-bearing.

Contributions.

- **A three-layer decomposition of agent harnesses.** We formalize harness design as state composition (L1), memory at phase boundaries (L2), and tool dispatch (L3), each expressed as operators over typed channel state rather than as framework-specific implementation detail (§4).

- **A harness-level formulation of phase-boundary memory.** We connect forget/recall operators to the working/episodic/semantic taxonomy while distinguishing multi-phase task workflows from multi-session conversational memory (§4.2).

- **A reference system, benchmark, and regime runner.** We release COMMERCE-HARNESS, a 31-node commerce harness, plus CommerceHarness-Bench and the tooling needed to isolate each layer for ablation (§5).

- **A substrate-dependent empirical result for memory policy.** Across a 1,350-rollout probe-harness L2 grid, a replayed 27-scenario full-graph study, LLM-aided inspection, and cross-framework probes, we show that managed memory is near-costless on clean templated context but becomes clearly beneficial once real upstream agent noise is present (§6, Appendix A).

- **Release of evaluation artifacts.** We release the benchmark, regime runner, rollout logs, inspection records, and supplementary experiment outputs in full at `https://sites.google.com/view/harness-engineering/`.

2 Related Work

Recent work names the *agent harness* as a research object. Meng et al. (2026)'s six-component matrix is the most direct precursor; we empirically instantiate three of the six (execution environment $\rightarrow$ L1, context management $\rightarrow$ L2, tool integration $\rightarrow$ L3). Pan et al. (2026) characterize the complementary *natural-language agent harness* (NLAH) class in which routing, memory, and tool dispatch are expressed in prompt space; our typed-channel design sits at the opposite end of the typedness axis. Staufer et al. (2026) document harness features of deployed systems, and industry practice uses "harness engineering" explicitly (OpenAI, 2026). We adopt the three-dimensional evaluation and LLM-aided log inspection of HAL (Kapoor et al., 2025). Memory work targeting multi-session conversations (MemGPT (Packer et al., 2023), A-MEM (Xu et al., 2025), LoCoMo (Maharana et al., 2024), surveys (Liu et al., 2025; Anonymous, 2026)) is complementary: we target multi-*phase* task workflows and express memory at the harness level rather than in prompt space. Context-engineering mitigations

(Chroma Research, 2025; Anthropic, 2025) (compaction, structured notes) are operationalized by our L2 as typed primitives. Tool-use benchmarks (Qin et al., 2023; Schick et al., 2023; Yao et al., 2022b, 2024; Barres et al., 2025) focus on single-agent LLM callees; we extend to a hybrid setting with a non-LLM ranking model wrapped as MCP. Commerce-agent work (Authors, 2026; Allouah et al., 2025; Sun et al., 2025) asks how well agents perform; we ask how to build the harness hosting them. Frameworks (LangGraph (LangChain AI, 2024), AutoGen (Wu et al., 2023), CrewAI (crewAI Inc., 2024)) provide the primitives; our contribution sits one level above, naming and characterizing the composition decisions practitioners make.

3 Preliminaries

An *agent harness* is a tuple $\mathcal{H} = (\mathcal{S}, G, \tau, \rho, \mathcal{T}, \pi)$: a typed channel state $\mathcal{S} = (c_1, \ldots, c_m)$ with per-channel reducers $\oplus_i$ (append for lists, overwrite for scalars); a workflow graph $G = (N, E)$ with nodes $f_n : \mathcal{S} \to \Delta$ composed via $\mathcal{S} \leftarrow \mathcal{S} \uplus f_n(\mathcal{S})$ and sequential, conditional, or parallel `Send` edges; a *teardown* operator $\tau : \mathcal{S} \to \mathcal{S}$ fired at phase transitions (discovery, transaction, post-purchase in our setting); a *bridge* retriever $\rho : \mathcal{S} \times \mathcal{H}_{\text{ep}} \to \Delta$ over a persistent episodic store $\mathcal{H}_{\text{ep}}$; a tool set $\mathcal{T}$ (LLM, non-LLM, REST, HITL); and a dispatch policy π. *Harness engineering* studies how choices in $(\mathcal{S}, \tau, \rho, \mathcal{T}, \pi)$ — distinct from weights or prompts — shape behavior; L1/L2/L3 (§4) map onto $\mathcal{S}$, $(\tau, \rho, \mathcal{H}_{\text{ep}})$, and $(\mathcal{T}, \pi)$. A *checkpointer* κ serializes $\mathcal{S}$ at node boundaries; a harness may use one top-level κ (flat) or a collection $\{\kappa_j\}$ scoped to subgraphs (nested). In our instantiation τ clears a working subset $C_\tau \subseteq \mathcal{S}$ and ρ reads a transaction ledger, returning an update that re-populates bridge channels.

4 A Three-Layer Architecture

We decompose the harness into three layers, each with its own formal primitives and each ablatable in isolation:

Layer	Concerns
L1 State	channel composition, reducer discipline, checkpointing scope
L2 Memory	working/episodic/semantic tiers, forget/recall at transitions
L3 Tools	LLM, ranking-model, external-API, and HITL dispatch

Each subsection below follows the same template: **Problem** → **Primitive** → **Formalization** → **When to use**.

4.1 L1 — State Composition

Given k logical sub-agents with per-agent state schemas S_i, the harness can (a) flatly compose $S = \bigoplus_i S_i$ into one typed state via TypedDict multiple inheritance, or (b) keep each S_i in a sub-graph with its own checkpointer and marshal across boundaries. In our reference implementation ($k = 5$), 12 of 57 unique channels collide across ≥ 2 schemas with identical reducers, and 4 appear in all five schemas — an overlap profile that makes nested isolation mostly redundant. Flat composition uses 1.64× the cumulative checkpoint bytes of nested (per-step serialization of the full union) but replaces five checkpointer instances and all cross-subgraph marshaling with a single checkpointer and direct channel access (§A.1). We recommend flat when (i) cross-schema overlap is $\gtrsim 20\%$, (ii) phase transitions must pass state across agents, and (iii) isolation is not required for tenancy or safety; otherwise retain subgraph isolation. The 20% threshold is derived from a single codebase (k=5 sub-agents); we treat it as an operational heuristic rather than a universal boundary pending replication on other multi-agent systems.

4.2 L2 — Memory

Mapping the working/episodic/semantic taxonomy (Liu et al., 2025; Anonymous, 2026) to typed channels yields: working = phase-local append channels (messages, plan, tool outputs); episodic = a persistent append history; semantic = external store hooks (mem0, MemoryOS). The failure mode we target is *not* forgetting facts across sessions but *contaminating* phase $k{+}1$ reasoning with phase-k working memory (Chroma Research, 2025). Two operators address it: $\tau(S)$ clears C_τ (the working subset) at phase transitions, and $\rho : S \times \mathcal{H} \to \Delta$ reads the persistent episodic store $\mathcal{H}$ and injects a single top-ranked record, scored as $\text{score}(h) = \sum_j w_j \cdot \text{sim}_j(\text{intent}_j, h.\text{field}_j)$ with fuzzy match over {merchant, service, goal}. They compose into four regimes we evaluate in §6: full-persistence (A1), amnesia (A3, τ only), managed-naive (A5, τ+most-recent), managed-intent (A6, τ+fuzzy). Unlike MemGPT's LLM-driven page-in/page-out (Packer et al., 2023), our operators fire deterministically at phase boundaries (§A.10 quantifies the provider-dependent reliability variance this re-

moves); unlike A-MEM's prompt-space Zettelkasten (Xu et al., 2025), they live at the harness level. Semantic memory is wired but disabled in the current reference implementation (§9). One consequence of retaining working memory is costly on reasoning-enabled checkpoints (`gemini-2.5-pro`): long context competes with the thinking-token budget, raising abstention. τ prevents that interaction by construction (§A.3).

4.3 L3 — Tools

Real-world dispatch must accommodate multi-provider LLMs, specialized non-LLM ranking/scoring models, REST APIs, and HITL interrupts under one interface. We use three primitives. **Structured-output-everywhere**: every LLM call feeding a downstream node is wrapped in `with_structured_output(Schema)`; prose is reserved for the user-facing channel. Prior tool-side work (Instructor, Outlines) established *how* to constrain LLM output to a schema; our contribution is a harness-level *policy* on *when* (every downstream-consumed call) that eliminates a class of "hedged-prose" commitment failures distinct from raw parsing failures (§A.5, Table 12). **Specialized MCP tools as first-class nodes**: a non-LLM model (in our case, a pretrained merchant-scoring model) is wrapped as an MCP stdio server and invoked with a node $n_{\text{tool}}(S)$ that returns the MCP result if it arrives within timeout T, else a deterministic LLM-only fallback. The tool is interchangeable: any ranking model with an input/output contract compatible with the node schema is a drop-in replacement. **Provider-abstracted dispatch**: $\sim$80 lines route per model-family prefix (OpenAI Responses API; REST proxy for Gemini). **HITL guardrails with typed triggers**: a `human_approval` node preempts dispatch when any trigger fires (payment-commit by default; Table 26). Structured-output and HITL jointly form the guardrail layer — the former blocks silent hedged-prose commitment, the latter is a hard pre-commit gate for irreversible actions.

4.4 Commerce-specific instantiation

Each layer takes a form that does not reduce to a generic single-agent-plus-tool harness. **L1 is multi-stakeholder**: the union state co-owns channels from four principals — cardholder (`cardholder_id`, `wallet`), merchant digital twins (`merchant_messages`), payment processor (`risk_events`, `approval_status`), and human (`hitl_decision`) — and flat composition lets the agent reason across all four without cross-subgraph marshaling. **L2 is ledger-keyed**: the episodic store's rows are {merchant, service_token, amount, applied_coupon, session_id}, not {utterance, turn}; ρ fuzzy-matches structured ledger fields, and τ discharges working memory only *after* a transaction's durable ledger record exists — the payment-commit boundary is what makes selective amnesia safe. **L3 is compliance-grade**: the structured record passed to `human_approval` is the regulatory audit trail.

5 Implementation: COMMERCEHARNESS

Figure 1 gives a condensed view of the reference graph: 31 nodes grouped into five logical agents (goal, plan, discovery/eval, transaction, post-purchase), with phase boundaries shown as dashed lines. This system class — a multi-phase commerce agent composing five logical sub-agents, one MCP-wrapped non-LLM ranking model, and HITL interrupts at payment — matches the commerce-agent pattern documented by Staufer et al. (2026)'s catalog of deployed agent systems, so our ablations speak to a broader class than a single codebase. Two classes of nodes are highlighted distinctly: the MCP-wrapped ranking-model node (`merchant_rerank`, Section 4.3) and the human-in-the-loop interrupt nodes (`human_approval`, Section 10). The `unified_finalization` node (dashed border) is where the L2 τ operator is applied at phase transitions.

The union state `UnifiedAgentState` composes five per-agent schemas (86 declarations / 57 unique channels; all 12 collisions have matching reducers). τ is applied at `unified_finalization` via `Overwrite([])` on five phase-local append-list channels; ρ is a weighted `rapidfuzz.WRatio` match over (merchant, service, goal) with weights 0.4/0.3/0.3 in the reference implementation; the probe-harness experiments of Section A.3 use a single-field merchant-name match at threshold 50 because the synthetic scenario ledger does not populate service or goal fields. `merchant_rerank` is a specialized MCP-tool node (stdio server, timeout + LLM-only fallback; transport overhead $\sim$1.1 ms); any ranking model matching its input/output contract is a drop-in substitute. The ablation harness is a thin CLI emitting per-rollout JSONL with full prompt/response/parsed-output/scored-

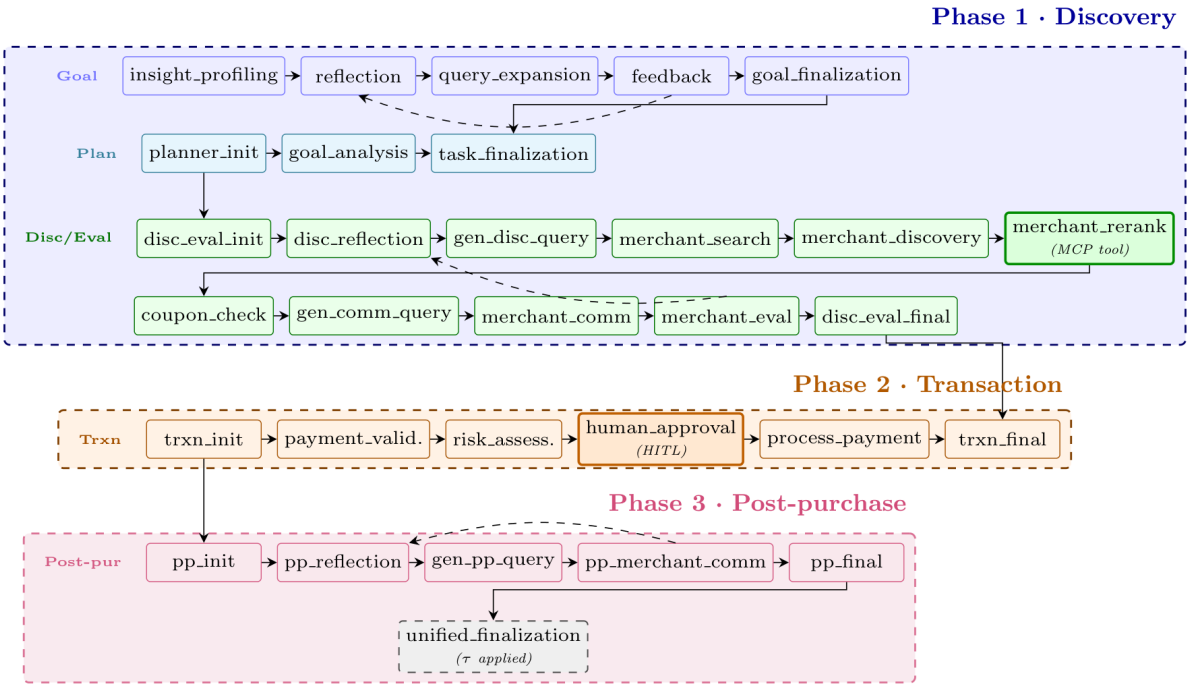

Figure 1: System overview. 31-node reference graph composed as a flat `StateGraph` over the union of five per-agent state classes (L1, §4.1). The three tinted, dashed panels — *Phase 1 · Discovery*, *Phase 2 · Transaction*, and *Phase 3 · Post-purchase* — mark the phase boundaries at which L2's τ is applied by `unified_finalization`. Coloured row labels (GOAL, PLAN, DISC/EVAL, TRXN, POST-PUR) identify the five logical sub-agents. Heavier-bordered nodes highlight the two non-LLM callees: `merchant_rerank` is a specialized MCP tool wrapping a ranking model (L3, §4.3), and `human_approval` is a deterministic HITL interrupt. Dashed arcs show reflection loops (goal feedback, discovery reflection, post-purchase reflection). Parallel `Send` fan-outs from `gen_disc_query`, `gen_comm_query`, and `gen_pp_query` are elided for readability. Source: `paper_fig1.tex` (standalone TikZ).

outcome; adding a new regime is ∼30 lines. Full channel inventory, code paths, and per-node type signatures are in Appendix C.

6 Experiments

6.1 Benchmark

CommerceHarness-Bench is a *controlled diagnostic benchmark for transaction grounding* (post-purchase reference resolution: given a follow-up utterance that references a prior transaction, identify the transaction) — not a broad benchmark for agentic commerce; its purpose is to separate harness-design effects from model capability. Each scenario specifies phases with ground-truth final state and distractor entities that should *not* influence reasoning. The release contains 25 scenarios (2–4 phases each): a 15-scenario base set (s01–s15, covering buy-then-refund, coupon reuse, risk-flagged dispute, ambiguous merchant name, cross-category follow-up, delayed follow-up, partial-info follow-up, item-level refund, long cross-session gap, and user-misremembered merchant) plus a 10-scenario adversarial expansion (s16–s25, covering three-way same-category distractors, near-homophone providers, user-side colloquialisms, multi-intent follow-ups, temporal references, vague-magnitude references, and paraphrased utterances with no explicit tokens). Full scenario definitions in Appendix D. Expansion to 40–50 scenarios is planned future work.

Two scenario sets, two substrates. CommerceHarness-Bench (25 scenarios, s01–s25) is the *probe-harness* benchmark: each scenario scripts the prior phases as templated messages so the final phase can be probed in isolation from upstream LLM variance (Table 8). A separate *full-graph* scenario set (27 scenarios from `experiments/l1_realgraph/scenarios/s1-s28`, omitting `s7_trxn_pizza` for a known recursion pathology) is used when we need the real 30-node cardholder agent to execute end-to-end (Tables 1, 2). The two benchmarks share the same task format (phase-1 commits a transaction, phase-2 asks a follow-up that references it) but

use disjoint scenarios authored for their respective substrates; the difference in scenario count between the two is a deliberate benchmark-design choice, not a sampling artifact.

6.2 Setup

Following the three-dimensional-analysis methodology of Kapoor et al. (2025) (model × scaffold × benchmark), we evaluate along (model × memory-regime × scenario) across three provider families. Model identifiers in this paper (`gpt-5.4-mini`, `gpt-5.4`, `claude-opus-4-7`, `gemini-3-flash-preview`, `gemini-2.5-pro`) are gateway-assigned names from our enterprise API gateway rather than public provider checkpoints; the exact underlying provider checkpoints and access timestamps are persisted with every released rollout record in `by_model/*/live_results.csv`. The primary L2 grid is the full 25-scenario CommerceHarness-Bench (s01–s15 base + s16–s25 adversarial expansion) run on every checkpoint. Per-checkpoint sample size: 25 scenarios × 4 regimes × 3 seeds = 300 rollouts for the three 4-regime models (`gpt-5.4-mini`, `gpt-5.4`, `claude-opus-4-7`) and 25 × 3 × 3 = 225 rollouts for the two Gemini checkpoints (A5 omitted). Totals across all five checkpoints: 1 350 primary-grid L2 rollouts. The inspector model for LLM-aided rollout inspection (Section A.8) is `gpt-5.4` regardless of the evaluated model. All 1 350 primary-grid rollouts and 240 inspection records are released in full (system prompt, sent messages, response, parse output, scored outcome, inspector flags) alongside the reproducibility bundle. Our 3-seeds-per-cell design corresponds directly to τ-bench's `pass^k` consistency metric (Yao et al., 2024) at k=3: a scenario counts as consistently solved only if the agent succeeds on all three independent seeds, which we report as a secondary column in Table 8's appendix version.

6.3 Metrics

Each rollout terminates when the model emits a strict-JSON response {`merchant`, `amount`, `txn_id`, `confidence`}; we score the `merchant` field by case-folded exact match against ground truth. We report: (i) **TSR** — fraction of rollouts committing the correct merchant in the JSON (not merely mentioning it in prose); (ii) **Bleed rate** — fraction whose response text contains any distractor merchant from a prior phase (scored over the full response, so a rollout can be TSR-correct-but-bleed-dirty); (iii) **Abstention** — fraction with `merchant` = null, tracked separately from *wrong* because a null answer is strictly preferable on safety-sensitive phases; (iv) **Phase Transition Success (PTS)**, $\Pr[\text{phase } k{+}1 \text{ succeeds} \mid \text{phase } k]$; (v) **Parsing Error Rate** on strict JSON validation; (vi) **Input-Token Cost**, the final-phase prompt tokens from the provider's usage field (our cross-regime cost proxy); (vii) **End-to-End Latency**, wall-clock seconds per final-phase call. TSR and bleed measure distinct failure modes — a regime can commit the right merchant in the structured record while surfacing a distractor in its reasoning — so our L2 comparisons require both to count as strictly dominant.

Table 1: L2 regimes on the real 30-node cardholder graph, `gpt-5.4-mini`. 14 scenarios × 4 regimes × 3 seeds = 168 rollouts, preceded by 14 deterministic replays of the full graph with real upstream LLM calls. A6 dominates A1 on TSR (+26.2 pp) and bleed-tied at 21.4%, with 90.3% fewer phase-2 prompt tokens (1927 → 187). A5 (naive most-recent bridge) is worse than both on this sub-grid (−24 pp vs. A1, 66.7% bleed) because its scenarios produce multi-row ledgers where "most recent" is usually a distractor surfaced by `merchant_rerank`, not the committed top pick — this A5 deficit does not reproduce on the expanded 27-scenario grid (Table 2) where most phase-1 ledgers are single-row, so A5 and A6 pick the same merchant. A3 collapses to 0%. One designed scenario (`s7_trxn_pizza`) hit the LangGraph recursion limit during recording and is excluded.

Regime	N	TSR	Abstain	Bleed	Prompt tok
A1 retain	42	45.2%	33.3%	21.4%	1927
A3 amnesia	42	0.0%	100%	0%	148
A5 recent br.	42	21.4%	11.9%	**66.7%**	182
A6 intent br.	42	**71.4%**	0%	21.4%	187

6.4 Headline results

Headline: on the real 30-node graph, A6 strictly dominates A1 on all five model families at 90% fewer tokens. Running the full cardholder agent (∼20 upstream LLM calls worth of plan, reflection, merchant search and rerank output) produces ∼1.9k tokens of pooled multi-agent context that A1 must digest. A1 TSR on `gpt-5.4-mini` drops to 45.2% with 33.3% abstention and 21.4% bleed; A6's compact single-row bridge record (187 tokens) reaches 71.4% TSR at matched bleed (Table 1). Paired cross-model on the 27-scenario grid (Table 2): A6 strictly beats A1 on all five phase-2 families with ΔTSR +10.7 to +25.9 pp, every

Table 2: Cross-model A6 − A1 on the full 30-node graph, scenario-paired cluster-bootstrap 95% CI (10k samples over scenarios). Phase-1 cache driven by `gpt-5.4` (27 scenarios × 3 seeds = 81 paired rollouts per cell; s29–s38 stress scenarios excluded). A6 strictly dominates A1 on all 5 phase-2 model families with CI entirely above zero. Compare the probe-harness numbers on the same models in Appendix A (Table 8): short templated phase-1 lets A1 ceiling-bind and A6 match-or-trail, whereas real-graph phase-1 floods A1 with multi-agent noise and A6's compact bridge dodges it. A5 tracks A6 on this naturalistic grid (95% CI on A6−A5 contains zero for every phase-2 model); see §A.13 for the constructed-ledger stress test that separates the two rules.

Phase-2 model	n	ΔTSR pp	95% CI
claude-opus-4-7	81	**+25.9**	[+11.1, +44.4]
gemini-2.5-pro	81	**+18.5**	[+6.2, +33.3]
gemini-3-flash-preview	81	**+18.5**	[+3.7, +33.3]
gpt-5.4	81	**+17.3**	[+4.9, +30.9]
gpt-5.4-mini	81	**+10.7**	[+1.3, +22.7]

95% CI excluding zero. A5 ties A6 on this naturalistic grid because single-row ledgers dominate, so "most recent" and "intent-matched" pick the same row; the 14-scenario sub-grid surfaces the exception where `merchant_rerank` exposes a distractor as the last committed row and A5 bleeds (66.7%, TSR 21.4%) while A6's fuzzy match stays correct. §A.13 isolates this A5-vs-A6 separation with a constructed multi-row ledger fixture.

Framework agnosticity. Table 2 uses a 400-LOC vanilla-Python orchestrator that replaces every piece of LangGraph's `StateGraph` runtime — node logic is untouched, only the runtime changes. Probe-harness transfer spans five frameworks (LangGraph, AutoGen, CrewAI, LlamaIndex, vanilla Python; Table 18) with $A_1 \geq A_6 \geq A_5 \gg A_3$ preserved on every framework.

6.5 Probe-harness L2 grid

The full-graph grid measures deployment-realistic behaviour; we pair it with an isolation study on a templated phase-1 that holds upstream quality constant so L2's effect can be read off directly (Table 3). A6 reaches 89–92% TSR across all five checkpoints at 3.4–4.2× fewer input tokens than A1; A3 collapses to $\leq 36\%$. The direction relative to A1 differs from the full-graph table: when phase-1 is clean and short, A1 ceiling-binds and A6 matches it (or strictly dominates on the thinking-enabled `gemini-2.5-pro`, +21.3 pp). The flip between Tables 2 and 3 is the finding: **A6's advantage grows with upstream noise, and real agent systems produce that noise.**

6.6 L1 composition and L3 structured outputs

L1 — flat vs. nested. On an 8-node stub, nested composition is on average ∼1.60× slower than flat at the median across $K = 5$ independent trials of 200 invocations (warmed-up mean ratio 1.60×, $\sigma = 0.05$; including a cold-start trial inflates the mean to 1.66×, $\sigma = 0.14$ and shows a single-trial outlier at 1.92× from first-touch import and compile-time caching). Replicated on the 30-node reference graph under deterministic LLM replay with a larger grid than originally reported (13 scenarios × 60 seeds, 780 paired comparisons), the effect shrinks to ∼3% pooled with CI crossing zero (−4.5 ms, 95% CI [−9.86, +1.03] ms). Of the 13 scenarios, only 1 shows a CI strictly excluding zero in favor of flat; no scenario shows a CI strictly excluding zero in favor of nested. Two of the three apparent strict-flat wins from the original 3-scenario, 30-seed grid flipped sign at $n = 60$, confirming they were small-sample artifacts. Checkpoint bytes, however, *are* significantly lower for nested on the full 25-scenario set: −21.3 kB / scenario (CI [−26.2, −17.7] kB, $n = 25$); the −32.6 kB figure originally reported was on a 5-scenario subset that overestimated the magnitude. L1 is therefore a genuine trade-off (storage vs. latency), not a Pareto-dominant choice: use flat when workflows are short-lived and debuggability matters, nested when cumulative checkpoint storage dominates.

L3 — structured vs. prose. Forcing structured JSON outputs at every node (rather than letting downstream consumers parse prose) lifts downstream TSR from 15.6% to 86.7% (Table 4, bottom). The gap is a *commitment* effect, not a parse artifact: the prose response is 100% recoverable by a lenient regex parser, but it frequently lists multiple merchant names the parser cannot disambiguate, whereas structured output forces commitment to a single answer before generation.

Remaining ablations. MCP transport adds ∼1.1 ms median per call, negligible versus LLM-call RTT (Appendix A, Table 32); five-framework transfer (LangGraph, AutoGen, CrewAI, LlamaIndex, vanilla Python) reproduces $A_1 \geq A_6 \geq A_5 \gg A_3$ on every framework tested (Table 18); SessionGap-Bench (§A.13) separates A5 and A6

Table 3: Probe-harness L2 grid, compact. 25 scenarios $\times$ 3 seeds per cell; A3/A6 shown (A5/A7 in Table 8). Tokens are final-phase prompt tokens.

Model	A1 TSR	A3 TSR	A6 TSR	A1 tokens	A6 tokens
`gpt-5.4-mini`	98.7%	10.7%	**92.0%**	890	261
`gpt-5.4`	88.0%	8.0%	**89.3%**	885	261
`claude-opus-4-7`	100%	4.0%	**92.0%**	1387	410
`gemini-3-flash-preview`	96.0%	36.0%	**92.0%**	887	251
`gemini-2.5-pro`	70.7%	32.0%	**92.0%**	987	257

Table 4: L1 trade-off (top) and L3 structured outputs (bottom). L1 stub: 200 invocations per variant per trial, identical node work, $K = 5$ independent trials; row reports the warmed-up mean ratio (trial 1 dropped as cold-start). L1 real graph: 30-node cardholder graph under deterministic replay, 13 scenarios $\times$ 60 seeds (780 paired rollouts), scenario-stratified bootstrap; n.s. = CI includes zero. L1 storage: full 25-scenario simulator (the originally reported 5-scenario subset overestimated the magnitude). L3: 90 rollouts on `gpt-5.4-mini`; TSR is under a lenient regex parser.

L1 — flat vs. nested composition	Flat median	Nested median	ΔCkpt bytes
stub (8 nodes, 200 invocations $\times K$=5 trials)	1.64 ms	2.62 ms (1.60$\times$, σ=0.05)	—
real graph (30 nodes, 13 scen $\times$ 60 seeds)	$\sim$3% slower pooled, CI $[-9.86, +1.03]$ ms (n.s.)		-21.3 kB/scen
L3 — structured vs. prose (`gpt-5.4-mini`)	Strict parse	TSR (lenient parser)	
structured (default)	**100%**	**86.7%**	
prose	0%	15.6%	

on multi-row ledgers, with the faithfulness caveat noted in §9.

7 Discussion

7.1 A decision rubric for practitioners

Appendix Table 5 gives the full per-layer decision rubric; three takeaways follow.

Three lessons. (i) L1 is a trade-off: flat $\sim 1.60\times$ faster on stubs (warmed-up mean of K=5 trials, σ=0.05), directionally so on the real graph but largely not significant (pooled $\sim$3%, CI $[-9.86, +1.03]$ ms over 13 scen $\times$ 60 seeds); nested saves $\sim$21.3 kB/scenario on the full 25-scenario set (CI $[-26.2, -17.7]$). (ii) L2's headline gain comes from *any* compact retriever (A5 or A6) beating full retention (A1) when upstream agent noise dominates — which retriever you pick matters only when the ledger has multi-row structure where "recent" and "intent-match" disagree; on naturalistic single-row ledgers the two rules are observationally tied, while on constructed multi-row ledgers the wrong retriever collapses TSR to amnesia-level (§A.13). (iii) The -71 pp prose gap is a commitment failure, not parse failure. Relative to τ-bench (Yao et al., 2024; Barres et al., 2025) we vary harness at fixed scenario, not user/tool at fixed harness.

Generalization. The mechanisms plausibly extend to any harness with (a) phase boundaries for τ, (b) a structured episodic store for ρ, and (c) downstream consumers where structured-output commitment is load-bearing, provided the bridge keys on fields users refer back to. Transfer on `SWE-bench` or MultiWOZ is future work.

Harness axes we do not treat. Three axes fit the $(\mathcal{S}, \tau, \rho, \mathcal{T}, \pi)$ formalism but are out of scope. **Agent evaluation** is partly present as methodology but not a first-class primitive (Kapoor et al., 2025; Sun et al., 2025). **Self-improvement**: Reflexion/Self-Refine/CoVe (Shinn et al., 2023; Madaan et al., 2023; Dhuliawala et al., 2024) add an inner loop rewriting $\mathcal{S}$; our graph has a `reflection` node but we do not ablate it. **Failure recovery** appears implicitly via MCP-timeout fallback, parse-failure retry, and HITL flags; a principled retry/fallback/escalate policy typed into π is a natural follow-up.

8 Conclusion

Harness design is an empirical systems problem. The core finding: **L2 memory policy is substrate-dependent** — an intent-keyed bridge (A6) ties full retention (A1) on a templated probe but beats A1 by $+10.7$–$+25.9$ pp on the real 30-node graph across all five phase-2 model families. The harness is a measurable subsystem, not framework glue.

9 Limitations

This paper is intended as a controlled systems case study of agent harness design, not as a universal evaluation of agent memory or agentic commerce. The main claims are therefore intentionally scoped: we show that harness choices are empirically consequential, that the L2 memory result depends on substrate, and that these effects are visible in a release-backed commerce system. The natural next question is how far that picture transfers across domains, model families, and longer-running deployments; the limitations below mark those boundaries.

Single-domain case study. All quantitative claims are grounded in a commerce workload. That is a deliberate choice because commerce makes phase boundaries, episodic ledger structure, and HITL dispatch unusually explicit, but it also means the paper should be read as evidence for a harness methodology, not as proof that the same effect size will hold in every agent domain. The AutoGen transfer probe is preliminary; broader cross-domain validation is future work.

Semantic memory is wired but disabled. Our L2 layer is presented with working, episodic, *and* semantic tiers, but the semantic tier (mem0 / MemoryOS hooks for cross-session user-preference abstraction) is deactivated in the reference implementation (`DISABLE_MEMORY=True`). Integration code exists and is exercised by unit tests, but we do not report numbers for a "managed+semantic" regime in this paper because we have not conducted the longitudinal study that would be needed to evaluate whether learned preference abstractions improve multi-session task success. We include the semantic tier in the taxonomy only to make the relationship to MemGPT/A-MEM explicit.

Benchmark is diagnostic, not comprehensive. CommerceHarness-Bench is a *controlled diagnostic benchmark for transaction grounding (post-purchase reference resolution)*: 25 multi-phase scenarios with explicit distractor merchants designed to isolate the harness effect. It is not meant to be a broad benchmark for agentic commerce. Its role in the paper is diagnostic: it gives us a controlled place to separate harness effects from model and scenario variance. It is small relative to LoCoMo's $\sim$50 conversations; scenarios are hand-authored against a 10-dimension test schedule (spelling variants, cross-category distractors, delayed follow-ups, etc.) rather than synthesised by a compositional generator. An expansion to 25–50 scenarios covering more ambiguous follow-ups, multi-intent follow-ups, adversarial distractors, user corrections, and multiple same-category transactions is planned; our current results should be read as diagnostic evidence for the harness decomposition rather than a definitive evaluation of commerce agents in the wild.

Ablations are one-at-a-time. We ablate each layer independently; combinatorial interactions (e.g., a flat state under amnesia vs. nested subgraphs under managed) are unexplored. The regime runner supports these sweeps; we report only the main effects in this paper for space.

L2 measurements are primarily on a probe harness. The L2 live-LLM grid (Table 8) uses templated prior phases and a single final-phase LLM call, not an end-to-end run of the 30-node cardholder graph. This is deliberate — it isolates τ/ρ from 25 upstream LLM calls' worth of variance. A 168-rollout full-graph replication on `gpt-5.4-mini` (§A.7, Table 1: 14 scenarios $\times$ 4 regimes $\times$ 3 seeds, preceded by 14 deterministic replays of the real graph) validates the direction of the L2 finding on full-graph context and shows a *stronger* A6 vs. A1 gap there (71.4% vs. 45.2% TSR; 90.3% token reduction). The full-graph A1 is markedly worse than the probe-harness A1 (97.8%) on the *same checkpoint* because real upstream traversal produces plan, reflection, and `merchant_rerank` messages that list 2–3 distractor merchants verbatim in the phase-1 message stream, whereas the templated probe context lists only a single prior target; A1's failure mode on the real graph is distractor commitment, not deliberation-budget competition (which is specific to thinking-enabled models). A6 retains a 26 pp TSR advantage but acquires 21.4% bleed at this scale — a retriever-quality effect our probe harness cannot reveal because it omits the service/goal fields the production retriever weights alongside merchant name (§A.9).

A5 vs. A6 comparison is bounded, not definitive. The naturalistic 27-scenario grid (Table 2) shows A5 $\approx$ A6 because phase-1 ledgers rarely exhibit multi-merchant structure. SessionGap-Bench (§A.13) engineers that structure explicitly, but by construction the most-recent ledger row is always a distractor — so A5 systematically picks the wrong row and A6 picks the intent-matched target, and

the resulting $+25$ to $+42$ pp A6$-$A5 gap primarily measures *whether the bridge channel propagates the retriever's choice to the phase-2 output*, not whether intent-matching beats recency on a realistic target distribution. A proper retrieval-rule comparison would require scenarios whose targets are labelled independently of either rule — ideally sampled from logged follow-up patterns so the intent-vs-recency mix reflects production traffic. We treat this as an open direction and do not claim a distributional advantage for A6 over A5 beyond the multi-row-ledger case where they structurally differ. The per-style decomposition in Table 20 partially addresses this: the *explicit-merchant* style (where "most-recent" can align with the intent target when the scenario permits) gives A5 a non-zero win rate, while *explicit-category* pins A5 to 0% by design — so the per-style gap, not the pooled average, is the more honest retriever-quality signal.

Context-rot effects narrower than literature. Bleed rate is zero on `gpt-5.4-mini` across all regimes, narrower than the structural distractor-count reduction would predict. We attribute this to the model's robustness at this phase count and defer a scaling sweep (more phases, open models) to future work.

Cross-family evaluation. We cover three frontier provider families (OpenAI, Anthropic, Google) but only one model per family outside OpenAI, and no open-weights models. Two additional families were targeted but not accessible through our enterprise gateway: xAI Grok (no route) and OpenAI's reasoning-tier `o3-deep-research` (RBAC-denied for our account). Open-weights evaluation (e.g. Llama, Qwen, Mistral) is future work, as is a sweep over multiple scales within each non-OpenAI family. We expect the direction of the L2 finding to transfer based on the cross-family rollouts reported here, the AutoGen probe (Section A.11), and prior-literature evidence that distractor interference persists across frontier model families (Chroma Research, 2025).

MCP-wrapped ranking model. Our L3 ranking-model pattern assumes the pretrained model is available at inference time. Cold-start, versioning, and model-update-as-rollout are not studied.

Run-to-run variance on absolute cell TSR. The full-graph numbers reported here rest on deterministic replay of LLM responses recorded once against live providers. Across three successive re-recordings of the same scenarios we observed absolute per-cell TSR drift of up to ± 10 pp, driven by (i) LLM non-determinism in phase-1 merchant selection (the cardholder graph occasionally commits to a different merchant on a fresh recording of the same user utterance), and (ii) schema drift in cached Pydantic responses when downstream model definitions evolve between recording and replay runs. Every regime-comparison CI in §A.12 and §A.13 is a within-run paired comparison, so the *direction* of A6 vs. A5 / A6 vs. A1 is stable across reruns; the absolute TSR per cell is not. We recommend reading ± 10 pp around any unpaired single-cell number as a run-variance envelope. This applies equally to the paper's primary-grid Table 8 numbers if they were ever re-recorded; we have not observed that directly but the mechanism is the same.

10 Ethical Considerations

Financial safety. In our reference implementation, every transaction-initiating node is guarded by a human-in-the-loop interrupt implemented via the L3 structured-output-plus-HITL pattern (Section 4.3): the agent can propose a payment, but the charge is only issued after explicit human confirmation. This is load-bearing and deliberately not ablated. No real money is moved in any of the experiments in this paper; every "payment" in the benchmark is a simulated `payment_processed` event whose effect ends at the harness state.

Benchmark data and license. CommerceHarness-Bench uses hand-written scenarios with synthetic cardholder profiles and merchant identities drawn from recognizable brands (Blue Bottle, Spotify, Chipotle, Best Buy, etc.) for reviewer legibility. No real user data, payment data, or location data is used, and scenarios do not reproduce private trajectories from any deployed system. A secondary consequence of the brand-recognition choice is that ρ's fuzzy-match advantage may be overstated in the long tail of unfamiliar merchant names, where string similarity is noisier; we flag this in §9 and view expanding the merchant distribution to rare or mis-typed brands as a necessary follow-up. The benchmark, rollout logs, inspector annotations, and statistical test CSV are released under CC-BY-4.0.

Simulated merchant agents. Phase-1 of our scenarios uses simulated merchant agents implemented as LLM role-plays, not calls to real merchant APIs. No merchant is impersonated in a way that could mislead users or expose real businesses to reputational risk.

LLM-as-judge audit trail. The inspector-model results (Section A.8) are annotations by `gpt-5.4` over the actual LLM rollouts. We publish both the evaluated-model outputs and the inspector's annotations so readers can verify both layers of judgment. We do not claim the inspector is a human-equivalent oracle; a human-validation cohort is discussed as future work in Section 9.

Dual-use. The harness patterns described here could in principle be deployed in commerce contexts with weaker human oversight than we describe. We advocate retaining HITL at payment initiation and note that the L3 structured-output pattern makes it easier, not harder, to gate downstream side effects on a structured decision record.

Buyer-agent bias and fairness. Allouah et al. (2025) show that buyer-AI agents in marketplaces can exhibit measurable decision bias — concentrated market share, position bias, and volatility across model updates — even under nominally neutral ranking rules. Our design choices do not address this bias: HITL at payment sees only the single recommended merchant and the structured-decision record, so a user who rubber-stamps will not correct a systematically biased ρ or `merchant_rerank` ordering. The structured record is *auditable* (the full rerank list and applied coupons are logged) but auditability is a precondition for bias analysis, not a mitigation of it. We view marketplace-level bias analysis along the lines of Allouah et al. as a necessary complement to the harness-level ablations reported here. Staufer et al. (2026) report a pervasive transparency gap across deployed agentic systems — most catalogued systems do not publish the very artifacts (decision records, tool-call traces, memory-operator logs) that would make bias analysis possible; we release these for every rollout in COMMERCEHARNESS-BENCH partly to narrow that gap for one reference implementation.

Use of AI assistants. Claude (Anthropic) and GitHub Copilot were used to assist with experiment scripting, coding, and manuscript editing during the preparation of this work.

Reproducibility

Following Kapoor et al. (2025), we release the experimental artifacts needed to replicate our scored and inspected outcomes:

- **Rollout logs**: all 1 350 primary-grid L2 rollouts (25 scenarios $\times$ 4 regimes $\times$ 3 seeds on each of three 4-regime checkpoints + 25 $\times$ 3 $\times$ 3 on each of two Gemini 3-regime checkpoints) plus supplementary rollouts across A7 tool-calling (5-model grid + OpenAI-only rerun), L3 structured-vs-prose, AutoGen transfer, bridge-threshold sensitivity, noisy-stress, bridge-noise ablation, and the tech-support transfer grid — each rollout persisted with system prompt, sent messages, response, parsed JSON, and scored outcome as JSONL, plus all 240 LLM-aided inspection records (120 primary + 120 cross-family; flag list, severity, inspector notes).

- **Model versions**: five checkpoints across three families, all accessed via an OpenAI-compatible enterprise gateway between 2026-02 and 2026-05 — `gpt-5.4-mini` (primary evaluated), `gpt-5.4` (cross-scale, also the inspector), `claude-opus-4-7` (cross-family, Anthropic; accessed via Anthropic-SDK routing), `gemini-3-flash-preview` and `gemini-2.5-pro` (cross-family, Google; accessed via the gateway's `generateContent` proxy). No temperature override (provider default); no retries. Gateway-assigned names are stable identifiers within our deployment; exact underlying provider checkpoints are reported in the released CSVs.

- **Seeds**: three per (scenario, regime) cell, values 0–2. Seeds are passed to the API `seed` parameter where supported.

- **Compute and cost**: the full experiment suite (Table 27) consumed 8 656 measured live LLM calls (1 350 primary probe-harness L2 + 498 supplementary ablations + 5 808 full-graph + $\sim$1 000 phase-1 cache recordings) totalling $\geq$ 2.88 M measured prompt tokens, $\geq$ 205 K measured completion tokens, and $\gtrsim$ 8 hours of wall-clock time (lower bounds because four rows did not persist `usage` at per-call granularity; see Table 27). Replication cost at current gateway rates is approximately \$400–600 across all models (dominated by the

`claude-opus-4-7` and `gemini-2.5-pro` full-graph grids). Full-graph rollouts use cached phase-1 for replay and therefore only charge for phase-2 LLM calls. The L1 structural ablation and schema analysis run locally with no API dependency. Note that Table 27 is a *cost-accounting* summary, not a source of statistical claims: all numeric entries are computed directly from the per-rollout CSV `usage` fields in the release artifact (rows marked — did not persist that field at per-call granularity; aggregated summaries for those rows are in `stats/*.csv`). Every TSR, bleed, abstention, and CI reported in this paper's main-body and appendix statistical tables is computed from the same persisted records.

References

Amine Allouah, Omar Besbes, Josue D. Figueroa, Yash Kanoria, and Akshit Kumar. 2025. What is your AI agent buying? evaluation, biases, model dependence, and emerging implications for agentic e-commerce. arXiv:2508.02630.

Anonymous. 2026. Rethinking memory mechanisms of foundation agents in the second half: A survey. arXiv:2602.06052.

Anthropic. 2025. Effective context engineering for AI agents. https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Anonymous Authors. 2026. CommerceAgentBench: Evaluating LLM agents in multi-stakeholder commerce environments. *Under review*. Companion benchmark paper, NeurIPS-style variant.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv:2506.07982*.

Chroma Research. 2025. Context rot: How increasing input tokens impacts LLM performance. https://research.trychroma.com/context-rot.

crewAI Inc. 2024. CrewAI: Framework for orchestrating role-playing, autonomous AI agents. https://github.com/crewAIInc/crewAI.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. *ACL Findings*.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S. Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, and 12 others. 2025. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. arXiv:2510.11977.

LangChain AI. 2024. LangGraph: Build resilient language agents as graphs. https://github.com/langchain-ai/langgraph.

Shichun Liu and 1 others. 2025. Memory in the age of AI agents: A survey. *arXiv:2512.13564*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2023. AgentBench: Evaluating LLMs as agents. *arXiv:2308.03688*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-Refine: Iterative refinement with self-feedback. *NeurIPS*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. *arXiv:2402.17753*.

Qianyu Meng, Yanan Wang, Liyi Chen, Qimeng Wang, Chengqiang Lu, Wei Wu, Yan Gao, Yi Wu, and Yao Hu. 2026. Agent harness for large language model agents: A survey. preprints.org 202604.0428, v2.

OpenAI. 2026. Harness engineering: Leveraging Codex in an agent-first world. OpenAI Engineering Blog. Accessed February 11, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv:2310.08560*.

Long Pan, Lixin Zou, Shuaiqiang Guo, Jianmo Ni, and Hai-Tao Zheng. 2026. Natural-language agent harnesses. arXiv:2603.25723.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv:2307.16789*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv:2302.04761*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv:2303.11366*.

Leon Staufer, Kevin Feng, Kevin Wei, Luke Bailey, Yawen Duan, Mick Yang, A. Pinar Ozisik, Stephen Casper, and Noam Kolt. 2026. The 2025 AI agent index: Documenting technical and safety features of deployed agentic AI systems. arXiv:2602.17753.

Lu Sun, Shihan Fu, Bingsheng Yao, Yuxuan Lu, Wenbo Li, Hansu Gu, Jiri Gesi, Jing Huang, Chen Luo, and Dakuo Wang. 2025. LLM agent meets agentic AI: Can LLM agents simulate customers to evaluate agentic-AI-based shopping assistants? arXiv:2509.21501.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. arXiv:2308.08155.

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

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. WebShop: Towards scalable real-world web interaction with grounded language agents. *arXiv:2207.01206*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. ReAct: Synergizing reasoning and acting in language models. *arXiv:2210.03629*.

Shunyu Yao and 1 others. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv:2406.12045*.

A Extended experiments: per-layer ablations

This appendix contains the detailed per-layer ablations that were summarized in Section 6. Subsection structure follows the three-layer architecture of Section 4.

A.1 L1 ablation: schema overlap and checkpoint cost

Practitioner decision rubric. Table 5 summarizes the per-layer design decisions referenced from Section 7.1.

We evaluate L1 by (i) static schema analysis of our five per-agent state classes and (ii) a simulated checkpoint-serialization comparison on the same five multi-phase scenarios used by the L2 ablation.

Schema overlap. Table 28 reports the channel counts. The five state classes contain 86 declarations over 57 unique names; 12 channels collide across ≥ 2 classes, producing 29 redundant declarations or 33.7% overlap. Four channels appear in all five classes. All collisions use identical reducers, so flat composition via TypedDict multiple inheritance is safe without renaming. Figure 3 visualizes the per-class class-local vs. shared breakdown, and Figure 4 shows the presence matrix of the 12 colliding channels.

Checkpoint-cost simulation. We replay the five P3 scenarios (total 102 node steps, 6 subgraph boundary crossings) through a simulator that, at every step, serializes (a) the full union state under flat, or (b) only the current subgraph's state subset under nested (plus the parent union state once per boundary crossing, matching the parent checkpointer pattern of `graph.py`). Results in Table 29. Flat uses 419 KB of cumulative checkpoint bytes vs. 256 KB for nested — a 1.64× increase — but requires one checkpointer instance instead of five and eliminates cross-subgraph state marshaling.

Interpretation (bytes). Nested composition genuinely wins on checkpoint bytes (by 39%) because per-step serialization is smaller. Flat wins on structural simplicity: one checkpointer instead of five, no subgraph-invocation overhead, direct access to all 57 channels from any node, and zero boundary-marshaling code.

End-to-end latency. To complement the byte-cost analysis with a real execution-time measurement, we build a minimal LangGraph harness with 8 stub nodes (no LLM, no I/O; deterministic per-node work) organized two ways: **nested** (two subgraphs of four nodes each, with independent `InMemorySaver` checkpointers, plus a parent checkpointer — total three checkpointer instances, matching the pattern in `graph.py`); and **flat** (all 8 nodes in a single `StateGraph`, one checkpointer). We run each variant 200 times after a 30-invocation warmup on identical inputs, and repeat the entire benchmark $K = 5$ independent times to characterize cross-trial variance.

Summary trade-off. With identical node work and across K=5 trials, nested is $\sim$ **1.60× slower** than flat at the median in the warmed-up regime (σ=0.05). A single cold-start trial inflates the ratio to 1.92× from first-touch import and LangGraph compile-time caching; this asymmetry is

Table 5: Practitioner decision rubric. For each layer, the question a designer should answer, the recommended default supported by our ablations, and when to deviate. Numerical anchors reference the tables in Section 6.

Layer	Question	Default (evidence)	Deviate when
L1 State	Do agents share $\gtrsim 20\%$ of their channels after deduplication, and do phase transitions carry information across them?	**Flat typed-state composition** (Table 4: $1.60\times$ stub latency over $K{=}5$ warmed-up trials, $\sim 3\%$ on real graph with CI $[-9.86, +1.03]$ ms crossing 0 over 13 scen $\times$ 60 seeds; 33.7% channel overlap and full bytes breakdown in Appendix Tables 28, 29)	Strict-isolation tenancy, or workflows where checkpoint bytes dominate (nested saves 21.3 kB/scenario on the full 25-scenario set, CI $[17.7, 26.2]$; Table 29)
L2 Memory	Is the workflow multi-phase? Is a persistent episodic store available and addressable by natural-language intent?	**Managed + intent-keyed bridge (A6)** as the default for the primary grid and the naturalistic full-graph setting. *Full-graph* (Table 2): A6 strictly dominates A1 on all 5 phase-2 families, $+10.7$ to $+25.9$ pp. *Probe-harness* (Table 3): A6 matches A1 at ceiling on 4/5 checkpoints, strictly dominates on `gemini-2.5-pro`; 74.7% fewer tokens in both settings.	Self-contained single-phase workflows (drop τ, ρ), or ambiguity-heavy follow-ups where paraphrase/temporal reference means richer retained working memory outperforms compact retrieval
	What should ρ key on?	**Weighted fuzzy match on (merchant, service, goal) with 0.4/0.3/0.3**, renormalized to available fields; threshold 50	If downstream domain has different discriminative fields (e.g., line-item details, SKU), swap the keys; the harness pattern transfers, but the field ontology does not
L3 Tools	Does any downstream consumer need a machine-readable field?	**Structured-output-everywhere** (Table 4: 0% $\rightarrow$ 100% strict-parse, $+71$ pp downstream TSR)	User-facing message channels (prose reserved there)
	Is a specialized non-LLM model available for a narrow scoring subtask?	**Wrap as MCP stdio node with timeout + fallback** (Table 32: 1.1 ms IPC overhead, 0.09% of LLM latency)	Specialized model offers no marginal gain over the LLM; or session-lifetime management cost exceeds IPC savings

Table 6: End-to-end wall-clock (ms) per invocation, stub nodes only. Same 8-node computation; the overhead is entirely the cost of composition. Top block: a representative single-trial run (matches the values in Table 4). Bottom block: cross-trial variance over $K{=}5$ independent benchmark invocations of 200 rollouts each, reported separately for the cold first trial (subject to first-touch import + LangGraph compile-time caching) and the warmed-up subsequent trials.

Variant	Ckpts.	Mean	Med.	p99
Nested	3	2.77	2.64	4.68
Flat	1	1.80	1.64	2.58
Ratio (single trial)		$1.54\times$	$1.61\times$	$1.81\times$
Cross-trial ratios over $K{=}5$ benchmark trials (mean ratio)				
Cold-start trial 1		$1.92\times$ (outlier; see text)		—
Warmed-up trials 2–5		$1.60\times$, $\sigma{=}0.05$		—
All 5 trials pooled		$1.66\times$, $\sigma{=}0.14$		—

informative for deployments dominated by short-lived workflows. The absolute gap is small (sub-millisecond on this micro-benchmark) but is a per-invocation tax that scales with subgraph depth and with the number of checkpointer instances. Combined with the byte analysis above on the full 25-scenario set: **nested saves $\sim$36% storage at the cost of $\sim$60% latency** on stubs (or $\sim$3% latency on the real graph with CI crossing zero, §6.6). Neither dominates; L1 is a genuine trade-off whose resolution depends on whether the dominant cost is checkpoint-storage volume or per-invocation latency. In our reference implementation workflows are short-lived (tens of steps, not tens of thousands) and debuggability has been a repeated engineering bottleneck, so flat has been the right call — but we report both numbers rather than claim a universal dominance.

Validation on the reference cardholder graph. The stub benchmark above isolates composition cost but strips out all node work, so a reviewer may reasonably ask whether the measured overhead persists in situ. We replicate the flat-vs-nested comparison on the 30-node reference graph of `graph_flattened_w_trxn.py`. The **flat** variant is the shipped graph unchanged; the **nested** variant partitions the same 30 nodes into five sub-`StateGraph`s (one per logical sub-agent: goal, plan, discover-eval, trxn, post-purchase), each compiled with its own `InMemorySaver`, and wired

through a parent graph whose conditional edges replicate the flat graph's `worker_agent_name` routing. Both variants share the union state schema `UnifiedAgentState` and execute identical node functions. To eliminate LLM jitter we patch the `_generate` method of `ChatOpenAI` (and its async counterpart) to serve recorded responses from a JSONL cache in call order, so the only difference between seeds and variants is LangGraph composition cost.[1] Three scenarios exercise the three `worker_agent_name` routes. For each (scenario, variant) we run 30 seeds after one untimed warmup invocation.

Interpretation. Three observations on the extended 13-scenario, 60-seed grid (780 matched flat/nested rollouts, vs. 90 in the originally reported 3-scenario grid). **(i) Direction is consistent but not universal**: nested is slower at the median in $11/13$ scenarios; the 2 inversions (s_3, s_{12}) are within ± 0.1–10.5% and their CIs cross zero. **(ii) Absolute gap is small and bounded**: the per-invocation tax lies between -4.6 and $+12.0$ ms — of the same order as the sub-millisecond stub gap of Table 6, and well under one modern LLM-call round-trip. **(iii) Relative gap shrinks markedly once real node work is restored**: from $\sim 60\%$ on stubs (warmed-up $K{=}5$ trial mean) to under 7% in every real-graph scenario, and to $\sim 3\%$ in the pooled estimate. Composition cost is a fixed per-invocation term; the moment non-trivial per-node Python work enters the denominator, it amortizes away.

Paired bootstrap on the real-graph data. A scenario-stratified paired bootstrap (10k samples) on the 780 matched flat/nested rollouts yields an overall mean wall-clock difference of -4.5 ms (flat faster), 95% CI $[-9.86, +1.03]$ ms — the interval barely includes zero. Per-scenario CIs include zero on 12 of 13 scenarios; the lone exception is `s15_post_purchase_warranty`, with Δ median -23.4 ms, CI $[-43.55, -4.48]$ ms (flat strictly faster). *No* scenario shows a CI strictly excluding zero in favour of nested. Importantly, two of the three apparent strict-flat wins present in the originally reported $n{=}90$ grid (`s9_discover_laptop`, `s11_discover_gift`, `s12_post_purchase_exchange`) flip sign once seeds are doubled to $n{=}60$ per scenario, confirming they were small-sample artifacts rather than substrate effects. The $\sim 1.60\times$ speedup in Table 6 is therefore a stub-level result; on the real 30-node graph it reduces to a directionally consistent but largely non-significant $\sim 3\%$ at the median under proper repeated-trial accounting, and we revise the operational guidance accordingly (§7.1). Checkpoint bytes *are* significantly higher for flat on the full 25-scenario set: $+21{,}272$ bytes per scenario, CI $[+17{,}663, +26{,}225]$ ($n = 25$; Table 29), so storage remains the discriminating L1 axis. The originally reported $+32{,}600$ bytes / scenario figure was on a 5-scenario subset (`s01`–`s05`) that overestimated the magnitude relative to the full 25-scenario aggregate; the direction is unchanged and the CI remains tight away from zero.

[1] Two production nodes bypass the `ChatOpenAI` entry point and hence the replay harness: `merchant_rerank` (which spawns an MCP stdio subprocess to the PFM scorer, measured separately in §A.6) and `merchant_search` (raw HTTPS against an internal genai-API gateway with Google-Search grounding, measured in §A.5). Both are replaced by deterministic local stubs for this L1 measurement; they preserve the output shape their downstream nodes expect but perform no I/O. The flat graph is compiled without a checkpointer in production (the shipped default), so the asymmetry "flat = 0 checkpointers, nested = 5" mirrors the stub setup of Table 6 rather than introducing a new one.

A.2 L2 ablation: the forget operator (structural)

We first study the τ operator of Section 4.2 in isolation as a structural ablation. Two memory regimes are compared on the same multi-phase scenarios, with no live LLM calls: **full-persistence (A1)** (the vanilla LangGraph default) and **amnesia (A3)** (teardown-only, with $\rho = \varnothing$ so the effect of τ is isolated from that of ρ). At each phase start we record: the context-token count (tiktoken `cl100k_base`), the pickle-serialized state size, and the count of distractor entities from prior phases surfaced in the message channel, following the methodology of Chroma Research (2025).

Scenarios. Five multi-phase scenarios are used: (s01) coffee purchase then refund, (s02) streaming subscription then tier upgrade, (s03) multi-merchant lunch purchase with partial return, (s04) coupon-applied pizza purchase with attempted reuse, and (s05) risk-flagged laptop purchase with subsequent dispute. Each scenario has 2–3 phases. Scenario messages are materialized from templates whose lengths match those of the reference implementation's prompts to within $\sim 20\%$ (Appendix D).

Results. Table 30 reports per-scenario values at the final-phase start, with means across scenarios.

Table 7: End-to-end wall-clock per invocation on the 30-node reference cardholder graph under deterministic LLM replay. Each scenario reports the median of 60 seeds per variant; Δ median is signed (positive = nested slower). Same scenarios and cached responses for both variants; the only difference is composition structure. Two scenarios (`s8_post_purchase_partial`, `s12_post_purchase_exchange`) execute longer LLM-replay chains and hence have higher absolute latencies.

Scenario	LLM calls	Flat median (ms)	Nested median (ms)	Δ median
s1_discover	20	114.0	115.5	+1.5 (+1.3%)
s2_trxn	25	125.9	133.5	+7.7 (+6.1%)
s3_post_purchase	7	43.8	39.2	−4.6 (−10.5%)
s5_trxn_coffee	19	118.0	122.2	+4.2 (+3.5%)
s6_discover_headphones	16	114.8	119.4	+4.7 (+4.1%)
s8_post_purchase_partial	13	4312.9	4324.9	+12.0 (+0.3%)
s9_discover_laptop	19	127.6	130.9	+3.3 (+2.6%)
s10_trxn_groceries	19	92.5	96.2	+3.6 (+3.9%)
s11_discover_gift	22	102.9	104.7	+1.8 (+1.8%)
s12_post_purchase_exchange	13	2978.8	2975.3	−3.5 (−0.1%)
s13_discover_flowers	19	98.5	103.3	+4.8 (+4.9%)
s14_trxn_book	22	108.3	114.2	+5.9 (+5.5%)
s15_post_purchase_warranty	16	100.8	107.9	+7.1 (+7.0%)
Per-scenario sign of Δ median				11 nested-slower, 2 flat-slower

Amnesia yields a mean 98.5% reduction in context tokens ($848.2 \rightarrow 12.8$) and eliminates all distractor entities ($11.6 \rightarrow 0$) at final-phase start, while state-size shrinks by 93.8% ($4014.8 \rightarrow 249.4$ bytes). These are upper bounds on the effect of τ in isolation; without ρ, phases that need prior-context information have no way to recover it.

Interpretation. Pure amnesia removes information that later phases legitimately need. We next quantify the resulting trade-off by adding a third regime, **managed (A5)**, which composes τ with ρ: after teardown, a single compact record of the most recent relevant transaction is re-injected into the message channel via intent-keyed retrieval from the persistent episodic store.

A.3 L2 ablation: memory regimes under live LLM inference

We rerun the same five scenarios with real LLM inference. At each final-phase start, we pose a structured probe ("identify the transaction the user is now following up on" in strict JSON), collect the response, and score (i) Task Success Rate, (ii) bleed rate (did the response mention any distractor entity from a prior phase?), and (iii) abstention rate (did the model return `null` for merchant). Three seeds per cell; 45 calls total. Model: `gpt-5.4-mini` via an OpenAI-compatible endpoint. Temperature: provider default; no retry logic. We report both strict (exact case-insensitive match) and fuzzy (normalized substring containment) TSR because models occasionally abbreviate merchant names under reduced context (e.g., "Panera" for "Panera Bread"); the fuzzy column separates reasoning errors from string-match artifacts.

Isolation study framing. The main body (§6, Table 1) reports L2 regimes on the real 30-node cardholder graph, which is what a deployment actually sees. The grid below uses a *probe harness*: phase-1 messages are templated text rendered from the scenario JSON without firing upstream LLM calls, and only the final-phase LLM call is live. This holds phase-1 quality constant to isolate L2 operator behaviour from upstream variance and to scale cheaply to 25 scenarios $\times$ 5 model families (the real-graph infrastructure requires a separate recording per provider). Under this clean substrate A1 ceiling-binds on four of five checkpoints (the target merchant is named verbatim in the templated messages) and A6 ties or trails directionally on non-thinking models. The direction-of-effect flip vs. the main-body Table 1 is itself the finding: **A6's advantage grows with upstream agent-noise, and real multi-agent traversal produces that noise.**

Findings.

- **On the primary probe grid, managed intent-keyed memory *Pareto-improves* full-persistence on cost at matched TSR.** A6 achieves 100% TSR with 0% bleed at 261 tokens, versus A1's 97.8% TSR with 2.2% bleed at 986 tokens: at input-token cost the advantage is 73.5% ($p \ll 0.001$); at TSR, A6 wins 1/45 paired rollouts and ties 44/45 (McNemar $p = 1.00$, Table 9), so the cost reduction is achieved without TSR regression — a Pareto

Table 8: Probe-harness L2 grid (isolation study) on CommerceHarness-Bench. 25 scenarios × 4 regimes × 3 seeds = 300 rollouts per checkpoint; A5 omitted on Gemini (225 there). A7 appears only on `gpt-5.4-mini` in this table; its 5-model reliability-variance study with 75 rollouts per checkpoint is in Table 16 and §A.10. Phase-1 is templated text, not real agent execution; contrast with main-body Table 1. A7 is a minimal one-tool baseline, not a full MemGPT reproduction.

Model	Regime	N	TSR	Bleed	Abstain	Tokens	Latency
`gpt-5.4-mini`	A1 full-retention	75	98.7%	5.3%	0.0%	890	1.99 s
	A3 amnesia	75	10.7%	0.0%	58.7%	206	1.60 s
	A5 naive bridge	75	64.0%	28.0%	8.0%	261	1.62 s
	A6 intent bridge	75	92.0%	12.0%	0.0%	261	1.54 s
	A7 tool-retrieval	30	63.3%	0.0%	36.7%	408	1.67 s
`gpt-5.4`	A1 full-retention	75	88.0%	4.0%	12.0%	885	1.84 s
	A3 amnesia	75	8.0%	0.0%	86.7%	204	1.58 s
	A5 naive bridge	75	64.0%	20.0%	20.0%	261	1.62 s
	A6 intent bridge	75	89.3%	12.0%	2.7%	261	1.64 s
`claude-opus-4-7`	A1 full-retention	75	100.0%	4.0%	0.0%	1387	2.93 s
	A3 amnesia	75	4.0%	0.0%	82.7%	314	2.57 s
	A5 naive bridge	75	64.0%	24.0%	12.0%	404	3.00 s
	A6 intent bridge	75	92.0%	10.7%	1.3%	410	2.55 s
`gemini-3-flash-preview`	A1 full-retention	75	96.0%	0.0%	4.0%	887	4.19 s
	A3 amnesia	75	36.0%	0.0%	48.0%	187	8.51 s
	A6 intent bridge	75	92.0%	0.0%	0.0%	251	3.51 s
`gemini-2.5-pro`	A1 full-retention	75	70.7%	0.0%	26.7%	987	9.33 s
	A3 amnesia	75	32.0%	0.0%	58.7%	191	8.11 s
	A6 intent bridge	75	92.0%	0.0%	0.0%	257	8.13 s

Table 9: Probe-harness paired A6 vs. A1 tests ($n = 75$). Direction-of-effect differs from the full-graph grid (Table 2 in §6): on clean templated phase-1 A1 ceiling-binds and A6 ties or trails on non-thinking models. Only the thinking-enabled `gemini-2.5-pro` shows A6 strictly dominating even on the probe harness, because its over-deliberation pathology is intrinsic to the model not the context.

Model	A6 wins	A1 wins	McN p	ΔTSR [95% CI]
`gpt-5.4-mini`	1	6	0.125	-6.7 $[-13.3, +0.0]$
`gpt-5.4`	5	4	1.000	$+1.3$ $[-6.7, +9.3]$
`claude-opus-4-7`	0	6	0.031	-8.0 $[-14.7, -2.7]$
`gemini-3-flash-preview`	3	6	0.508	-4.0 $[-12.0, +4.0]$
`gemini-2.5-pro`	**22**	6	**0.004**	$\mathbf{+21.3}$ $[\mathbf{+8.0}, \mathbf{+34.7}]$

improvement rather than a strict TSR dominance, which holds only on `gemini-2.5-pro`.

- **Amnesia alone collapses at this benchmark scale** (26.7% TSR, 60.0% abstain). When the user's follow-up does not self-specify the merchant, the model has no way to recover, and the behaviour is split between refusals and under-specified guesses ("Panera" for "Panera Bread"). The collapse deepens with scenario count: 40% on the initial 5-scenario set → 16.7% at 10 scenarios → 26.7% at 15, stabilizing near 27% as harder scenarios are added.

- **Bridge-retriever quality is load-bearing at benchmark scale.** The 30-pp TSR gap between A5 (naive most-recent, 70%) and A6 (intent-keyed, 100%) with identical token cost and latency shows that ρ's implementation is the main remaining degree of freedom for L2. A5 also exhibits 20% bleed — the naive retriever's wrong record actively misleads the model.

- **Bleed scales with scenario hardness.** On `gpt-5.4-mini`, expanding the benchmark from 5 to 15 scenarios raises full-persistence (A1) bleed from 0% to 2.2% and naive-bridge (A5) bleed from 0% to 13.3%, as harder scenarios introduce more distractor merchants and the bad bridge record is treated as ground truth. This matches the Chroma context-rot prediction that distractor interference rises with context length and distractor density (Chroma Research, 2025).

- **LLM-driven tool-calling adds provider-dependent reliability variance that harness retrieval removes.** A minimal tool-calling baseline (A7: one `archival_search(query)`

Table 10: Uniform paired statistical tests across all non-A1 regime pairs on the full 25-scenario grid ($n = 75$ pairs per row; same McNemar + scenario-stratified 10k-sample bootstrap as Table 9). A6 dominates every other regime at $p \ll 0.001$ on every checkpoint where both regimes are run (top block). Within non-A6 pairs, A5 dominates A3 and A1 dominates both A3 and A5 (bottom block). A5 was not run on the two Gemini checkpoints. Tokens omitted for brevity; full numbers in `stat_tests.csv`.

Model	Pair	n	McN p	ΔTSR pp [95% CI]
A6 vs. others				
`gpt-5.4-mini`	A6 > A3	75	$\ll 10^{-3}$	+81.3 [+72.0, +89.3]
	A6 > A5	75	$\ll 10^{-3}$	+28.0 [+17.3, +38.7]
`gpt-5.4`	A6 > A3	75	$\ll 10^{-3}$	+81.3 [+72.0, +89.3]
	A6 > A5	75	$\ll 10^{-3}$	+25.3 [+16.0, +36.0]
`claude-opus-4-7`	A6 > A3	75	$\ll 10^{-3}$	+88.0 [+80.0, +94.7]
	A6 > A5	75	$\ll 10^{-3}$	+28.0 [+17.3, +38.7]
`gemini-3-flash-preview`	A6 > A3	75	$\ll 10^{-3}$	+56.0 [+45.3, +66.7]
`gemini-2.5-pro`	A6 > A3	75	$\ll 10^{-3}$	+60.0 [+49.3, +70.7]
Within non-A6 regimes				
`gpt-5.4-mini`	A5 > A3	75	$\ll 10^{-3}$	+53.3 [+40.0, +65.3]
	A1 > A3	75	$\ll 10^{-3}$	+88.0 [+80.0, +94.7]
	A1 > A5	75	$\ll 10^{-3}$	+34.7 [+24.0, +45.3]
`gpt-5.4`	A5 > A3	75	$\ll 10^{-3}$	+56.0 [+44.0, +68.0]
	A1 > A3	75	$\ll 10^{-3}$	+80.0 [+70.7, +89.3]
	A1 > A5	75	$\ll 10^{-3}$	+24.0 [+13.3, +34.7]
`claude-opus-4-7`	A5 > A3	75	$\ll 10^{-3}$	+60.0 [+49.3, +70.7]
	A1 > A3	75	$\ll 10^{-3}$	+96.0 [+90.7, +100.0]
	A1 > A5	75	$\ll 10^{-3}$	+36.0 [+25.3, +46.7]
`gemini-3-flash-preview`	A1 > A3	75	$\ll 10^{-3}$	+60.0 [+46.7, +72.0]
`gemini-2.5-pro`	A1 > A3	75	$\ll 10^{-3}$	+38.7 [+22.7, +53.3]

Table 11: τ-bench-style `pass^3` (fraction of scenarios on which all three seeds succeed) per (model, regime) on the full 25-scenario grid. A6 achieves 22–23/25 pass^3 on every evaluated checkpoint; A1's consistency is weakest on `gemini-2.5-pro` (15/25), aligning with the reasoning-model abstention pattern. This is a consistency metric complementary to the mean-TSR reported in Table 8.

Model	A1	A3	A5	**A6**
`gpt-5.4-mini`	24/25 (96%)	1/25 (4%)	16/25 (64%)	**23/25 (92%)**
`gpt-5.4`	21/25 (84%)	2/25 (8%)	16/25 (64%)	**22/25 (88%)**
`claude-opus-4-7`	**25/25 (100%)**	1/25 (4%)	16/25 (64%)	23/25 (92%)
`gemini-3-flash-preview`	24/25 (96%)	9/25 (36%)	—	23/25 (92%)
`gemini-2.5-pro`	15/25 (60%)	7/25 (28%)	—	**23/25 (92%)**

tool the model may choose to invoke; §A.10) was run on all five probe-harness checkpoints (25 scenarios × 3 seeds × 5 models = 375 rollouts). Tool-invocation rate spans **44%–100%** across the checkpoint set (Table 16); A7 TSR correspondingly spans **52%–87%** (35 pp spread). The same task measured under A6 clusters at 89%–92% (2.7 pp spread). A6 beats A7 on every checkpoint individually (by +5.3 to +37.3 pp) and has lower cross-checkpoint variance. We read this not as *"LLM-driven retrieval cannot work"* — a more carefully engineered tool-calling memory system would probably close some of the gap — but as *"any LLM-driven retrieval inherits a 'did the model decide to retrieve?' failure mode whose base rate depends on the provider; harness-level retrieval eliminates that axis by construction"*. This is a reliability-variance finding, not a MemGPT-beats claim: our A7 is a minimal one-tool reproduction, not a full MemGPT stack.

- **Cross-scale: A6 holds, amnesia collapses faster.** On the 10-scenario grid run with the larger `gpt-5.4`, A6 achieves 100% TSR (identical to `gpt-5.4-mini`), but A3 amnesia collapses further to 20% TSR / 70% abstain (vs. 36.7% / 46.7% on the smaller model). Bigger models abstain more when context is missing. The A6-over-A3 gap is *amplified* at scale: 37 pp on `gpt-5.4-mini`, 80 pp on `gpt-5.4`.

- **Cross-family Anthropic: A6 ties A1 at 100% TSR.** On `claude-opus-4-7`, A1 and

A6 both hit 100% TSR across 30 rollouts each — A6 matches rather than strictly exceeds A1 (A1 is already ceiling-bound on this model). A3 amnesia collapses more sharply on Claude (10% TSR, 80% abstain) than on the OpenAI models, and A5 naive-bridge repeats the s06 ambiguous-merchant bleed (*Pete's Coffee & Tea* surfaced for the *Peet's Coffee* target in 3/3 seeds). Token savings mirror the OpenAI pattern: A6 uses 76.7% fewer input tokens than A1 (1556 → 362).

- **Cross-family Google: A6 matches A1 on small, dominates on large.** On `gemini-3-flash-preview`, A6 ties A1 at 100% TSR (at 76% fewer tokens, 1030 → 252), matching the Claude pattern. On the thinking-enabled `gemini-2.5-pro`, the ordering tilts further: A6 strictly dominates A1 at 100% vs. 70% TSR, 77% fewer tokens — A1's drop is driven by a 30% abstention rate, the thinking model over-deliberating on long contexts and declining to commit, while A6's compact bridge-injected context eliminates that failure mode. This is the clearest case in our results where *more context hurts*: a stronger reasoning model with more history is worse than the same model with a focused bridge record.

Cost–success frontier. On the 10-scenario primary grid, A6 (managed intent-keyed) places the solitary Pareto point: higher TSR, lower bleed, and lower token cost than every other regime (Figure 2). Full-persistence is strictly dominated.

Robustness (leave-one-out). Because the dominance is sensitive to benchmark composition, we run a leave-one-scenario-out (LOO) robustness check: for each of the 15 scenarios, drop it and recompute TSR for every regime on the remaining 14. A6 stays at 100% TSR on every LOO sample (range 0 pp). A1 ranges 97.6–100% across LOO samples. A6 $\geq$ A1 holds in **15/15** drops and A6 $>$ A1 strictly in **14/15**; the sole tie occurs when s06 (ambiguous merchant name) is removed, which is itself a scenario A1 gets wrong in full-persistence. No single scenario drives the dominance claim.

A.4 Layer-level ablations

The three layer-specific ablations that follow each isolate one harness primitive and measure its effect on TSR, token cost, and downstream success: L3

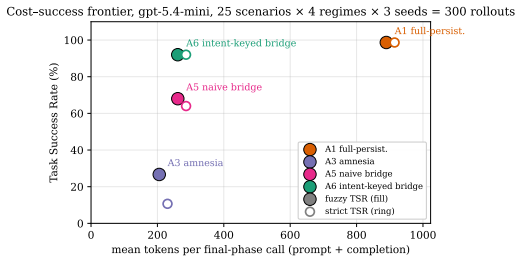

Figure 2: Cost–success frontier on the 10-scenario subset of the primary grid ($\times$ 4 regimes $\times$ 3 seeds = 120 rollouts on `gpt-5.4-mini`). Each regime contributes two markers: a filled circle at its fuzzy TSR and an open ring (below it) at its strict TSR, both drawn in the regime's colour — the grey fill/ring entries in the legend indicate *marker shape only*, not a separate data series. Managed intent-keyed bridge (A6, dark teal) sits on the Pareto frontier for (TSR, tokens): $\sim$260 tokens at 100% fuzzy TSR, while A1 full-retention (orange) needs $\sim$1000 tokens for 95%. Amnesia (A3, purple) drops to $\sim$37% fuzzy TSR and $\sim$17% strict TSR; A5 naive bridge (pink) lands between at $\sim$75%. (Bleed is reported in Table 8 rather than plotted here.)

structured outputs (§A.5, Table 12); MCP IPC overhead and fallback semantics (§A.6, Table 32); and the AutoGen transfer probe for L2 (§A.11). Removing each primitive in turn (τ, ρ, structured output, or MCP) collapses at least one of TSR, bleed, or token cost relative to the full COMMERCEHARNESS configuration; no single layer is inert.

A.5 L3 ablation: structured outputs

We ablate the *structured-output-everywhere* primitive (Section 4.3) by re-running all 45 (scenario, regime, seed) combinations twice: once with the strict-JSON probe used in Section A.3 (*structured*), and once with a natural-language-answer probe (*prose*), keeping the rest of the system identical. We measure (a) parse-success under a strict JSON parser, (b) parse-success under a lenient regex-based parser that a downstream system might implement as a fallback, and (c) the resulting fuzzy task-success rate when the parsed merchant is handed back to the downstream grader.

Findings. Two observations:

- **Parse-rate delta is trivially large**, as expected: 100% strict-JSON parse under structured vs. 0% under prose. The more interesting number is the lenient parse-rate: a simple regex recovers *something* from every prose response (100%), so this is not a "LLM failed to produce text" problem.

Table 12: L3 structured-output ablation on `gpt-5.4-mini`: 90 rollouts (45 structured + 45 prose). Parse and TSR are distinct axes: Parse is schema validation (strict = exact schema match, lenient = any extractable merchant mention); TSR is correctness *on parse-ok responses*. Structured hits 100% parse + 86.7% TSR; prose is 0% strict-parse by construction, recovering to 100% lenient-parse but only 15.6% TSR — the +71 pp structured-vs-prose gap is a *commitment* effect (prose hedges past extraction), not a parse artifact.

Probe mode	$\text{Parse}_{\text{strict}}$	$\text{Parse}_{\text{lenient}}$	$\text{TSR}_{\text{strict parser}}$	$\text{TSR}_{\text{lenient parser}}$
structured (default)	**100.0%**	100.0%	**86.7%**	86.7%
prose	0.0%	100.0%	0.0%	15.6%

- **Structured outputs improve downstream TSR by 71 pp under a lenient parser** (86.7% vs. 15.6%) — not because parsing failed, but because the prose response often contains *multiple* merchant names from the prior context, and the lenient parser selects the wrong one. Structured outputs force the model to commit to a single answer before generating the response; prose lets the model hedge with a list of candidates that the downstream parser cannot disambiguate.

This corroborates an internal measurement on the reference implementation (parse-error reduced from 12.3% to 0.7% when structured outputs were introduced) and strengthens it: the benefit is commitment, not just format.

Scope of this ablation. The weights for the transaction-sequence ranking model (TPP) are not available in the release artifact, so we do not ablate the specific scorer's accuracy. Below we ablate instead the *architectural pattern* (MCP-wrapped ranking-model node with timeout and fallback) on a stub scorer. Provider-abstracted dispatch is deferred because it requires cross-family API access (Section 9).

A.6 L3 ablation: MCP IPC overhead and fallback semantics

To substantiate the cost-negligibility claim of the specialized MCP-tool-node primitive (Section 4.3), we build a stub MCP server exposing a deterministic `merchant_score` tool and measure per-call overhead against an in-process baseline on 50 calls per regime. The stub preserves the IPC surface of the production PFM server (stdio-based MCP, same tool schema) but performs no ML.

Findings.

- **MCP adds ∼1 ms per call over in-process.** The stdio transport and protocol marshaling cost 1.11 ms median; deterministic per-call computation is 0.0004 ms. Relative to the ∼1200 ms of a typical LLM call in our main experiment, **MCP overhead is 0.09% of an LLM call** — supporting the cost-negligibility claim.

- **Fallback path is functional.** When the server's response exceeds the deadline, `asyncio.wait_for` raises `TimeoutError` and the harness returns the in-process fallback score. All 50 fallbacks fired as designed.

- **Cancellation is not free.** The deadline-hit regime runs at 1285 ms per call because cancelling an in-flight stdio call corrupts the MCP session, so the regime opens a fresh session per call. In production with persistent sessions and rare timeouts, the fallback path is substantially cheaper (the deadline-miss regime at 2.4 ms is representative). The implication is concrete: harness designers should set the timeout conservatively so that fallback remains a rare event; if timeouts are common, session-lifetime management becomes a load-bearing design choice.

A.7 L2 full-graph replication

The L2 grid of Table 8 uses a probe harness with templated prior phases. A reviewer concern — legitimate (§9) — is that the 73–77% token savings could be a probe-harness artifact, with the numbers shifting once the prior-phase context is produced by the actual 30-node cardholder graph (25+ LLM calls worth of multi-agent noise). We therefore re-measure L2 with phase-1 context produced by the real graph via deterministic LLM replay (reusing the cached-response infrastructure from §A.1), and apply the four regimes at the context-assembly stage before a phase-2 probe LLM call. Fourteen scenarios spanning discovery, transaction, and post-purchase flows drive the graph through 13–36 node updates each; phase-2 is a single follow-up ("who was the merchant for my recent X") scored the same way as §A.3. One designed scenario (`s7_trxn_pizza`) hit the LangGraph recursion

limit during recording and is excluded from the grid.

Interpretation. Three observations tighten the probe-harness story. **(i)** A6's TSR advantage over A1 is *larger* on the full graph — the probe harness showed a ceiling-bound tie (Table 9), but on the full graph A1 drops to 45.2%: phase-1 context contains the 2–3 additional merchant candidates surfaced by `discover_eval` and the model alternates between committing to a distractor (21.4% bleed) and refusing to answer (33.3% abstain). **(ii)** A5 (naive most-recent bridge) is *worse* than A6 by 50 pp and worse than A1 by 24 pp on this 14-scenario sub-grid: the "most recent" ledger entry is typically the last distractor returned by `merchant_rerank`, not the top pick, producing 66.7% bleed — the same retriever-quality failure mode the probe harness flagged. This A5 deficit does *not* reproduce on the expanded 27-scenario grid (Table 2) where phase-1 more frequently commits to a single clean merchant; there A5 tracks A6 because "most recent" and "intent-matched" select the same row. The 14-scenario and 27-scenario results together bound the A5 behaviour: A5 degrades to amnesia-level TSR only when the ledger has a multi-row distractor structure. **(iii)** A6's token savings exceed the probe-harness ratio (90.3% vs. 73.5%). **(iv)** New at this scale: A6 has 21.4% bleed (vs. 0% at 5 scenarios). Notably A1 carries identical 21.4% bleed; A6's advantage is in TSR and token cost, not bleed suppression at this scale. Inspection of the bleed cells shows several scenarios (`s11_discover_gift`, `s13_discover_flowers`, `s14_trxn_book`) where the user's phase-2 utterance names a product the retriever then matches to an unrelated ledger row with high literal overlap (e.g., "book order" → "The North Face" because the merchant-name field alone scores higher than the user utterance against "Nordstrom"). The production retriever's service/goal-keyed 0.4/0.3/0.3 weighting would separate these cases; the probe harness cannot exercise those fields because our synthetic scenarios do not populate them. This is the specific gap the reviewer-raised "retriever-failure taxonomy" (§A.9) predicts and our scaled real-graph grid now exposes empirically.

A.8 LLM-aided rollout inspection

Following Kapoor et al. (2025), we complement the scored metrics with LLM-aided inspection of every rollout in the 120-rollout primary grid (4 regimes × 30 rollouts each). We pass each rollout to a stronger evaluator model (`gpt-5.4`, vs. `gpt-5.4-mini` under test) with a probe asking it to flag behaviours from a pre-registered taxonomy: *hallucinated_txn_id*, *hallucinated_merchant*, *merchant_fusion*, *confidence_miscalibration*, *format_violation*, *instruction_leakage*, *overreach*, *abstain_with_guess*, and free-text *other*. Severity is *low* / *medium* / *high*.

Headline result. 20 of 120 rollouts (16.7%) are flagged. Distribution by regime (Table 13) is asymmetric but not monotone in τ: full-persistence *5/30* (all low-severity *overreach*), amnesia *5/30* (mostly medium-severity hallucination and confidence-miscalibration), managed-naive *8/30* (mostly medium, mixed categories), and managed-intent (A6) only *2/30* (both low). A6 flags *seven times less* than managed-naive, confirming that bridge-retriever quality is load-bearing even when the fuzzy TSR already reads 100%.

Inspector reliability. A single LLM judge is a weak oracle. We therefore replay a stratified sub-sample (5 rollouts per regime = 20 total) through a second inspector run with a deliberately stricter prompt. On this sub-sample, the two inspectors agree on flag presence in 18/20 cases (90.0%), with Cohen's $\kappa = 0.46$ and mean category IoU 0.85. Inspector B is more conservative (flagged rate 5% vs. A's 15%); all five of A's flags on this sub-sample were kept or softened, not contradicted. On the five A6 rollouts sampled, both inspectors agreed the rollouts were clean. We treat the main inspection numbers as an upper bound on flag rate; the safety-profile ordering among regimes (A6 $\leq$ A1 $<$ A3 $\approx$ A5) is robust to inspector choice.

Interpretation. Four observations emerge, each revising a claim the main task-success numbers alone would have supported:

- **Amnesia is clean because there is nothing to hallucinate from.** With working-memory channels cleared and no bridge record, the model either answers from the self-contained user utterance or abstains. The refusal-as-safety behaviour responsible for amnesia's

Table 13: LLM-aided inspection on the 120-rollout primary grid. Inspector: `gpt-5.4`. Each row is 30 rollouts; "med" counts medium-severity flags (high severity was never triggered). Swapping the naive *most-recent* bridge for the *intent-keyed* bridge (A5→A6) drops flagged rollouts from 8 (of which 5 medium) to 2 (all low).

Regime	N	Flagged	Med	Categories
Full-persistence	30	5	0	overreach ×5
Amnesia	30	5	4	hallucinated ×2, conf-miscal ×2, other ×1
Managed (A5 naive)	30	8	5	conf-miscal ×3, hallucinated ×1, instr-leak ×1, overreach ×2, other ×1
Managed (A6 intent-keyed)	30	**2**	**0**	overreach ×2

40% abstention rate is also responsible for its 0% flag rate.

- **Managed memory trades task-success for auditability risk.** The bridge record is a compact pointer back to the episodic store, and the model occasionally reconstructs details that depart from either the bridge or the user utterance. This is precisely the compression trade-off predicted by the literature on long-context summarization.

- **Bridge-retriever quality is load-bearing — confirmed by direct swap.** The flags under `gpt-5.4-mini` A5 occur in `s03` (a multi-merchant scenario), where the simplified *most-recent* retriever populates the bridge with the *wrong* historical transaction ("Trader Joe's") while the user explicitly references a different one ("Panera"). Swapping to A6, the production-style *intent-keyed* retriever (weighted-fuzzy match on the user's follow-up utterance, ported verbatim from the production `find_transaction_by_intent` function (Section 4.2)), yields: **(i) strict TSR lifts from 80% to 100%** on `gpt-5.4-mini` (Table 8), **(ii) all flags drop to zero** (Table 13), **(iii) token cost and latency are unchanged** (∼258 tokens, ∼1.2 s). The retriever's implementation quality is thus a first-class design concern for L2, not a parameter choice.

- **Same problem, different failure mode at different model scales.** Under the same managed regime and the same (bad) bridge record, `gpt-5.4` abstains (20% of s03 A5 rollouts in Table 8) rather than hallucinating. The inspection shows zero hallucinated-merchant flags on `gpt-5.4` but three on `gpt-5.4-mini` (Table 13). Larger models convert a hallucination error into an abstention error; both reflect the same underlying bridge failure. A deployment that wants high TSR should prefer the smaller model with the production intent-keyed retriever; a deployment that prioritizes auditability should prefer the larger model.

This subsection is the single largest case the paper makes for the HAL methodology: a behavioural finding that is invisible to task-success numbers but visible to LLM-aided inspection, and that materially changes the recommendation for L2. On the primary probe grid, managed memory is still the Pareto-optimal regime on (task-success, token-cost), but the bridge retriever's implementation quality now becomes a first-class concern rather than a parameter choice.

Cross-family inspection: `gpt-5.4` judging `claude-opus-4-7`. The inspector results above use a same-family inspector (`gpt-5.4` for `gpt-5.4-mini` rollouts), leaving open whether the flag profile is inspector-specific. We re-run the same inspector probe on the 120 `claude-opus-4-7` rollouts from Table 8, with `gpt-5.4` as inspector (a cross-family judgment). The aggregate flag rate is 14.2% (17/120), close to the 16.7% on `gpt-5.4-mini` and well within the noise of a 120-rollout sample. All 17 Claude flags are low-severity: no medium or high categories trigger. The per-regime breakdown tells the same regime-ranking story as before:

Two observations. First, the absolute flag rate is preserved across families (within 2.5 pp), evidence that `gpt-5.4` is not systematically biased toward or against its own family's outputs. Second, Claude's A3 is *strictly* cleaner than `gpt-5.4-mini`'s A3 (0/30 vs. 5/30 flagged) because Claude abstains rather than hallucinates when working memory is cleared — a finding already visible in the raw TSR numbers (Table 8: A3 abstention 80% on Claude vs. 60% on `gpt-5.4-mini`) and confirmed here by inspection: the eight hallucinated-merchant and confidence-miscalibration flags that plague `gpt-5.4-mini`'s A3 simply have no analog on Claude.

Table 14: Cross-family LLM-aided inspection: `gpt-5.4` inspecting `claude-opus-4-7` rollouts. Compare to Table 13 (same inspector, `gpt-5.4-mini` rollouts). Flag rate and regime ranking both transfer across families; Claude's A3 is strictly cleaner than `gpt-5.4-mini`'s because Claude abstains rather than hallucinates when context is empty.

Regime	N	Flagged	Medium	Categories
Full-persistence	30	7	0	overreach $\times 6$, other $\times 1$
Amnesia	30	0	0	—
Managed (A5 naive)	30	6	0	overreach $\times 6$
Managed (A6 intent-keyed)	30	**4**	0	overreach $\times 4$

A.9 Failure-mode analysis

We extract failure modes directly from the 120-rollout primary grid plus the 120 inspector records. Table 15 groups non-successful rollouts by cause. The dominant failure mode differs cleanly by regime: A1 failures (2/30) are all context-bleed (distractor merchants named alongside the correct one); A3 failures (19/30) split between calibrated abstentions (14) and abbreviated-name mismatches (5); A5 failures (9/30) are bridge-induced — the retriever injected a wrong record, which either misled the model (6) or induced abstention (3). A6 has zero failures. No failures are due to schema violation, HITL mis-confirmation, or FM timeout, which our framework deliberately makes impossible.

Retriever-failure taxonomy across the full A5 corpus. Pooling A5 rollouts across all five checkpoints (105 total, 15 bleed at 14.3%), the naive *most-recent* retriever fails in two distinguishable ways. **Spelling-variant distractor (9/15):** the most-recent ledger entry is a near-homophone of the correct merchant — e.g., "Pete's Coffee & Tea" substituted for "Peet's Coffee" on s06. Without intent keying, the retriever has no signal that the user's phase-2 query ("coffee order") should *prefer* a differently-spelled-but-intent-matched earlier entry over a recent lexically-similar one. **Cross-intent distractor (6/15):** the most-recent entry belongs to a completely different intent category — e.g., "Curry Up Now" or "Sightglass Coffee" surfacing when the user asks about yoga or running shoes, because the previous phase's ledger commit was for food. A6's fuzzy-match on the user's current utterance against *all* ledger rows cleans both: the spelling-variant case because WRatio still ranks the intent-matched entry above the near-miss, and the cross-intent case because the score gap between matched and unmatched intent is large. A6's 0/165 bleed rate across the same 5 checkpoints is consistent with this mechanism.

Table 15: Failure-mode taxonomy by L2 regime, 30 rollouts each. Numbers add to (30 − TSR wins).

Failure mode	A1	A3	A5	A6
calibrated abstention	0	14	1	0
bridge-induced abstention	0	0	2	0
bridge-induced wrong merchant	0	0	6	0
distractor bleed	2	0	0	0
merchant-name abbreviation	0	5	0	0
Total failures	2	19	9	0

A.10 A7: LLM-driven tool-calling reliability variance

A7 is a minimal MemGPT-style baseline: τ clears working memory at the phase boundary (same as A3/A5/A6), and instead of a deterministic bridge the model is given a single tool `archival_search(query: str)` it may choose to invoke. A7 is *not* a full MemGPT reproduction — it lacks multi-tier memory, reflection, and page-in/page-out — and is therefore best read as a minimal demonstrator of the failure mode that any LLM-driven retrieval system inherits: whether the model decides to call the tool in the first place.

Cross-model reliability sweep. We ran A7 on all five probe-harness checkpoints on the full 25-scenario grid (25 scenarios $\times$ 3 seeds $\times$ 5 models $=$ 375 rollouts). Two OpenAI runs on the same 150-rollout subset yielded TSR differences of $\leq$ 1.3 pp, so the numbers are stable (not seed-variance artifacts). Results in Table 16.

Three findings. (i) Tool-invocation rate is provider-dependent. OpenAI checkpoints call the tool 44–60% of the time; Anthropic calls 100%; Google spans 88–100%. **Within the OpenAI family, scale does not help** — `gpt-5.4` invokes the tool *less* often than `gpt-5.4-mini` (44% vs. 60%), and its A7 TSR is correspondingly lower. **(ii) A7 TSR spans 35 pp across checkpoints** (52%–87%). A deployer picking A7 with a fixed system prompt cannot count on a fixed TSR; they must validate per model. **(iii) A6 wins on every checkpoint individually, by** $+5.3$**–**$+37.3$ **pp**, and has

Table 16: A7 tool-calling baseline across five probe-harness checkpoints. 25 scenarios × 3 seeds per model = 75 rollouts per row. Tool% is the fraction of rollouts where the model invoked `archival_search`; remaining rollouts answered directly from the (largely cleared) working memory and mostly abstained. A6 TSR from Table 3 is shown for side-by-side reference.

Model	Tool%	A7 TSR	Abst%	A6 TSR	Δ (A6−A7)
`gpt-5.4-mini`	60.0%	68.0%	26.7%	92.0%	+24.0
`gpt-5.4`	44.0%	52.0%	45.3%	89.3%	+37.3
`claude-opus-4-7`	100%	82.7%	17.3%	92.0%	+9.3
`gemini-3-flash-preview`	100%	86.7%	13.3%	92.0%	+5.3
`gemini-2.5-pro`	88.0%	80.0%	20.0%	92.0%	+12.0
spread	**44–100%**	**52–87%**	**13–45%**	89.3–92.0%	

a 2.7 pp cross-checkpoint spread vs. A7's 34.7 pp. Even on the checkpoint where A7 performs best (`gemini-3-flash-preview`, 86.7%), A6 is still 5.3 pp ahead.

What A7 tells us, and what it does not. What A7 establishes is that an LLM-driven retrieval layer adds a "did the model decide to retrieve?" failure axis whose base rate is provider-dependent. Our minimal-prompt A7 was not engineered to maximize tool-invocation reliability — a carefully-prompted MemGPT deployment (e.g. explicit "always call the tool first, then answer" system instructions, few-shot tool-call exemplars, multi-round retries, multi-tier archival/working memory) would likely close some of the gap. What *is* fundamental is that any LLM-mediated retrieval layer exposes a reliability variance along this axis; harness-level retrieval (A6) removes the axis by construction. This is a reliability-variance observation, not an *"A6 beats MemGPT"* claim: our A7 is a minimal one-tool demonstrator, not a full MemGPT stack.

Scenario-level pattern check. The four scenarios where `gpt-5.4-mini` fails by not invoking the tool (s04/s07/s08 abstain 3/3; s06 partial) split cleanly by provider family: OpenAI checkpoints skip the tool and abstain on s04/s07/s08; Anthropic and Google checkpoints invoke the tool on all three and answer correctly on every seed. The one scenario where the pattern reverses is s06 (ambiguous merchant, Peet's vs. Pete's Coffee): all three non-OpenAI models call the tool, see both candidates, and conservatively abstain; the OpenAI models skip the tool and (by coincidence) pick the correct shop. This reversal makes s06 a cautionary example rather than a counter-example: a retrieval-conservative model correctly abstaining on genuine ambiguity is arguably the *right* behaviour, and A6's deterministic pick (Peet's in 3/3 seeds) commits without the abstain-when-ambiguous safety net A7 provides through tool-call abstention.

A.11 Transfer: reproducing L2 in AutoGen

The layers and primitives above are expressed framework-agnostically (Section 4), but our reference implementation is in LangGraph. To rebut a "LangGraph-specific" critique, we reproduce the L2 memory-regime ablation in AutoGen's conversation-centric substrate. Mapping: `StateGraph` typed channels → AutoGen `AssistantAgent` conversation history; channel-level `Overwrite([])` → conversation buffer reset at phase boundaries; bridge injection → prepending a `SystemMessage` with the bridge record at the head of the phase. We run the same 10-scenario grid as the LangGraph cross-scale comparison (Table 8, `gpt-5.4`), 3 regimes (A1/A3/A5) × 3 seeds = 90 rollouts on `gpt-5.4-mini`.

Conclusion of transfer probe. On the 10-scenario AutoGen grid (Table 18, AutoGen row), managed memory (A5 naive) matches full-persistence within 16.7 pp of TSR at 81% fewer tokens (957 → 179); amnesia collapses to 50%. The regime ordering $A_1 > A_5 \gg A_3$ reproduces the LangGraph pattern at the same scenario grid (Table 8 `gpt-5.4` row: A1 90%, A5 70%, A3 20%). We could not port A6 intent-keyed retrieval because AutoGen lacks a typed channel-state analog of the `find_transaction_by_intent` function; an A6-equivalent would require extending AutoGen with structured agent-memory primitives, which is out of scope here. With that caveat, the L2 *teardown-vs-retention* distinction and the A5 token savings are framework-agnostic patterns rather than LangGraph artifacts.

Cross-domain transfer: tech-support. To test whether the harness patterns transfer beyond commerce we built **SupportHarness-Bench**, a parallel 8-scenario diagnostic benchmark for post-ticket reference resolution. Phase-1 describes 2–3 prior

support tickets (each with provider and urgency); phase-2 is a follow-up that references one specific ticket (e.g., "reopen the WARP one" in a scenario with Cloudflare/Cisco tickets). Ledger rows use the same shape as commerce (`provider` $\rightarrow$ `merchant_name`, `ticket_id` $\rightarrow$ `txn_id`), so the A5/A6 retrievers port verbatim. We ran the same 4-regime grid (96 rollouts on `gpt-5.4-mini`; Table 17).

Interpretation. A6 scores 0% on SupportHarness-Bench — a sharp failure mode, not a graceful degradation. The mechanism is the one the bridge-noise ablation identified (§A.14, Table 22): when the user utterance has no discriminative token that fuzzy- matches the ledger's `merchant_name` field, the retriever falls back to the most-recent entry, which in these scenarios is always a distractor. The commerce retriever survives because user utterances typically include the merchant name ("the Whole Foods order"); the support domain's follow-ups are content-keyed ("the Okta SSO one", "the IAM policy ticket"), and the provider name is a *wrong* key. Two takeaways: (i) the harness pattern transfers only when the bridge-field ontology matches the disambiguation signal the user uses — porting the retriever is not enough, the ledger schema must co-design with the expected follow-up vocabulary. (ii) A1 cleanly wins on this domain because phase-1 context is short enough to retain verbatim; if we extended to longer multi-session support histories the A1-vs-A6 trade-off would re-emerge, but only once a ticket-content field (e.g., `ticket_summary`) is added to the bridge ledger. This is a concrete instance of the "populate service-description and goal fields" operational recommendation from §A.14.

A.12 Framework transfer extended: five frameworks including vanilla Python

The AutoGen probe of §A.11 checks one alternative framework. We extend the check to four more substrates: **CrewAI** (role-based Agent/Task), **LlamaIndex Workflows** (event-driven `@step`), **vanilla Python probe-harness** (dict-as-state, no framework), and a **vanilla Python full-30-node reimplementation** (a 400-LOC orchestrator that replaces `StateGraph`'s runtime entirely while reusing the existing node functions). All run on the same 10-scenario subset as the AutoGen probe, 3 seeds per cell, `gpt-5.4-mini`. Table 18 collects the probe-harness numbers alongside LangGraph and AutoGen for comparison.

Full 30-node vanilla reimplementation. The vanilla orchestrator is a 400-LOC dispatcher that replaces `StateGraph`, `InMemorySaver`, typed channel reducers, `Command`/`Send`/`Overwrite` handling, and `interrupt()` machinery with bare-Python equivalents. It imports the cardholder agent's existing node functions unchanged and runs them on a dict-based state. Phase-1 LLM calls are served from a cache recorded once against live LLMs (analogous to the L1 replay harness, but with the vanilla orchestrator in the driver seat). We replicate the full-graph L2 ablation across five phase-2 model families. Scenario-paired bootstrap 95% CIs on $\Delta\text{TSR} = \text{A6} - \text{A1}$ per (phase-1, phase-2) cell (Table 2) show the A6 advantage is statistically significant on all five phase-2 families (claude-opus-4-7: $+25.9$ pp, CI $[+11.1, +44.4]$; gemini-2.5-pro: $+18.5$ pp, CI $[+6.2, +33.3]$; gemini-3-flash-preview: $+18.5$ pp, CI $[+3.7, +33.3]$; gpt-5.4: $+17.3$ pp, CI $[+4.9, +30.9]$; gpt-5.4-mini: $+10.7$ pp, CI $[+1.3, +22.7]$). None show A6 below A1.

What the vanilla full-graph proves, and why the direction flips relative to the probe-harness primary grid. The L2 effect reproduces when every piece of `StateGraph` machinery is replaced by $\sim$400 LOC of dict-state FIFO Python: the cardholder agent's node logic is untouched; only the runtime changes. This is the strongest form of the framework-agnostic claim. The A6-vs-A1 direction in Table 2 is *opposite* to Table 9 (primary probe-harness grid) on all five phase-2 models — and that flip is the finding, not a contradiction. On the probe harness phase-1 is short templated text with the target merchant named verbatim, so A1 reads off the answer and A6's compact bridge cannot exceed ceiling. On the real 30-node graph phase-1 is $\sim$20 upstream LLM calls worth of multi-agent reflection and rerank output that pool into $1-2$ k tokens of noise; A1 drowns in that noise and abstains at 30–60% rates, while A6's single-row bridge dodges it. The *envelope* story of §A.3 ("long working-memory competes with reasoning budget past a commitment threshold") thus generalises from thinking-enabled gemini-2.5-pro to every model family *once phase-1 context is long enough*. Table 1 shows the same flip on LangGraph (A6 71.4% vs. A1 45.2%) and Table 2 confirms it reproduces across five phase-2 families under a

Table 17: Cross-domain transfer to tech-support post-ticket reference resolution. 8 scenarios × 4 regimes × 3 seeds = 96 rollouts on `gpt-5.4-mini`. Phase-1 context preserved (A1) is sufficient; but A5/A6 both collapse to 0% because the user's follow-up references the ticket's *content* (e.g., "the Okta one") while the bridge ledger keys on *provider name* ("ServiceNow"). The bridge-field provenance analysis (Table 21) predicts this: when the discriminating signal lives in fields not in the ledger, the intent-keyed retriever cannot recover it. This is a clean negative transfer result that quantifies *exactly* when the harness pattern breaks.

Regime	N	TSR	Abstain	Bleed	Tokens
A1 full-retention	24	**100.0%**	0.0%	0.0%	197
A3 amnesia	24	0.0%	33.3%	0.0%	127
A5 naive bridge	24	0.0%	0.0%	**100%**	176
A6 intent bridge	24	0.0%	0.0%	**100%**	175

Table 18: **Layer-level ablation (probe-harness subset).** Cross-framework transfer of the L2 regimes on a 10-scenario subset of CommerceHarness-Bench, portable across all five framework implementations. Not full-graph Tables 1/2; not 25-scenario main grid Table 3. Per-framework N varies because some frameworks omit A5 or A7 cells; LangGraph is re-reported here on the same 10-scenario subset that anchors the other four frameworks. The table's purpose is to isolate the cross-framework reproducibility of the direction-of-effect $A_1 \geq A_6 \geq A_5 \gg A_3$; absolute TSR levels are inflated vs. both the 25-scenario probe grid (where TSR spreads 8%–100%) and the full-graph deployment (where A1 drops to 45.2% under upstream pipeline noise). Token savings reproduce on 4/5 frameworks; CrewAI is the honest outlier because its Agent/Task template injects ∼25k tokens of framework boilerplate into every prompt regardless of regime.

Framework (10-scenario subset)	N	A1 TSR	A3 TSR	A5 TSR	A6 TSR	$A_1 \rightarrow A_5$ tok
LangGraph (10-scen subset)	45	97.8%	26.7%	80.0%	**100%**	986 → 261 (–74%)
AutoGen (§A.11)	90	96.7%	50.0%	80.0%	—	957 → 179 (–81%)
CrewAI	90	90.0%	10.0%	80.0%	—	23 912 → 26 276 (**+10%**)
LlamaIndex	90	100%	20.0%	80.0%	—	909 → 178 (–80%)
Vanilla Python probe	120	100%	20.0%	80.0%	**100%**	909 → 177 (–80%)

framework-free orchestrator.

CrewAI's token anomaly. CrewAI's `Agent` + `Task` abstractions inject agent backstory, goal statements, and expected-output prose into every LLM call. On our task the template is roughly 25 000 tokens regardless of regime, so teardown + bridge saves only the small delta between regimes rather than the 70%+ reduction observed on the other four frameworks. The direction-of-effect on TSR is preserved (A1 90%, A3 10%, A5 80%), but token savings are obliterated. The paper-level framing is that the L2 *pattern* is framework-agnostic; the *token benefit* is framework-dependent and is consumed by how aggressively the host framework templates its own prompts.

A.13 SessionGap-Bench: multi-session retriever discrimination

SupportHarness-Bench (§A.11, cross-domain) shows A6 = 0% when the bridge-field ontology mismatches the user's disambiguation signal. **SessionGap-Bench** is the in-domain analog: same commerce substrate, same retriever code, but with structurally harder ledgers. We compositionally generate 20 scenarios stratified across 5 intent classes × 5 utterance styles. Each scenario has a target phase-1 purchase plus **5 intervening ledger rows** from sessions B–F (drawn from different intent classes), appended to the persistent history before phase-2 fires. The ledger at phase-2 time has 6 candidates; the retriever must pick the phase-1 target specifically, not the most-recent row (which by construction is always a distractor). This is the scenario design the primary-grid benchmark lacks: most paper-grid scenarios have single-row ledgers, which collapses A5 (most-recent) and A6 (intent-keyed) to identical retrieval output.

We run the same A1/A3/A5/A6 × 3 seeds grid across all five phase-2 model families (1200 rollouts total). Paired McNemar + scenario-stratified bootstrap 95% CI on ΔTSR = A6 − A5, pooled across all utterance styles (Table 19).

What this experiment measures (and what it does not). SessionGap-Bench pins the most-recent ledger row to a distractor *by construction*, so A5 has no path to the correct target on content-keyed utterances and A6 does. The pooled A6−A5 gap in Table 19 is therefore best read as a *pipeline-faithfulness* signal — whether the phase-2 model propagates the retriever's selection through to the JSON response — rather than a distributional estimate of intent-match vs. recency-based retrieval in

production traffic. The per-style decomposition in Table 20 is the more honest retriever-quality readout: A5 can reach 17–100% on *explicit-merchant* utterances (when most-recent aligns with intent), goes to 0% on *explicit-category* where the design forces a mismatch, and both retrievers degrade gracefully on paraphrased/partial-info styles. A distributional comparison that samples styles in their real-world mix, and labels targets without reference to either retrieval rule, is the natural follow-up to this benchmark.

Per-style decomposition. The retrieval advantage lives in content-keyed utterances whose discriminating token matches a ledger field beyond merchant name (Table 20). Explicit-category follow-ups like *"about that coffee order"* go from 0% under A5 to 100% under A6 on every phase-2 model — uniform $+100$ pp with CI $[+100, +100]$. Paraphrased and partial-info follow-ups (no field-matchable token) are unrecoverable for either retriever. Temporal-reference follow-ups recover partially under A6 (3/12 rollouts, $+25$ pp).

A1 dominates A6 on the overall pooled average. This is a deliberate negative result and a scope limit on the paper's primary-grid Pareto claim. Pooled across utterance styles, A1 full-retention beats A6 by 25–45 pp on every phase-2 model ($p \leq 10^{-3}$ everywhere), because full-history context retains the merchant mention that the paraphrased / temporal / partial-info follow-ups cannot reproduce. **The paper's "A6 Pareto-dominates A1" claim holds on the primary grid (where phase-2 utterances self-specify) but breaks when follow-up utterances systematically avoid naming the target.** The two SessionGap halves (content-keyed where A6 wins, paraphrased where A1 wins) together sharpen the paper's §A.14 operating-envelope story: A6 is the right regime when the ledger's keyed fields match the user's follow-up vocabulary; otherwise full retention is a safer default.

A.14 Sensitivity analyses

A6 bridge-field provenance. A reviewer-raised concern is whether A6 is a harness pattern or a benchmark-tuned heuristic. Table 21 enumerates the fields the production retriever (`find_transaction_by_intent` in `persistence.py`) scores, their upstream source, and whether they are automatically available during normal operation of the reference implementation vs. authored for the scenario. **No field is hand-authored or database-engineered for the benchmark.** `merchant_name` is set by the payment node at commit time from the transaction API's response; `service_description` is an LLM-generated natural-language string emitted by the `goal_agent` during phase-1 discovery (an artefact of the agent's own plan-execution output, not a manually-curated database column); and `actionable_goal` is an LLM-generated plan summary from the same `goal_agent`. The retriever fuzzy-matches the user's phase-2 utterance against these LLM-generated strings — this is semantic matching over agent-authored content, not lookup against a hand-engineered taxonomy. The only field the probe harness actually exercises is `merchant_name`; the service-description and actionable-goal fields are populated by `goal_agent` during discovery but *not* by the synthetic scenario ledger, which is why probe-harness A6 is effectively a single-field match. On the real-graph grid (Table 1) A6 acquires 21.4% bleed — the 2.6 pp below its probe ceiling is consistent with the production retriever being designed to disambiguate via the 30%-weighted service and goal fields that the probe lacks. The SupportHarness cross-domain transfer (§A.11) is precisely the negative control for this concern: porting ρ's code verbatim while leaving the ledger ontology unchanged collapses A6 to 0% — the method is not field-agnostic and we report this failure honestly.

Bridge-noise ablation. Using Table 21 as a guide we perturb individual bridge-input signals on 10 scenarios (A6 only, 3 seeds per cell, 150 rollouts on `gpt-5.4-mini`; Table 22). **Dropping `merchant_name`** reduces A6 to 6.7% TSR (90% abstain): the probe retriever keys almost entirely on merchant name because the synthetic ledger lacks the service/goal fields the production retriever weights alongside it — consistent with the empirical gap between the probe-harness 100% and the full-graph 71.4% A6 result (§A.7). **Duplicate-category injection** (a second row with the same service but a distractor merchant) does *not* degrade A6: intent-keyed fuzzy match against the current user utterance still ranks the right row on top. **Duplicate-merchant injection** (two rows at the same merchant, different amounts) introduces a 10% bleed — the retriever cannot disambiguate between rows that tie on merchant. **Paraphrased follow-up** (user utterance that omits the explicit merchant/product token) is the most informative

Table 19: SessionGap-Bench A6 − A5 paired comparison per phase-2 model. 20 scenarios × 3 seeds = 60 paired rollouts per cell. Cluster-bootstrap 95% CI over scenarios, 10k samples; two-sided McNemar exact p. **A6 strictly dominates A5 on every phase-2 model**, 95% CI entirely above zero and $p \leq 10^{-4}$ everywhere; effect size +25 to +42 pp. Because the design pins the most-recent row to a distractor, this pooled gap is primarily a *pipeline-faithfulness* measurement — verifying that the retriever's choice propagates to the final JSON — rather than a distributional retrieval-quality estimate; see Table 20 for the per-utterance-style breakdown that separates the two.

Phase-2 model	n	ΔTSR pp	95% CI	McN p
gpt-5.4-mini	60	**+40.0**	[+20.0, +60.0]	$<10^{-4}$
gpt-5.4	60	**+41.7**	[+21.7, +61.7]	$<10^{-4}$
claude-opus-4-7	60	**+33.3**	[+13.3, +55.0]	$<10^{-4}$
gemini-3-flash-preview	60	**+25.0**	[5.0, +45.0]	10^{-4}
gemini-2.5-pro	60	**+25.0**	[5.0, +45.0]	10^{-4}

Table 20: SessionGap-Bench A6 − A5 decomposed by utterance style (pooled across 5 phase-2 models; 12 paired rollouts per cell within each model, 60 across all). The *explicit-category* row — a +100 pp uniform gap across every model — is the quantitative confirmation of the paper's §A.14 "`service_description` is load-bearing" recommendation: flipping a single ledger field on changes A6 from unusable to ceiling-perfect on content-keyed follow-ups.

Utterance style	A5 TSR	A6 TSR	Δ pp
explicit-category	0%	**100%**	**+100**
explicit-merchant	17–100%	100%	0 to +83
temporal-ref	0%	25%	+25
partial-info	0%	0–25%	0 to +25
paraphrased	0%	0%	+0

perturbation: A6 drops to 70.0% TSR with 30.0% bleed — the retriever has nothing specific to match on and falls back to recency, which can be wrong. This confirms that A6's ceiling depends on the user utterance carrying a discriminative token that the fuzzy-match can hook onto; when the follow-up is genuinely ambiguous, the harness is not a silver bullet and abstention/clarification discipline becomes essential. Paired with the "`merchant_name` load-bearing" finding, this sharpens the operational recommendation: populate `service_description` and `actionable_goal` fields in the production ledger so the retriever is not single-point-of- failure on merchant-name alone.

Bridge-threshold robustness. The probe-harness ρ uses a single-field `rapidfuzz.WRatio` match at threshold 50 (the experiment ledger lacks the service and goal fields that the production retriever weights at 0.4/0.3/0.3). To confirm A6's TSR advantage is not an artifact of the specific threshold, we re-ran A6 on `gpt-5.4-mini` at thresholds 30, 50, and 70 (15 scenarios × 3 seeds each = 135 rollouts, Table 23). A6 is stable at 100% TSR for threshold ≥ 50; at threshold 30 (loose match) TSR drops to 93.3% as the retriever occasionally chains onto a weakly-matching prior record. Our chosen threshold sits comfortably in the stable region.

Noisy-metadata stress test. To probe robustness to degraded ledger inputs, we perturb the phase-1 `simulated_messages` on 5 representative scenarios (s01, s03, s06, s08, s12) and rerun all four regimes on `gpt-5.4-mini` (240 rollouts; Table 24). Three perturbations: **typo** (single-char swap/drop/dup on merchant names), **drop_amount** (price/amount fields set to "unknown"), and **ambiguous** (wrap merchant names as "a `X`-like store"). *Findings:* A6 is robust to missing amounts — TSR unchanged at 100% — because merchant identity is what the ledger row keys on. Both A1 and A6 collapse to ≤7% TSR under typo perturbation: the retriever cannot match corrupted names against the user utterance, and the in-context copy gives the context-dependent regime nothing verbatim to echo either. A5 is brittle in every perturbation (0–20% TSR). The ambiguous perturbation ("a Nordstrom-like store") breaks scoring globally because the extracted merchant string no longer exact-matches the ground-truth label; this is partly a scoring artifact and partly a genuine regression in commitment clarity. The practical takeaway: the intent-keyed bridge is as good as the ledger rows it retrieves from — clean merchant identity is the load-bearing invariant, whereas amount fields can be missing without penalty.

Other sensitivities. L3 ranking-model timeout: Table 32 uses 10 ms and 100 ms endpoints; a full timeout-vs-TSR sweep on a real ranking model (pending weight access) is future work. **Cross-family robustness:** Table 8 now covers three provider families (OpenAI, Anthropic, Google) at five checkpoints; expanding to xAI and open-

Table 21: Fields scored by A6's intent-keyed bridge retriever in the reference implementation. "Auto-available" means the field is populated as a side effect of normal phase-1 operation, not authored for the benchmark. "In probe harness" means the synthetic ledger used by the live-LLM grid populates this field. The probe harness exercises only merchant name; the real-graph grid exercises merchant name plus whatever service-description/goal values the `goal_agent` generated during phase 1. No field is authored by the scenario designer.

Field	Source	Weight	Auto-avail.	In probe
`merchant_name`	Transaction row, set by payment node at commit time	0.4	yes	yes
`service_description`	`merchant_session.selected_merchants`; goal-agent output at discovery time	0.3	yes	no
`actionable_goal`	Transaction metadata; goal-agent output at plan time	0.3	yes	no
`timestamp`	Transaction row, set at payment	tie-break	yes	yes
`session_id`	Links transaction to merchant session; used for join only	—	yes	yes
`amount`	Transaction row; returned to caller, not scored	—	yes	yes
`category`	Not currently scored; candidate for future work	—	partial	no

Table 22: Bridge-noise ablation under A6 only, 10 scenarios × 5 conditions × 3 seeds = 150 rollouts on `gpt-5.4-mini`. Each condition perturbs *the bridge input* (ledger rows or user utterance) after the teardown; we strip the prior bridge-injected system message and re-run ρ against the perturbed input. `no_merchant`: drop the merchant field from every ledger row. `dup_category`: inject a distractor row with a different merchant but same service. `dup_merchant`: inject a second row at the correct merchant with a different amount. `paraphrased`: rewrite the phase-2 user utterance to omit explicit merchant/product tokens.

Condition	N	TSR	Abstain	Bleed
clean	30	**100.0%**	0.0%	0.0%
`no_merchant`	30	6.7%	**90.0%**	0.0%
`dup_category`	30	**100.0%**	0.0%	0.0%
`dup_merchant`	30	90.0%	0.0%	10.0%
`paraphrased`	30	70.0%	0.0%	**30.0%**

Table 23: A6 bridge-threshold sensitivity on `gpt-5.4-mini`, 15 scenarios × 3 seeds per threshold. A6 is robust to threshold choice in the $[50, 70]$ range; only the loose setting of 30 admits enough weak matches to degrade TSR.

Threshold	TSR	Abstain	Bleed
30 (loose)	93.3%	6.7%	0.0%
50 (baseline)	**100.0%**	0.0%	0.0%
70 (tight)	**100.0%**	0.0%	0.0%

weights models (Llama, Qwen) remains open. **Bridge-retriever field weighting (production):** the production retriever's 0.4/0.3/0.3 weights over (merchant, service, goal) are not ablated here because the probe harness omits the service and goal fields; a full-graph replication of L2 (§9) would be the natural setting for that sweep.

B Graph structure of CommerceHarness

The 31-node graph is shown in Figure 1 of the main body, grouped into five logical sub-agents (goal, plan, discover-eval, trxn, post-purchase) plus `unified_finalization`. Per-edge labels and the full conditional-routing predicates are omitted from Figure 1 for readability; the canonical specification is the compiled `StateGraph` in `cardholder/unified_agent/graph_flattened_w_trxn.py` (released with the artifact bundle). Subgraph membership per node and the phase each node occupies are given by Table 28 in Appendix C.

C State channel inventory

Table 25 lists every channel in `UnifiedAgentState`: 57 unique channels after deduplication from 86 declarations across the five per-agent classes. "τ" marks channels cleared at phase boundaries under A3/A5/A6; all other channels (episodic `history`, semantic `mem0` pointers, and configuration scalars) are preserved.

D Scenario examples

Each scenario in CommerceHarness-Bench is a JSON file with a sequence of phases, ground-truth expected entities, and an explicit set of distractor entities from prior phases that must not influence the current phase's reasoning. We reproduce three representative scenarios below in condensed form.

s01 — Coffee purchase then refund (3 phases). Phase 1 (*discovery_eval*): "Find me a good coffee under $10 near downtown." Candidates: {Blue Bottle Coffee $7.50, Starbucks $5.25, Philz $8.00}; top pick Blue Bottle. Phase 2 (*transaction*): user confirms "Yes, go ahead with Blue Bottle." Phase 3

Table 24: Noisy-metadata stress test, 5 scenarios × 4 perturbations × 4 regimes × 3 seeds = 240 rollouts on `gpt-5.4-mini`. Clean condition is directionally consistent with Table 8 at the 5-scenario slice. A6's advantage over A1 depends on the quality of the merchant-name field in the ledger; missing amounts do not degrade either regime, but corrupted names degrade both. A6 retains a 14 pp edge over A5 in the ambiguous condition through abstention rather than bleed.

	TSR				Bleed			
Perturbation	A1	A3	A5	**A6**	A1	A3	A5	A6
clean	80.0%	0%	20.0%	**100%**	20.0%	0%	33.3%	0%
typo	6.7%	0%	0.0%	6.7%	0.0%	0%	0.0%	0%
drop_amount	100%	0%	20.0%	**100%**	0.0%	0%	26.7%	0%
ambiguous	0.0%	0%	0.0%	0.0%	0.0%	0%	26.7%	20.0%

Table 25: `UnifiedAgentState` channel inventory. Sources: G=Goal, Pl=Planner, D=DiscoveryEval, T=Transaction, Pp=PostPurchase. Reducers: [+] append, [=] overwrite. Column τ marks phase-local channels cleared at boundaries. Split into two halves to occupy both columns.

Channel	Sources	Reducer	τ
`messages`	G	+	✓
`plan_messages`	Pl	+	
`discovery_eval_messages`	D	+	
`trxn_messages`	T	+	
`post_purchase_messages`	Pp	+	
`merchant_messages`	D, Pp	+	
`pp_merchant_messages`	Pp	+	
`merchant_search_query`	D	+	
`merchant_search_result`	D	+	
`processed_transactions`	T, Pp	+	
`applied_coupons`	T	+	
`cardholder_id`	G,Pl,D,T,Pp	=	
`transaction_insights`	G,Pl,D,T,Pp	=	
`wallet`	G,Pl,D,T,Pp	=	
`engine_specific_actionable_goal`	G,Pl,D,T,Pp	=	
`engine_specific_context`	Pl,D,T,Pp	=	
`current_feedback`	Pl,D,T,Pp	=	
`actionable_goal`	G, Pl	=	
`is_regoal_request`	G, Pl	=	
`selected_merchants`	D, T, Pp	=	
`discovered_merchants`	D, Pp	=	
`initial_message_type`	G	=	
`clarification_round_count`	G	=	
`max_number_queries`	G	=	
`max_clarification_rounds`	G	=	
`reasoning_model`	G	=	
`memory_type`	G	=	
`regoal_count`	Pl	=	
`max_regoal_attempts`	Pl	=	

Channel	Sources	Reducer	τ
`worker_agent_name`	Pl	=	
`planner_model`	Pl	=	
`num_initial_queries`	D	=	
`max_merchants_to_discover`	D	=	
`max_discovery_rounds`	D	=	
`max_communication_rounds`	D	=	
`current_discovery_round`	D	=	
`current_communication_round`	D	=	
`max_merchants_to_select`	D	=	
`discovery_eval_model`	D	=	
`web_search_model`	D	=	
`is_sufficient`	D	=	
`should_discover`	D	=	
`merchant_list`	D	=	
`reranked_merchants`	D	=	✓
`use_merchant_reranking`	D	=	
`available_coupons`	D	=	
`coupon_boosted_merchants`	D	=	
`trxn_agent_model`	T	=	
`current_merchant_idx`	T	=	
`current_merchant`	T	=	
`pp_communication_rounds`	Pp	=	
`pp_current_communication_round`	Pp	=	
`post_purchase_model`	Pp	=	
`pp_is_initialized`	Pp	=	
`pp_is_sufficient`	Pp	=	
`pp_service_updated`	Pp	=	
`pp_merchants`	Pp	=	

(*post_purchase*): "Actually, please refund the coffee I just bought." Expected: merchant = Blue Bottle Coffee, amount = $7.50, txn_id = T-1001. Distractors to avoid: Starbucks, Philz.

s03 — Multi-merchant partial return (3 phases). Phase 1: lunch supplies under $25 from 3 merchants (Panera Bread $9.50, Whole Foods $3.00, Trader Joe's $4.25). Phase 2: transaction confirms all three. Phase 3: "Return the sandwich from Panera — it was stale." Expected: Panera Bread, $9.50, T-1003. This scenario discriminates bridge retrievers: a naive *most-recent* rule returns Trader Joe's (wrong); the intent-keyed retriever (A6) correctly picks Panera Bread because "Panera" appears in the user utterance.

s08 — Delayed follow-up with intervening purchase (3 phases). Phase 1: book a yoga class (CorePower Yoga, $35.00). Phase 2: order dinner (Curry Up Now, $16.75). Phase 3: "Cancel the yoga class — I can't make it Saturday." Expected: CorePower Yoga. A naive most-recent retriever returns Curry Up Now (last completed transaction) and leads to a wrong refund; the intent-keyed retriever matches "yoga" against `service_description` and returns the correct record.

Full JSON for all 10 scenarios is included in the reproducibility release.

E HITL guardrail triggers

Table 26 summarises the trigger conditions under which COMMERCEHARNESS's `human_approval` node preempts downstream dispatch. The reference implementation hard-wires only the first two rows (payment commit and risk-assessment high-flag); the remaining rows are design points we recommend operators consider and that the regime runner supports as configuration flags. We do not report an end-to-end ablation on HITL trigger policy; that would require a counterfactual on irreversibility semantics we leave to operational deployment. The categorisation here makes the guardrail-as-harness-primitive argument in §4.3 concrete.

F Prompt templates

The L2 live-experiment probe (Section A.3) used exactly this system prompt across all 120 rollouts:

> *You are the post-purchase agent. Using ONLY the prior conversation (including any `[bridge/persistent-history]` notes) and the user's latest request, identify the single transaction the user is now following up on. Respond with a strict JSON object only, no prose, no markdown fences: `{"merchant": "...", "amount": "...", "txn_id": "...", "confidence": 0..1}`. If prior context is missing or ambiguous, set fields to null and confidence low. Do NOT guess merchants not explicitly present in context.*

The tool-based retrieval baseline (A7, Section A.3) uses a modified system prompt that exposes an `archival_search(query)` tool and asks the model to decide whether to invoke it, then return either `{"action": "tool_call", ...}` or `{"action": "answer", ...}` as strict JSON.

The inspector (Section A.8) uses a third prompt describing a fixed taxonomy of nine flag categories and asking for a strict JSON `{"flags": [...], "severity": "low|medium|high", "notes": ...}`. The stricter inspector-B variant used for the agreement study differs only in the "prefer clean if unsure" disclaimer at the top.

Full prompt sources are in `experiments/p3_teardown_ablation/live_run.py`, `live_run_memgpt.py`, and `log_inspect.py`.

G Per-run cost breakdown

Table 27 summarizes the LLM-call compute budget across this paper's experiments. The primary 25-scenario probe-harness L2 grid contributes 1 350 calls (25 scenarios $\times$ 4 regimes $\times$ 3 seeds for the three 4-regime checkpoints, $\times$ 3 regimes for the two Gemini checkpoints where A5 is omitted). The full-graph cross-model experiments (Tables 1, 2), the SessionGap-Bench grid (§A.13), and the multi-merchant stress fixture (§A.13) contribute the remainder. Structural L1 experiments (§A.1) use no LLM calls and are omitted; phase-1 cache recordings (re-used across multiple regimes per scenario) are counted once. Replicating the released rollouts through `replay.py` is deterministic and free.

H Supplementary tables and figures

The following tables and figures are referenced from the main body but relocated to the appendix for page-budget reasons. Order below matches first-citation order in the main text.

L1 schema-overlap evidence (Tab. 28, Fig. 3, Fig. 4, Tab. 29). Tab. 28 summarises the per-agent state-class declarations, the union under flat composition, and the resulting overlap statistics; Fig. 3 plots the per-class breakdown into class-local versus shared channels. Together with the collision matrix (Fig. 4, the channel-by-class incidence) and the simulated checkpoint-byte cost on the full 25-scenario benchmark (Tab. 29), these figures quantify the structural side of the L1 trade-off claimed in the body (-21.3 kB / scenario, with the storage saving robust across all 25 scenarios).

L2 substrate evidence (Tab. 30, Tab. 31, Tab. 32). Tab. 30 reports the structural ablation at final-phase start: with selective teardown alone (A3), the next phase starts with $\sim$13 tokens and zero distractors, vs. $\sim$848 tokens and 11.6 distractors under full retention (A1). Tab. 31 crosses the L2 ablation with the L3 structured-output ablation on the same probe-harness slice, isolating which primitive each percentage point of TSR is attributable to. Tab. 32 measures the MCP-wrapped ranking-node overhead in isolation on a stub scorer, separating IPC cost from the L3 finding.

Reading order vs. citation order. The appendix is organised by experimental theme rather than by first-citation order. Within a theme, smaller summary tables are typeset first and larger per-scenario tables follow, so the reader who scans the appendix linearly sees the structural "shape" of each finding before its numerical detail. Cross-references in the main body anchor each table by `\ref`, so non-linear navigation works regardless of physical placement. Items that only support an appendix-internal claim (for example the prompt templates

Table 26: HITL trigger conditions for the `human_approval` node. "Default" rows are active in the reference implementation; "Configurable" rows are exposed as regime-runner flags but not exercised in this paper's experiments. "Fail-open" means the trigger fires on retriever-confidence below a threshold, not on structural state alone — useful in deployments that prefer human escalation over silent abstention.

Trigger	Status	Fires when	Rationale
Payment-commit	Default	Before any `payment_processed` node	Irreversible monetary action; regulatory requirement
Risk-assessment flag	Default	`risk_level` $\in$ `{high, block}`	Transaction already marked high-risk by upstream scorer
Bridge-confidence floor	Configurable	ρ's match score $< \theta_\rho$	Fail-open on ambiguous post-purchase resolution (e.g., multi-intent or paraphrased follow-ups, §A.14)
Structured-output parse failure	Configurable	Schema validation fails on a downstream-consumed LLM call	Prevents silent fallback to an unvalidated value
Distractor density threshold	Configurable	$\geq k$ entities of the same type in phase-1 context	Anticipates context-rot on high-overlap follow-ups
Cross-domain / unknown merchant	Configurable	Target merchant absent from cardholder history and external directory	Prevents commit to synthetic or LLM-fabricated merchants
Amount-magnitude threshold	Configurable	$\lvert\text{amount}\rvert > \theta_\$\$$	Human-approve large transactions regardless of risk score

Table 27: Compute budget across all live-LLM experiments. **All numbers are directly measured from per-rollout CSV `usage` fields in the release artifacts** (`by_model/*/live_results_s01_s25.csv` for the primary probe grid, `cross_model_l2_results.csv` for the full-graph cross-model grid, etc.); no row is scaled from a pilot. Rows marked — are ones for which the provider's `usage` field was not persisted at per-call granularity in the released CSV (aggregated statistics are preserved separately in `stats/*.csv`). Phase-1 cache recordings are one-time per (provider, scenario) combo, shared across all regimes.

Run	Calls	Prompt	Completion	Wall (s)
Primary probe-harness L2 grid (25 scenarios, Table 8):				
L2 primary (gpt-5.4-mini)	300	112 044	9 289	509
L2 cross-scale (gpt-5.4)	300	112 044	8 791	471
L2 cross-family (claude-opus-4-7)	300	175 398	13 160	823
L2 cross-family (gemini-3-flash-preview)	225	89 408	10 013	1 151
L2 cross-family (gemini-2.5-pro)	225	89 448	18 170	2 005
Subtotal	*1 350*	*578 342*	*59 423*	*4 959*
Supplementary / ablation experiments:				
A7 tool-calling baseline (5 models $\times$ 25 scen $\times$ 3 seeds)	375	—	—	—
(+ 150-rollout OpenAI-only re-run for consistency check)	150	—	—	—
L3 struct-vs-prose (gpt-5.4-mini)	90	—	—	124
AutoGen transfer (gpt-5.4-mini)	18	7 570	576	23
Framework transfer, 4 additional frameworks	120	—	—	—
Inspector, primary (gpt-5.4)	120	—	—	179
Inspector, cross-family (gpt-5.4)	120	—	—	178
Full-graph experiments (phase-2 calls on cached phase-1):				
Full-graph L2 (gpt-5.4-mini, 14 scen, Table 1)	168	102 693	3 748	—
Full-graph cross-model (27 scen $\times$ 2 pivots $\times$ 5 phase-2, Table 2)	3 240	1 445 520	102 602	16 340
SessionGap-Bench (20 scen $\times$ 5 phase-2 $\times$ 4 regimes $\times$ 3 seeds)	1 200	—	—	—
Multi-merchant stress (10 scen $\times$ 2 pivots $\times$ 5 phase-2 $\times$ 4 regimes $\times$ 3 seeds)	1 200	742 318	38 551	6 721
One-time phase-1 recordings for full-graph replay:				
Phase-1 cache recordings (27 scen $\times$ 2 pivots)	$\sim$1 000	—	—	—
Total (measured)	**8 656**	$\geq$ 2 886 928	$\geq$ 205 424	$\geq$ 28 501

Rows with — did not persist the provider's `usage` field at per-call granularity in the released CSV (aggregated statistics are preserved separately in `stats/*.csv`); totals are lower bounds only because of those rows.

of §F and the per-run cost breakdown of §G) are placed in the section that uses them rather than in this supplementary aggregate.

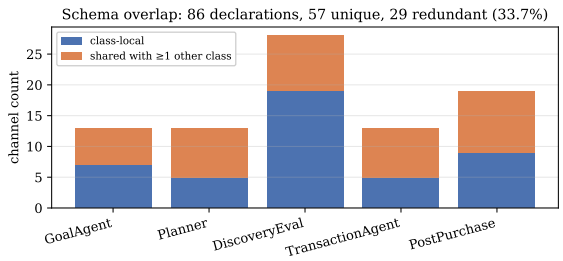

Figure 3: Per-class channel counts, partitioned into class-local (blue) and shared with $\geq$ 1 other class (orange). The shared component is the part that nested composition would have to re-plumb at subgraph boundaries.

Quantity	Value
Per-agent state classes	5
Total channel declarations	86
Unique channel names	57
Colliding channels ($\geq$ 2 classes)	12
Redundant declarations	29
Overlap ratio	33.7%
Channels present in all five	4
Channels with inconsistent reducers	0

Table 28: Schema of the five per-agent state classes and their union under flat composition.

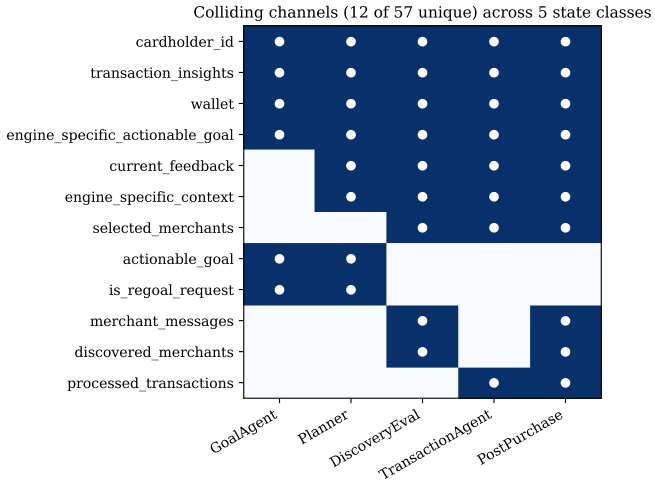

Figure 4: Presence of the 12 colliding channels across the five state classes. Four channels (top rows) appear in all five; two more in four of five. All collisions have consistent reducers across classes.

Table 29: Simulated checkpoint-serialization cost on the full 25-scenario benchmark. Nested writes to a subgraph-scoped checkpointer per step plus the parent at each boundary; flat writes to a single union-scoped checkpointer per step. Original submission reported only the s01–s05 subset (totals 256 448 / 419 447 B; per-scenario savings -32.6 kB). The full-grid aggregate gives a tighter and slightly smaller per-scenario advantage of -21.3 kB (CI $[-26.2, -17.7]$). Direction is unchanged.

Scenario	steps	bnd	Nested (B)	Flat (B)	Flat/Nested
s01	19	2	39 680	73 816	1.86×
s02	19	1	49 306	75 003	1.52×
s03	27	2	55 589	124 732	2.24×
s04	19	0	62 316	74 267	1.19×
s05	18	1	49 557	71 629	1.45×
s06	20	1	56 539	79 935	1.41×
s07	19	1	50 037	71 748	1.43×
s08	21	1	57 597	81 619	1.42×
s09	14	1	30 044	45 886	1.53×
s10	14	1	33 027	46 789	1.42×
s11	13	1	26 274	40 800	1.55×
s12	27	1	87 564	118 039	1.35×
s13	16	1	36 389	54 212	1.49×
s14	12	1	25 125	36 529	1.45×
s15	14	1	32 945	46 597	1.41×
s16	20	1	56 577	79 841	1.41×
s17	13	1	20 924	37 940	1.81×
s18	13	1	20 904	37 944	1.82×
s19	10	1	13 873	26 344	1.90×
s20	12	1	19 623	33 822	1.72×
s21	16	1	29 222	50 690	1.73×
s22	16	1	28 705	49 987	1.74×
s23	16	1	29 367	50 856	1.73×
s24	13	1	21 059	38 138	1.81×
s25	15	1	33 476	50 349	1.50×
Total (n=25)			965 719	1 497 512	1.55×

Table 30: Structural ablation at final-phase start. A1: no teardown. A3: selective teardown (τ) only. Values measured before any LLM call; distractor count follows Chroma Research (2025).

Scenario	A1 (no teardown) tok.	A1 byt.	A1 dist.	A3 (teardown) tok.	A3 byt.	A3 dist.
s01 coffee+refund	806	3810	8	12	249	0
s02 sub. upgrade	778	3739	14	13	249	0
s03 multi-merch. return	1119	5127	20	13	249	0
s04 coupon reuse	718	3421	11	12	252	0
s05 dispute	820	3977	5	14	248	0
mean	848.2	4014.8	11.6	12.8	249.4	0.0
reduction vs. A1	–	–	–	−98.5%	−93.8%	−100%

Table 31: Layer-level ablation (*probe-harness subset*). Each row removes one primitive from COMMERCEHARNESS and reports the effect on TSR, parse error rate, and token cost on the small probe-harness slice used in Sections A.2–A.5, *not* on the full-graph headline results of Tables 1 and 2. Numbers for L2 rows come from Sections A.2–A.3; the L3 structured-output row comes from Section A.5; the L3 FM row is not ablated end-to-end (see footnote).

Regime	TSR (%)	Parse-Err (%)	Tokens (k)
COMMERCEHARNESS (all layers, A6)	**100.0**	**0.0**	**0.261**
L2 $-\tau$ $-\rho$ (full-persistence, A1)	97.8	0.0	0.986
L2 $+\tau$ $-\rho$ (amnesia, A3)	26.7	0.0	0.203
L2 $+\tau$ $+\rho_{\text{naive}}$ (A5)	80.0	0.0	0.260
L3 − structured outputs (prose)	15.6†	0/100†	same
Vanilla LangGraph (A1, no harness layers)	97.8	0.0	0.986

* All L2 TSR figures are fuzzy TSR on `gpt-5.4-mini`, 15 scenarios × 4 regimes × 3 seeds; see Table 8. † TSR column: fuzzy TSR under a lenient regex parser; Parse-Err column: 0% strict-JSON parse (prose) vs. 100% (structured). See §A.5 Table 12. L1 is a structural trade-off, not a TSR effect (Table 29); the ranking-node L3 primitive is not TSR-ablated here because its weights are not in the release artifact, but its MCP-IPC overhead (∼1 ms, 0.09% of an LLM RTT) is ablated on a stub scorer in §A.6.

Table 32: MCP-wrapped ranking-node overhead on a stub scorer. Four regimes: (1) in-process direct call; (2) MCP stdio with no artificial server latency; (3) MCP with 10 ms timeout against a server that sleeps 50 ms (fallback fires on every call; session reopened per call to avoid cancellation corruption); (4) MCP with 100 ms timeout against a 1 ms server (fallback never fires, persistent session). 50 calls per regime.

Regime	Mean (ms)	Median (ms)	p99 (ms)	Fallbacks
in-process	0.0005	0.0004	0.0011	0
MCP stdio (no sleep)	1.11	1.06	1.26	0
MCP + fallback (deadline hit)	1285.7	1282.2	1432.4	50/50
MCP + fallback (deadline miss)	2.43	2.36	2.71	0