THE TOOL-AGENT OBSERVABILITY BOUNDARY: PREDICTING RECOVERABLE TOOL-CALL FAILURES BEFORE ANY RECOVERY RUN

Anonymous authors
Paper under double-blind review

ABSTRACT

Adding more validation to a tool-using agent, whether schemas, entity checks, ranges, or self-consistency, feels like it should always help. It does not, and which failures it can reach is decidable in advance. Tool-output reliability is governed by an **observability boundary**: some corruptions leave evidence in the returned observation, and some cannot. A wrong-entity, partial, or malformed response violates a formal invariant and is recoverable. A **stale-valid** observation, a real past state of the same entity, is schema-valid, id-consistent, in-enum, and in-range. We prove a **no-go theorem**: no gate in the non-temporal invariant family (schema, id-echo, enum, range) separates it from a fresh record, so more reasoning or a larger model provably cannot cross the boundary. A freshness sweep confirms this: recovery falls to 0% when the verifier returns only same-cache evidence.

Our contribution is to make the boundary *predictable* rather than to assert it. **LOCO** assigns per-observation causal criticality labels by leave-one-corruption-out replay, and shows that the obvious structural proxies fail unpredictably: read→write recall ranges from 0.0 to 1.0 across domains, so the boundary cannot be eyeballed. **StructAudit** compiles admission predicates from tool schemas and clean examples alone, then emits a frozen *coverage map* that places each corruption class on the correct side *a priori* and label-free, at 87.9% binary recoverability agreement across three causal domains (72.7% under the stricter three-class scoring) and predicate-level precision 1.000 with recall 0.700 on retail. **FreshVERIFY** supplies the fresh read where the map predicts a freshness limit.

Measured against the strongest cheap gate rather than the weakest, one invariant does most of the work: on gpt-oss-120B, id-echo alone matches or exceeds the full predicate family on recovery, and it is the most robust cheap gate net of lost clean rescues; where the fuller family recovers more, it does so at higher rescue-loss, and declared-schema validation trails both. This corroborates the boundary rather than undercutting it, since the recoverable corruptions are exactly the entity-observable ones. The resulting operating rule for tool-output admission: validate what is observable, verify what is not, and do not expect in-trajectory reasoning to manufacture missing freshness. Whether the same informational constraint governs verification signals more broadly remains open; we settle it where it can be measured.

1 INTRODUCTION

Tool-augmented agents (Yao et al., 2023; Schick et al., 2023) increasingly make decisions from external observations: retrieved records, account states, reservations, bills, and API responses. When those observations are corrupted, the agent may act confidently on the wrong world state. The standard reliability instinct is to add a guardrail before the observation reaches the model: validate the schema, check the entity id, reject out-of-range values (Rebedea et al., 2023; Willard & Louf, 2023), or ask the model to self-verify (Madaan et al., 2023; Dhuliawala et al., 2024). These interventions are useful, but they answer the wrong first question. The first question is not "which validator should we use?" It is *whether the corruption is observable from the returned observation at all*.

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

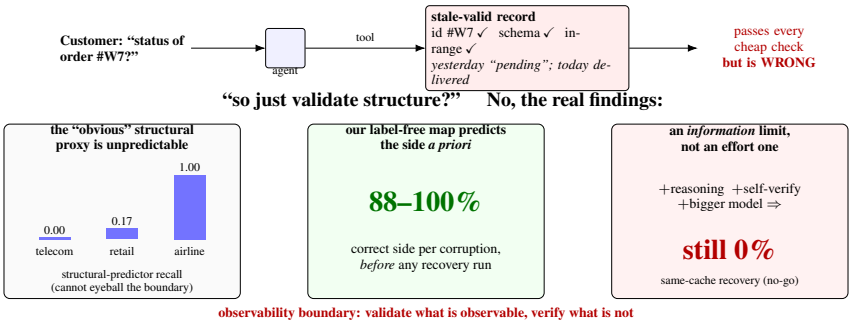

Figure 1: **The boundary is not the obvious part.** A stale-valid tool observation passes every cheap structural check yet is wrong (top). The non-obvious findings: **(left)** the "obvious" structural proxy for recoverability is wildly domain-dependent (read→write recall $0.00/0.17/1.00$ on telecom/retail/airline), so one cannot eyeball which corruptions are recoverable; **(middle)** StructAudit's label-free map predicts the correct side *a priori*, before any recovery run (88–100%); **(right)** the limit is *informational*: more reasoning or a larger model cannot cross it (0% recovery at same-cache; §6).

This paper argues that tool-output reliability has an **observability boundary**. On one side are corruptions that leave evidence in the observation: the wrong entity echoes the wrong id, a partial record is missing required fields, and a malformed response fails to parse. Cheap admission control can detect and recover these failures. On the other side are corruptions that preserve all local evidence. A stale-valid record is a real past state of the same entity; it can be schema-valid, id-consistent, in-enum, and in-range. No amount of in-trajectory computation can infer the current state from evidence that does not contain it. The missing resource is freshness.

This boundary matters because it changes the design problem. Without it, an agent developer sees a menu of reliability techniques and a noisy end-task success number. With it, the design becomes a routing problem: *validate* observations whose corruption is locally observable, *verify* observations whose correctness depends on fresh state, and avoid spending verification budget where a cheap gate is already decisive. It also changes evaluation. End-task success cannot tell whether a gate worked for the right reason, destroyed a lucky rescue, or was simply spending verification at random. We therefore need causal labels for the corrupted observations that actually break the task.

We make three contributions:

- **A causal measurement protocol. LOCO** assigns per-observation causal criticality labels by leave-one-corruption-out replay with a matched no-op control. It shows why structural proxies are insufficient: read→write linkage recall is 0.13–0.17 in retail, 0.0 in telecom, and 1.0 in airline.
- **A formal boundary.** We prove a no-go theorem for stale-valid observations: any gate built from non-temporal schema/type, id-echo, enum, and range predicates gives the same verdict on fresh and stale-valid records. Freshness is not an implementation detail; it is the information needed to cross the boundary.
- **A predictive operating rule. StructAudit** compiles label-free predicates and a frozen coverage map that predicts which corruption classes are formally recoverable before any recovery run. **FreshVERIFY** supplies the complementary fresh read where the map predicts a freshness limit.

Why this is more than "stale data is hard." That intuition is folklore; the contribution is to make the boundary predictable and falsifiable. From tool structure alone, before any recovery run, StructAudit's coverage map places each corruption on the correct side, label-free, while the obvious structural proxies do not: read→write recall ranges from 0.0 to 1.0 across domains. Because the limit is informational rather than computational, more in-trajectory reasoning cannot cross it (Wang et al., 2023). The headline is the causal recoverability map, LOCO's per-observation labels together with StructAudit's label-free predictor of them; the no-go theorem is the mechanism behind one side of that map, not a claim to have discovered that freshness matters. Empirically the frame predicts behaviour across domains and models: per-break agreement is 100% on retail and 87.9% across

retail, telecom, and airline (72.7% three-class), the gate beats declared-schema validation on retail while preserving clean tasks, transfers zero-shot to telecom, replicates on Qwen-32B, and recovery decays monotonically to 0% as verifier freshness degrades to same-cache.

2 RELATED WORK

Tool-use agents and benchmarks. Tool/API agents (Yao et al., 2023; Schick et al., 2023; Patil et al., 2024; Qin et al., 2024) are scored by end-task-success suites (τ-bench (Yao et al., 2024), τ^2-bench (Barres et al., 2025), AgentBench (Liu et al., 2024b), WebArena (Zhou et al., 2024)): these measure *whether* an agent succeeds, not *which* corrupted observation breaks it; LOCO fills that gap.

Hallucination, self-verification, uncertainty. Methods that flag or quantify model error (self-consistency (Wang et al., 2023), SelfCheckGPT (Manakul et al., 2023), semantic entropy (Farquhar et al., 2024), Self-Refine (Madaan et al., 2023), Chain-of-Verification (Dhuliawala et al., 2024)) reason from in-trajectory or parametric information; a stale-valid observation looks self-consistent and low-uncertainty, so it sits in their blind spot (the no-go region needing fresh evidence).

RAG robustness. Work on noisy/irrelevant/stale retrieval (corrective and self-reflective RAG (Yan et al., 2024; Asai et al., 2024), robustness to irrelevant context (Yoran et al., 2024), positional effects (Liu et al., 2024a), the effect of noise (Cuconasu et al., 2024)) targets single-shot QA; we add causal labels and a recoverability boundary for tool observations inside multi-step trajectories.

Runtime validation, guardrails, enforcement. Invariant inference / data validation (Ernst et al., 2007; Great Expectations Labs, 2024; Breck et al., 2019) and guardrails / constrained decoding (Rebedea et al., 2023; Willard & Louf, 2023) are not tied to agent task outcomes and do not characterize an observability boundary; StructAudit applies label-free schema-inference-plus-checking to *runtime tool observations* and ties each predicate to causal recoverability (GATE-A isolates the lift over declared-schema validation).

Tool-call gating and contract verification. Recent agent-tooling work gates or verifies tool use: contract-grounded execution with Hoare-style pre/postconditions (Liu et al., 2026), deterministic pre-execution gates that block policy-violating write actions (Reddy et al., 2026), schema-gated orchestration that blocks execution until a complete workflow validates against a specification (Strickland et al., 2026), and weak-versus-strong verification policies that decide when a cheap check suffices versus deferring to a human (Kiyani et al., 2026). These enforce or prevent at selection/execution time, or target reasoning steps; none predict, label-free and before running, which corrupted *tool observations* are recoverable, and none establish a freshness no-go. Benchmarks of staleness detection (Chao et al., 2026) corroborate our no-go: agents struggle to act on later evidence that invalidates stored data. StructAudit is an instantiation of the observability boundary, not the contribution: the contribution is the boundary itself (the causal recoverability map and the no-go theorem).

Causal failure attribution. Recent work attributes multi-agent task failures to a step or agent using LLM-judge methods (Zhang et al., 2025). LOCO instead performs a controlled leave-one-corruption-out counterfactual intervention with a matched no-op control, yielding ground-truth per-observation labels rather than a model's judgment.

Concurrent work. Several preprints inside the concurrent-work window reach this boundary from other directions. Zheng et al. (2026), on cost-aware commit gates for state-changing actions, find that a verifier's evidence source matters far more than its model diversity: a cross-model vote over shared evidence approves 62.9% of unsafe proposals against 22.9% with an independent source, a 40.9-point effect against 11.3. That corroborates Theorem 1 empirically on a different task, without an impossibility argument. Mazumder & Lia (2026) find stale-data faults resist mitigation in an MCP fault-injection workbench while timeout faults are repaired by retry, replicating our freshness limit in another harness. Shraga et al. (2026) study staleness of a guardrail *verdict* between check and use; the stale object there is the approval and the hazard a time-of-check/time-of-use race, not the indistinguishability we prove. Wan et al. (2026) survey robot-policy verifiers and find credibility falls as availability rises; our boundary gives that tradeoff a decidable criterion for tool outputs.

3 EXPERIMENTAL SETUP

Environments. τ-bench (retail, airline) (Yao et al., 2024) and τ^2-bench (telecom, ported into the same harness) (Barres et al., 2025). Per-domain task counts and clean-solvable sets (a task is *clean-solvable* if the agent succeeds with no corruption): retail 115 tasks (clean-solvable $29/115$ @14B seed-42; $41/115$ @32B); airline 50 (7 solvable); telecom 28 ($28/28$ solvable). **Break-recovery** analyses are conditioned on the *clean-solvable* (clean-PASS) set, since only those can be "broken," while **rescue** accounting is computed separately over the clean-FAIL tasks in the same task pool, since corruption can occasionally turn a clean failure into a lucky pass.

Models and inference. Qwen2.5-14B-Instruct (primary) and Qwen2.5-32B-Instruct (robustness) (Qwen Team, 2024), run locally; **greedy decoding (temperature** 0**)** for both the agent and the LLM user-simulator, so the pipeline is fully deterministic given a seed (verified: clean re-runs reproduce exactly) and corruption is the only source of trajectory divergence. Hardware: A100-80GB (14B fits one GPU; 32B ≈62GB/GPU). Seeds: s42–s50 (corruption RNG).

Corruption model. On every *entity-lookup* tool observation, with probability **qc** (the "corruption rate"), inject one of {`stale` (a real past value of the same entity), `wrong_entity` (another entity's record), `partial` (drop decision-critical fields), `malformed` (unparseable)}. Corruption is **entity-keyed**: the RNG is seeded by (seed, task, entity, occurrence), making it *trajectory-invariant*: suppressing one entity's corruption leaves all others bit-identical (this is what makes LOCO a clean counterfactual). A `content_differs` gate ensures a corruption counts only if it changes parsed content (a stale no-op on a field with no stale-able value is not a corruption). VERIFY = re-fetch the clean observation (oracle); the freshness sweep returns the current value w.p. p_{fresh}, else a same-cache (stale) value.

4 METRICS

Break metrics are computed over clean-PASS tasks and rescue metrics over clean-FAIL tasks from the same pool. The **break set** is the clean-PASS tasks that fail under corruption, which a gate should recover; the **rescue set** is the clean-FAIL tasks that pass under corruption, lucky passes a gate may destroy. We report **break-recovery rate** (recovered / |break set|, higher better), **rescue-loss rate** (lost / |rescue set|, lower better), **clean-preserved** (clean-PASS tasks still passing at noise=0, the false-flag cost), and **cost**×, the tool-call multiplier against the `none` policy, for which `verify_all` is a 2× oracle ceiling rather than a deployable baseline. We also check that a gate beats a **cost-matched random** flagger, that is, that it outperforms spending the same verify budget at random. **Net-utility** (recovered − lost) is reported with a rescue-loss column but demoted from the headline: it conflates detection quality with the seed-random rescue population (§8.1).

5 MEASUREMENT: LOCO CAUSAL CRITICALITY

Protocol. Corruption is entity-keyed and trajectory-invariant (seeded by (seed, task, entity, occurrence)), so suppressing one entity's corruption while preserving the RNG stream yields a clean counterfactual. A break is *causally attributable* to an observation if suppressing only that corruption flips the task fail→pass. A **matched no-op control** replays with all corruption suppressed (must pass).

Results (retail, Qwen-14B, s42). At qc_30, seed 42 has 9 breaks, 8 single-attributable, and 1 cascade; at qc_50 it has 12 breaks, 11 single, and 1 cascade; no-op control 100% (0 violations) in every seed. Pooled across seeds s42/s43/s44 (n=3): single-attribution $=43/51=84\%$ (per-cell range 67–100%), so most failures concentrate in one observation, with a minority of cascades at higher corruption. The structural read→write linkage predictor has precision 0.5–0.67 but **recall** 0.13–0.17 (s42): structural proxies under-recall, and LOCO is necessary. Per-break labels (s42 qc_50): wrong_entity ×7, partial ×3, stale ×2. LOCO validity replicates cross-domain (§E.1): telecom no-op control 100%, all breaks single-attributable, structural predictor recall =0.0 on telecom (a second domain where structure-only fails); airline LOCO confirms it again with the opposite structural contrast (§E.3).

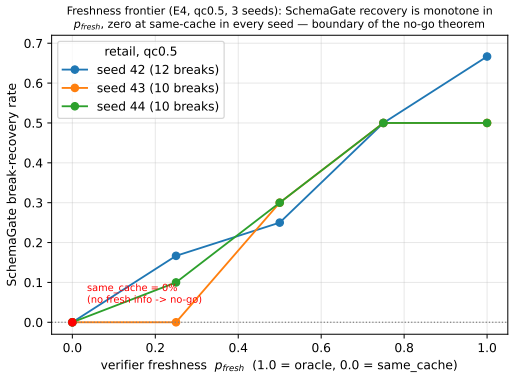

Figure 2: **Freshness frontier (E4), three seeds.** StructAudit break-recovery vs verifier freshness p_{fresh} (retail, qc0.5, seeds 42–44). In every seed recovery is monotone non-increasing and reaches 0% at same-cache (s42 $67\to50\to25\to17\to0$; s43 $50\to50\to30\to0\to0$; s44 $50\to50\to30\to10\to0$). The same-cache zero is the empirical boundary condition of the no-go theorem (§6); per-point rates are draw-sensitive but the frontier shape and the same-cache corner replicate.

6 The Observability Boundary and No-Go Theorem

A *formal* gate is any predicate over (query, in-trajectory history, observation) with no fresh external read. StructAudit's predicate family $F = \{\text{schema/type, id-echo, enum, range}\}$. The formal setup and complete proofs are in Appendix A.

Lemma 1. *A stale-valid observation $o'{=}\text{render}(S_{t'}(e))$ is admitted by every $p \in F$: it is a real past record of the same entity, hence schema-valid, id-echoing, in-enum, and in-range.*

Theorem 1 (no-go)**.** *Every gate built only from F returns the same verdict on the fresh o and the stale-valid o'; on the stale sub-population its detection power equals its false-positive rate.*

Corollary 1. *Stale recovery requires a verifier whose output has positive mutual information with the* current *state; expected recovery is monotone in verifier freshness p_{fresh} and is 0 at $p_{fresh}{=}0$.*

The statement is intuitive, but its value here is operational: it makes "cannot detect stale data locally" a *measurable, predictable* property inside agent trajectories: LOCO labels which corruptions actually break tasks, and the coverage map (§7) locates this boundary label-free *before* any recovery run, telling the designer which corruptions to route to verification rather than gating.

Empirical control (E4, FreshVERIFY-on-StructAudit). We sweep the verifier's freshness $p_{\text{fresh}} \in \{1.0, 0.75, 0.5, 0.25, 0.0\}$ on the StructAudit policy (retail, two corruption rates, seeds 42–44). Break-recovery is monotone in p_{fresh} and collapses to 0 at same-cache at both rates in every seed, with clean-pass preservation $29/29$ throughout (Figure 2). The fully stale corner is net-negative in all six seed$\times$rate cells: it recovers nothing yet still loses latent clean rescues (net -3 to -5 at qc0.3, -2 to -6 at qc0.5), so gating on cached evidence without freshness is dominated by the no-op policy. This is the no-go theorem's boundary measured directly: freshness, not flagging, is the causal resource. What does *not* replicate is where the break-even sits: at qc0.5 net holds positive down to $p_{\text{fresh}}{=}0.5$ in s42, flips there in s43, and is negative throughout in s44. Per-point rates are draw-sensitive at 4–12 breaks per cell, so we report the frontier shape and the same-cache anchor, which replicate, and treat the break-even as seed-dependent, which independently corroborates net-utility's demotion (§8.1).

Scope. The no-go is over **non-temporal** invariants of the query, in-trajectory history, and returned observation: schema validity, entity-echo, enum membership, and value ranges. It is not the claim that no predicate can ever detect staleness. A trusted freshness-bearing signal, such as a monotone version counter or timestamp, or any external read carrying mutual information about current state, can flag a stale-valid record and lies outside F. StructAudit deliberately does not compile such temporal invariants, since they require a reliable last-seen value; naming the case identifies the one in-trajectory escape and why we route stale observations to FreshVERIFY. We use *observability*

boundary for the system-level phenomenon and *information boundary* for the mechanism in the theorem.

7 METHOD

StructAudit compiles, *label-free*, from a tool's schema + clean example records: required/type predicates, id-echo (entity-consistency), enum sets, and numeric ranges. It mines freshness-candidate (mutable) fields *before* value predicates and excludes those fields from enum/range checks. The runtime gate is **3-valued** (PASS/FAIL/UNOBSERVABLE): for example, it abstains via `echo_absent` when an id is not echoed in the record body, avoiding the id_match false positive seen on user records. It also emits a per-tool **coverage map** that classifies each corruption type as formal-detectable, freshness-limited, or unobservable. **FreshVERIFY** supplies fresh evidence where the map predicts freshness limits.

Compilation (per tool, label-free). From the domain's clean entity records alone, with no task outcomes, StructAudit mines three predicate families and freezes them before any recovery run: a three-valued *echo* predicate per lookup key, which fails on mismatch where records reliably echo the key and abstains to UNOBSERVABLE where they do not, removing the `id_match` false positive on records such as users; *completeness and type* from fields present in almost all examples; and *enum or range* over the remaining value fields, which are the predicates a declared JSON schema cannot express. Mutable fields are routed to the freshness-limited set *before* any value mining and never receive value predicates, so they reach FreshVERIFY rather than ever failing. The *coverage map* then classifies each corruption type per tool: wrong_entity is formal-detectable iff an echo predicate exists, partial iff required fields were inferred, malformed always, and stale is freshness-limited whenever the tool exposes a mutable field. A zero-shot variant predicts a held-out tool's map from field names and fixed priors alone, which is what the LOTO and cross-domain transfer results use. Thresholds and the full procedure are in Appendix C. The runtime gate (`flag(tool, kwargs, observation)`) applies the frozen predicates in order and returns FAIL on the first violation ($\rightarrow$ trigger VERIFY), else ADMIT.

8 EXPERIMENTS

As a first check (E2, retail s42), StructAudit improves on a hand-coded L1 gate while preserving clean behaviour: qc_30 $4/9$ recovered against L1's $3/9$, qc_50 $8/12$ against $5/12$, clean-preserved $29/29$ via three-valued abstention, at 1.2–$1.31\times$ cost against `verify_all`'s $2\times$.

8.1 BEYOND DECLARED-SCHEMA VALIDATION (GATE-A; RETAIL; n=4 SEEDS)

Against `naive_schema` (completeness + enum + malformed; **no** id-echo, **no** inferred range, i.e., declared-schema validation), the breaks-recovered margin is seed-invariant (Table 1). A paired sign test (SA>naive in $8/8$ seed$\times$rate cells) gives one-sided p=0.0039; clean-preserved $29/29$ for all policies and all 4 seeds. The lift is id-echo (entity-consistency) + inferred-range, predicates a declared JSON schema cannot express. *Methodological finding:* net-utility-vs-none is *not* the right headline, because it conflates detection quality with a seed-random *rescue* population (range $+6\rightarrow-3$, CIs straddle 0) while the SA$-$naive Δ barely moves. We therefore report $\Delta\pm$CI and population-normalized recovery / rescue-loss rates (Figure 3).

Against the strongest cheap gate, not the weakest. `naive_schema` is the declared-schema ablation, so it is the *weakest* cheap gate; reporting only against it would overstate the case. We therefore also report `id_match` (id-echo alone) on all eight retail seed$\times$rate cells. The two metrics disagree, and the disagreement is informative. On *breaks recovered*, StructAudit is $\geq$ `id_match` in $8/8$ cells (6 strict wins, 2 ties). On *net* (recovered minus lost clean rescues), `id_match` is non-negative in $8/8$ cells (0 to $+4$) while StructAudit is negative in $4/8$ (-3 to $+6$): the enum and inferred-range predicates buy extra detections and pay for them in rescue-loss. StructAudit's lift over declared-schema validation is thus attributable chiefly to the id-echo predicate, and on net the two are comparable. We keep recovery as the primary metric because a missed corruption propagates into a state-changing action while a lost rescue only forfeits an accidental success, but we report net rather than omit it, and we do not claim StructAudit dominates `id_match`.

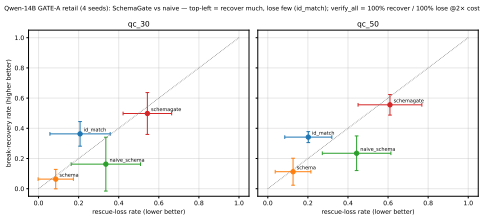

Figure 3: **Recovery-rate vs rescue-loss-rate**, 14B GATE-A: StructAudit recovers 50–56% against naive's 16–23%. y = recovery (higher better), x = rescue-loss (lower better). `id_match` sits at low loss and moderate recovery; StructAudit recovers most but at higher rescue-loss, so the axes are a genuine tradeoff rather than a single dominant corner, which is why net conflates them (§8.1). The Qwen-32B panel is Figure 5.

	qc_30	qc_50
SA−naive Δ (mean)	**+2.00**	**+3.25**
per-seed Δ	+2, +2, +2, +2	+3, +4, +3, +3
SD	0.00	0.50
95% t-CI (df=3)	[2.00, 2.00]*	[2.45, 4.05]

Table 1: GATE-A: StructAudit − naive_schema breaks recovered, 4 seeds. *qc_30 is degenerate (zero observed variance: Δ=+2 in every seed); read as "exactly +2 in all 4 seeds," not as a tight variance bound (n=4 cannot bound tail variance). Sign test over 8/8 cells: p=0.0039.

8.2 Cross-family generality (gpt-oss-120b)

To test whether StructAudit's advantage over declared-schema validation is Qwen-specific, we replicate GATE-A on **gpt-oss-120B** (OpenAI, 2025) (OpenAI; a Mixture-of-Experts model using the harmony format, architecturally distinct from Qwen, served locally via vLLM). It is a strong tool-caller on retail (clean-solvable 48–50/115 across seeds, above Qwen-14B's 29), giving a healthy break set. Under the identical protocol (retail, entity keying, oracle verify), config-consistent across three seeds and two corruption rates, **StructAudit recovers more corruption-induced breaks than `naive_schema` in all six conditions** (margins +1 to +6; Table 2). Pooled across cells, the discordance is asymmetric (StructAudit-only 29 vs naive-only 7): **McNemar** p=0.0005 (exact binomial 0.0003), *stronger* than the Qwen-14B effect (p=0.0039). The schema-admission advantage therefore replicates on a second, architecturally distinct model family. A clean-pass (qc=0) re-run showed run-to-run variance of ±5–10 tasks/policy from the LLM user-simulator, so we report cross-family recovery generality but do not re-assert the zero clean-false-positive property on gpt-oss. (net-utility not run, consistent with its demotion; 42 verifier-parse failures across cells are counted as task failures, which can only understate StructAudit.)

The id-echo predicate carries most of the recovery. Measured against the *strongest* cheap gate rather than the weakest, `id_match` (id-echo alone) recovers a mean 42.7% of breaks on gpt-oss, against StructAudit's 38.0% and `naive_schema`'s 23.1% (Table 2); it exceeds StructAudit in 3/6 cells, ties in 1, and trails in 2. StructAudit's lift over declared-schema validation is therefore attributable primarily to the id-echo component, with enum and inferred-range buying additional detections at the cost of additional rescue-loss (§8.1, Figure 3). We read this as *support* for the observability boundary rather than as a weakness of the instantiation: the recoverable corruptions are exactly the entity-observable ones, and id-echo is the predicate that reads entity observability most directly. Accordingly, the McNemar p=0.0005 above is a claim about StructAudit versus declared-schema validation — the weakest cheap gate — and we do not present it as a win over `id_match`.

8.3 Coverage map predicts the boundary

We compare each break's predicted class from the frozen coverage map against the observed cheapest-recovering mechanism. **Retail: binary agreement** 100% (19/19); predicate-level pre-

seed	rate	breaks	id_match	naive_schema	**StructAudit**	SA−naive
42	qc0.3	28	8	8	10	+2
42	qc0.5	23	8	4	8	+4
43	qc0.3	25	14	7	11	+4
43	qc0.5	27	10	6	11	+5
44	qc0.3	20	10	6	7	+1
44	qc0.5	24	12	3	9	+6
mean recovery rate			42.7%	23.1%	38.0%	

Table 2: Cross-family generality: GATE-A on gpt-oss-120B (retail, breaks recovered), config-consistent across 3 seeds × 2 rates. StructAudit > `naive_schema` in all $6/6$ cells (margins $+1$–$+6$); pooled McNemar p=0.0005 (exact 0.0003), replicating the Qwen-14B result (Table 1) more strongly on an architecturally distinct (OpenAI MoE) family. We also report the *strongest* cheap gate: `id_match` (id-echo alone) has the highest mean recovery rate here (42.7% vs StructAudit 38.0%), consistent with the boundary claim that the recoverable corruptions are the entity-observable ones.

cision 1.000, recall 0.700, F1 0.824 over the 12 predicate-labelled breaks. Every predicted-formal break is formal (0 FP); the 3 FN are `get_user_details` wrong_entity breaks where the id is *not* echoed, correctly routed to unobservable/verify_all, and the stale pair are true negatives. **Multi-domain (retail + telecom + airline, all three with causal LOCO labels): binary agreement 87.9% (29/33 scored)**: 33 causal breaks across the three domains; the 4 misses are multi-corruption-type / boundary tasks (e.g. partial+stale, where the formal gate recovers the partial component the map labels freshness, or vice versa). The coverage map was frozen *before* the recovery runs, so this is genuine prior prediction (Figure 4); agreement degrades gracefully from single-domain (retail 100%) to the harder cross-domain mix, with the residual entirely on multi-type boundary breaks.

We report two scorings of the same 33 breaks. The operationally relevant question is binary, can a cheap gate recover this break at all, on which the map agrees 87.9% (29/33). Scored as an exact three-way class match over {formal-detectable, freshness-limited, unobservable}, agreement is 72.7% (24/33), the figure's printed number. The gap is one confusion: five breaks the map calls unobservable are in fact recovered by fresh evidence. That error is conservative, routing work to the costlier verifier rather than admitting a corrupted observation, but it is a real misprediction and the stricter number is the honest one when the question is the exact regime rather than the side of the boundary.

8.4 Label-free generalization (E5/LOTO)

In leave-one-tool-out zero-shot coverage prediction, StructAudit reaches 94% agreement (15/16). The lone miss is an inferred enum that requires examples, which is expected because enum/range predicates are not predictable from field names alone.

8.5 Additional domains and models

Three further settings appear in full in Appendix E. **Telecom** (τ^2-bench, a structurally distinct held-out domain) tests zero-shot transfer: the coverage map agrees 20/20 model-free, and at qc0.9 over n=9 seeds the StructAudit−`naive_schema` margin is positive in every seed (mean +1.67, 95% t-CI [1.12, 2.21], p=0.0020) with clean-preserved 28/28. **Qwen-32B** shows the boundary is model-dependent: the formal region shrinks as failures shift toward content and freshness, and `id_match` is again the robust low-cost operating point. **Airline** is freshness-dominated, the regime the map predicts, with formal gates net-negative and `verify_all` at +3. Together the three give the structural-proxy contrast the abstract rests on: read→write linkage recall is 0.0 on telecom, 0.13–0.17 on retail, and 1.0 on airline. A proxy ranging from useless to perfect cannot site the boundary in a new domain, which is what LOCO is for.

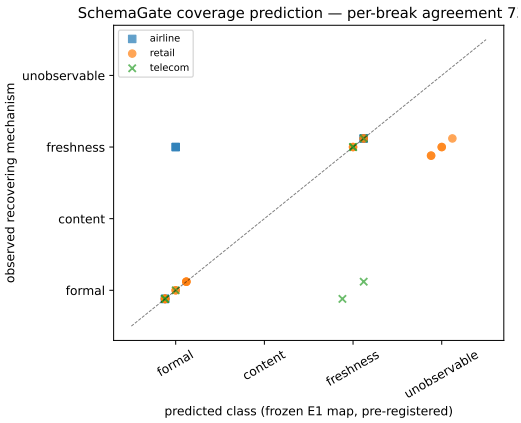

Figure 4: **Coverage prediction over three causal domains.** Predicted formal/content/freshness class (frozen coverage map) vs observed cheapest-recovering mechanism, over 33 causal LOCO breaks across retail + telecom + airline. Binary cheap-gate-recoverability agreement degrades gracefully from retail 100% $(19/19)$ to 3-domain 87.9% $(29/33)$; the 4 misses are multi-type boundary breaks. The panel title reports the stricter exact three-class agreement, 72.7% $(24/33)$; the difference is the five unobservable-predicted breaks that fresh evidence in fact recovers. The map is frozen before the recovery runs (prior prediction).

9 Limitations

Model coverage is cross-family but bounded: the GATE-A advantage holds on two architecturally distinct families (Qwen-14B/32B and gpt-oss-120B, McNemar $p{=}0.0005$; §8.2), but the frontier and no-go experiments are Qwen-only. The cross-domain margin is directional rather than powered: the telecom StructAudit>naive margin sits at qc0.9 ($n{=}9$, $p{=}0.0020$) where naive≈random, so the load-bearing cross-domain claim is StructAudit≫random, and retail ($n{=}4$, $p{=}0.0039$) is the primary powered result. Break populations are small (telecom 2–8 per seed, airline 3), so LOCO grounds the detectors but does not give powered effect sizes, and per-cell claims are uncorrected. The freshness frontier is qualitative: its monotonicity, the same-cache 0% corner, and zero clean false-positives replicate across three seeds, but per-point rates are draw-sensitive at 4–12 breaks per cell, so we report frontier shape rather than tight per-point intervals. Finally, compiler invariant-inference is not itself novel (Ernst et al., 2007; Great Expectations Labs, 2024); the contribution is the observability framing around it.

10 Conclusion

The interesting failure mode in tool-using agents is not that observations can be corrupted, but that some corruptions are locally visible and others are not. That asymmetry is an observability boundary. LOCO measures which corrupted observations causally matter, the no-go theorem explains why stale-valid records survive every non-temporal predicate, StructAudit predicts the recoverable side label-free, and FreshVERIFY supplies the missing evidence on the other side. Validate when the observation carries the evidence needed to reject it; fetch fresh state when it does not.

AI use statement

Large language models appear in this work in two roles. *As objects of study*, they are the agents, user simulators, and judges of §3; that is experimental apparatus, specified there. *As research and authoring assistance*, we disclose the following against the ICLR taxonomy.

Tasks with required disclosure. We used generative AI tools to help develop the conceptual framework (the observability boundary), to propose and refine hypotheses, to formulate the mathematical claim of the no-go theorem and contribute ingredients to its proof, to assist in writing that proof, to

give feedback on experimental design, to implement methods, and to help interpret results. Method implementation was substantial: the LOCO harness, the StructAudit compiler, the ported telecom environment, and the analysis and figure scripts were written with assistance. We also used assistance to convert the telecom domain's source database into the harness format, a reformatting step with no generated content. We did *not* use generative AI to generate synthetic data sets: every corruption is produced by seeded, deterministic perturbation code (§3), and the τ-/τ^2-bench records are the benchmarks' own. Translation and thematic analysis of qualitative data are not applicable to this work.

Tasks with recommended disclosure. We additionally used generative AI tools to draft and edit parts of the manuscript, to improve readability, to suggest structure, to search for and summarise related literature and identify gaps, to format references, to propose the title and keywords, to write and edit software, and to create the figures.

Review and responsibility. We have reviewed all AI-assisted work. Every number in the paper is recomputed by released code from recorded run artifacts, so no reported quantity rests on a model's assertion. The formal statement, lemma, and proof in Appendix A were checked line by line by the authors against the definitions they use. Every citation surfaced by an assisted literature search was verified against the primary source for existence, for faithfulness to the claim we attach to it, and for correct metadata; candidates that could not be verified were discarded rather than cited. No LLM is an author. The authors take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

This is defensive reliability research on synthetic benchmark tool data (τ-/τ^2-bench); no human subjects, no private data. StructAudit is an admission gate that could in principle be used to suppress legitimately changed tool outputs; the 3-valued/abstaining design and the freshness no-go (which routes mutable state to verification rather than rejection) are intended to mitigate exactly that failure mode.

Reproducibility statement

All runs are deterministic (greedy agent + user-sim, temperature 0; seeded entity-keyed corruption), so re-runs reproduce break/rescue sets exactly. We specify exact model revisions (Qwen2.5-14B/32B-Instruct), seeds (s42–s50), task splits and clean-solvable sets (§3), the corruption model and metric definitions (§3–§4), and the StructAudit compilation algorithm (§7). Coverage maps are frozen before recovery runs. Code (LOCO, StructAudit, the domain envs, figure harnesses) and per-result raw outputs will be released with the submission package; the models (Qwen2.5-14B/32B-Instruct, gpt-oss-120B) and benchmarks (τ-/τ^2-bench) are public, though exact serving-stack reproduction is not guaranteed.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Whang, and Martin Zinkevich. Data validation for machine learning. In *Proceedings of Machine Learning and Systems (MLSys)*, 2019.

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. STALE: Can LLM agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*, 2026.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining re-

trieval for RAG systems. In *Proceedings of the 47th International ACM SIGIR Conference (SIGIR)*, 2024.

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

Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McNamara, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 69(1–3):35–45, 2007.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/s41586-024-07421-0.

Great Expectations Labs. Great expectations: Always know what to expect from your data. `https://github.com/great-expectations/great_expectations`, 2024. Open-source data validation framework.

Shayan Kiyani, Sima Noorani, George Pappas, and Hamed Hassani. When to trust the cheap check: Weak and strong verification for reasoning. *arXiv preprint arXiv:2602.17633*, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 12:157–173, 2024a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024b.

Yanming Liu, Xinyue Peng, Jiannan Cao, Xinyi Wang, Songhang Deng, Jintao Chen, Jianwei Yin, and Xuhong Zhang. ToolGate: Contract-grounded and verified tool execution for LLMs. *arXiv preprint arXiv:2601.04688*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Aritra Mazumder and Nusrat Jahan Lia. AgentCheck: A reproduce-intervene-mitigate workbench for LLM agents over MCP. *arXiv preprint arXiv:2607.11098*, 2026.

OpenAI. gpt-oss-120b and gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, et al. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations (ICLR)*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo guardrails: A toolkit for controllable and safe llm applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2023.

Vikas Reddy, Sumanth Reddy Challaram, and Abhishek Basu. Reason less, verify more: Deterministic gates recover a silent policy-violation failure mode in tool-using LLM agents. *arXiv preprint arXiv:2607.07405*, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ilai Shraga, Roei Eshel, and Lior Gorelik. Approved too late: Verdict staleness in LLM-guarded self-adaptive systems. *arXiv preprint arXiv:2608.26306*, 2026.

Joel Strickland, Arjun Vijeta, Chris Moores, Oliwia Bodek, Bogdan Nenchev, Thomas Whitehead, Charles Phillips, Karl Tassenberg, Gareth Conduit, and Ben Pellegrini. Talk freely, execute strictly: Schema-gated agentic AI for flexible and reproducible scientific workflows. *arXiv preprint arXiv:2603.06394*, 2026.

Yang Wan, Xihang Yue, Zhirui Liu, Ziyuan Chu, Shuxun Wang, Yuhan Chen, Xiaonan Jiang, Xukun Zhu, Yubo Dong, and Linchao Zhu. No free checker: A survey of verifiers for robot policies. *arXiv preprint arXiv:2609.09250*, 2026.

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

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. In *International Conference on Learning Representations (ICLR)*, 2024.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025.

Zihao Zheng, Baichuan Li, Junyi Yao, and Jiayu Long. Engineering reliable commit gates for agentic AI: Cost-aware verification portfolios under common-mode data failures. *arXiv preprint arXiv:2609.10969*, 2026.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A THE FRESHNESS NO-GO THEOREM: FULL STATEMENT AND PROOF

This appendix gives the formal setup and complete proofs for the observability boundary stated in §6.

A.1 SETUP

An agent issues tool calls; an entity-lookup tool $T(e)$ for entity e returns an observation o. Let $S_t(e)$ be the true state of e at time t. A correct observation is $o{=}\mathrm{render}(S_t(e))$.

Formal gate. A formal admission gate G is any predicate $G : (q, H, o) \to \{\text{ADMIT}, \text{FLAG}\}$ that is a function only of the query q (the kwargs, including the looked-up id), the in-trajectory history H (prior observations/actions in this run), and the observation o itself, *with no access to any fresh external read of* $S_t(e)$. The theorem below applies to gates built from the non-temporal formal-predicate family F that StructAudit compiles: **schema/type** (o parses and has the declared required fields); **id-echo** (o's id field equals q's looked-up id); **enum** (categorical fields in their compiled value sets); **range** (numeric fields in their compiled [lo, hi]).

Stale-valid corruption. A stale-valid corruption returns $o'{=}\mathrm{render}(S_{t'}(e))$ for some earlier $t'{<}t$ such that o' was a *legitimate* observation of the *same* entity at t' (e.g., an order whose `status` was `pending` yesterday and `delivered` today; the yesterday-record is well-formed, echoes the id, in-enum, in-range).

A.2 LEMMA (STALE-VALID IS IN THE ADMIT-KERNEL OF EVERY FORMAL PREDICATE)

Lemma 2. *For every predicate $p \in F$, $p(q, H, o'){=}$*ADMIT.

Proof. $o'{=}\mathrm{render}(S_{t'}(e))$ is a real past rendering of the *same* entity e. *schema/type:* a valid past record satisfies the (time-invariant) schema, so ADMIT. *id-echo:* o' is entity e's own record, so its id field $= e$'s id $= q$'s id, so ADMIT. *enum/range:* past field values were valid states of e, drawn from the same domains the compiler mined, so they are in-enum / in-range, hence ADMIT. □

This lemma is exactly why StructAudit's compiler, by construction, places mutable fields in the freshness-limited set *before* domain mining (it never emits a value-predicate over a field that can legitimately change), and why `echo_absent` abstention exists: the gate only asserts what is time-invariant.

A.3 THEOREM (NO-GO) AND COROLLARY

Theorem 2. *Let $o{=}\mathrm{render}(S_t(e))$ (fresh) and $o'{=}\mathrm{render}(S_{t'}(e))$ (stale-valid), and suppose $o \neq o'$ (the stale value differs: a real corruption, not a no-op; cf. `content_differs`). Then for every formal gate G built from F, $G(q, H, o'){=}G(q, H, o)$: G cannot distinguish the corrupted observation from the correct one. Consequently, on the stale-valid sub-population any such G has detection power equal to its false-positive rate: a formal gate cannot* detect *stale-validity, so any apparent break-recovery attributable to G on this sub-population arises only from flag-triggered verification (i.e. chance), not from detection.*

Proof. G is a Boolean combination of predicates in F. By the Lemma each $p \in F$ admits o'; each p also admits the fresh, valid o. Hence every p returns ADMIT on both inputs, so any combination G returns the same verdict on o' and o. G's verdict is therefore independent of whether the observation is fresh or stale-valid, so its ROC on this sub-population is the chance line. □

Corollary 2. *To recover a stale-valid break, a policy must invoke a verification oracle V whose output has positive mutual information with $S_t(e)$, i.e. a* fresh *read. Write $V_{p_{fresh}}$ for a verifier that returns the current value w.p. p_{fresh} and a same-cache (stale) value w.p. $1 - p_{fresh}$. Then expected stale-recovery is monotone non-decreasing in p_{fresh} and is 0 at $p_{fresh}{=}0$ (same-cache), because at $p_{fresh}{=}0$ the verifier adds zero mutual information about $S_t(e)$.*

A.4 SCOPE

The theorem is about **stale-valid** corruptions (the value is a real past state of the *same* entity). It does *not* apply to wrong_entity (id-echo catches it, unless the id is not echoed, the separate `echo_absent`/unobservable case), partial (completeness), or malformed (type): those are formally

detectable, which the coverage map predicts and the data confirm. "Freshness-limited" and "unobservable" (echo_absent wrong_entity, e.g. user records) are *distinct* boundary regions, both requiring V with fresh evidence; the theorem covers the freshness-limited region, while the unobservable region is an *observability* gap (no id redundancy), not a staleness gap. This is an *information* boundary, not a complexity one: a more powerful in-trajectory gate does not help, because the limitation is missing information, not missing computation.

B TERMINOLOGY

Terminology. *break*/*rescue*: a clean-pass task that fails / a clean-fail task that passes, under corruption; *qc*: per-entity-read corruption rate; *LOCO*/*LOTO*: leave-one-corruption-out / leave-one-tool-out; *naive_schema*: declared-JSON-schema validator (no id-echo, no inferred range); *StructAudit* (abbrev. *SA*): the cheap formal admission gate.

C STRUCTAUDIT COMPILATION ALGORITHM

Algorithm (StructAudit compilation, per tool, label-free).

1. *Examples* = the domain's clean entity-record collection for the tool (id $\rightarrow$ record); no task outcomes/labels used. The compiled map is *frozen before* any recovery run.
2. *Echo (entity-consistency, 3-valued).* For each lookup-key k (from the tool's declared arg schema + a fixed id lexicon), measure how often the record body echoes k. If echo-support $\geq \tau_{\text{hi}}$ (0.95) emit an `echo:k` predicate (FAIL on mismatch); if $\leq \tau_{\text{lo}}$ (0.05) mark `echo_absent` (abstain $\rightarrow$ UNOBSERVABLE, *not* FAIL; this fixes the id_match false-positive on records like users that don't echo their id); ambiguous $\rightarrow$ no predicate.
3. *Completeness/type.* Fields present in $\geq \tau_{\text{req}}$ (0.95) of examples $\rightarrow$ required; emit a completeness/parse predicate (FAIL on missing/malformed).
4. *Freshness first.* Before any value mining, route mutable fields to the freshness-limited set (name lexicon, e.g. `status`/`*_date`/usage, $\cup$ write-tool name overlap); these are *never* given value predicates and route to FreshVERIFY (never FAIL). Identity/lookup keys are excluded from freshness.
5. *Enum/range* over the remaining non-freshness value fields (requires $\geq N_{\text{MIN_ENUM}}{=}20$ examples; enum if cardinality $\leq \min(12, 0.2n)$, else numeric range $[\text{lo}, \text{hi}]$ with non-negativity for counts). These are the predicates a *declared* JSON schema cannot express, and (with id-echo) are the measured lift over `naive_schema`.
6. *Coverage map* = `predict_coverage(trace)`: wrong_entity$\rightarrow$formal iff an echo predicate exists (else formal_partial if an enum can catch an OOD swap, else unobservable); partial$\rightarrow$formal iff required fields; malformed$\rightarrow$formal; stale$\rightarrow$freshness-limited iff any mutable field. A **zero-shot variant** predicts a held-out tool's map from field names + the fixed priors only (no records, so no enum/range), used for the LOTO / cross-domain transfer claim.

D POSITIONING

approach	label-free	task-causal labels	predicts recoverability	freshness no-go
JSON-schema / declared validation	✓	×	partial	×
invariant mining / Great-Expectations	✓	×	×	×
verification / self-consistency	sometimes	×	×	×
this paper (LOCO + StructAudit + no-go)	✓	✓	✓	✓

Table 3: Positioning. The contribution is an observability boundary for tool-output reliability: causal per-observation labels tied to task outcomes (LOCO), a coverage map that predicts recoverability *before* running the gate, and a no-go theorem separating formal-detectable, unobservable, and freshness-limited regions.

E ADDITIONAL EXPERIMENTS

E.1 CROSS-DOMAIN TRANSFER (TELECOM τ^2-BENCH; ZERO-SHOT HELD-OUT DOMAIN)

We port telecom (plans/devices/lines/customers/bills, structurally distinct from retail) into the same harness. Model-free checks give coverage-map zero-shot agreement $20/20$, and the runtime gate fires correctly on $5/5$ tools. In recovery experiments with Qwen-14B, all $28/28$ telecom tasks are clean-solvable and StructAudit preserves $28/28$ clean tasks at every corruption rate, including qc0.9 (zero clean false positives cross-domain; Table 4).

rate	breaks	id_match	l1(hand)	naive	**StructAudit**	verify_all
qc0.3	2	0	0	2/2	**2/2**	2/2
qc0.5	2	0	0	2/2	**2/2**	2/2
qc0.7	3	0	0	2/3	**2/3**	3/3
qc0.9	5	1	1	2/5	**3/5**	5/5

Table 4: Telecom zero-shot transfer (breaks recovered). (a) Generated StructAudit recovers on a never-seen domain with zero clean false positives at all rates. (b) Generated gates transfer better than retail-authored hand-coded gates (l1/id_match recover $\sim$0). (c) Cross-domain novelty margin appears at qc0.9 (SA$-$naive $+1$). Residual qc0.9 breaks (StructAudit $3/5$ $<$ verify_all $5/5$) are freshness-limited, so the boundary replicates cross-domain and motivates FreshVERIFY.

Multi-seed qc0.9 (n=9: s42–s50). SA$-$naive is positive in all 9 seeds $(+1, +2, +1, +2, +1, +2, +3, +1, +2)$: mean $+1.67$, 95% t-CI $[1.12, 2.21]$ (excludes 0), sign test p=0.0020; pooled StructAudit $32/50$ (64%) vs naive $17/50$ (34%) vs random $\sim$34%; clean-preserved $28/28$ every seed; zero rescues, so net $=$ recovery (unconfounded cross-domain, unlike retail-32B). Mechanism: StructAudit recovers the *union* of id_match $+$ naive detections and is seed-consistent where naive is not. Caveat: at qc0.9 naive$\approx$random (high corruption inflates blind flagging), so the robust cross-domain claim is StructAudit $\gg$ random $+$ the all-positive SA$-$naive; naive is a weak baseline specifically at qc0.9.

Telecom LOCO (causal labels, seed 42). qc50/qc90 breaks are all single-attributable, 0 cascades, no-op control 100% (replicating retail LOCO validity cross-domain), and the structural predictor's recall is 0.0 on telecom, strengthening "structural proxies fail, LOCO necessary" in a second domain. Per-break types at qc90 (stale $\times$3, partial $\times$2, wrong_entity $\times$1) replicate the boundary picture: formal gates recover wrong_entity/partial, the stale tail needs FreshVERIFY. Small-N (single seed), so we use it to ground detectors, not to estimate effect sizes.

E.2 STRONGER MODEL (QWEN-32B; 4 SEEDS)

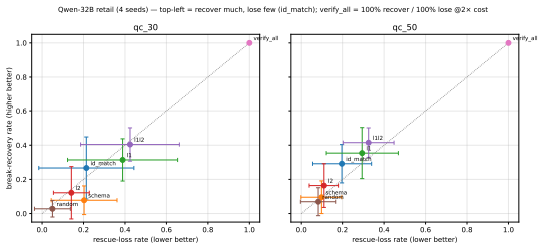

Figure 5: Recovery-rate vs rescue-loss-rate on Qwen-32B, 4 seeds (companion to Figure 3).

The StructAudit beats cost-matched random at both rates, and clean-preserved is $41/41$ for every policy/seed/rate. The boundary is **model-dependent**: the formal region is smaller on 32B, where failures shift toward content/freshness cases. The net-utility seed-confound replicates (l1 net $+3/+3$ s44 $\rightarrow$ $-2/-1$ s45), so the metric framing is cross-model. id_match is the robust low-cost operating point (net-positive in all four cells, $+5$–11% cost).

E.3 SECOND DOMAIN BOUNDARY AND AIRLINE LOCO

Airline reservations are freshness-dominated: formal gates are net-negative (qc_30 $1/3$ net -1), verify_all net $+3$, the regime the coverage map predicts (directional, 7 solvable). Airline LOCO (seed 42, entity-keyed): qc30 breaks $\{34, 37, 43\}$ (2 single, 1 cascade); qc50 breaks $\{34, 37, 43, 49, 24\}$ (4 single, 1 cascade); no-op control 100% both rates. Here the structural read→write predictor has **recall** 1.0 ($P{=}R{=}1.0$ qc30; $P{=}0.667/R{=}1.0$ qc50), the *opposite* of telecom (recall 0.0) and retail (0.13–0.17): airline's freshness-dominated, reservation-keyed breaks are structurally predictable. This domain contrast sharpens when structure suffices and when LOCO is needed.