Self-Healing Harness for Runtime Oversight of Agent Self-Modification

Sina Tayebati[1], **Divake Kumar**[1], **Nastaran Darabi**[1], **Ranganath Krishnan**[2], **Amit Ranjan Trivedi**[1]

[1] University of Illinois Chicago
[2] AI Labs, Capital One

arXiv:2609.24130v1 [cs.AI] 21 Sep 2026

Abstract

LLM agents can change their own future behavior, raising a basic control question of which self-generated changes should be allowed to persist. We formulate this as *admission control for self-modification*. The agent may propose changes to its operating instructions, while an external runtime gate controls persistence. We implement this principle as a model-agnostic self-healing harness that runs a **Detect**, **Notice**, **Heal**, **Validate** loop around an otherwise unmodified agent. The agent authors candidate behavioral rules in an external workspace, where they receive *provisional* execution authority during evaluation and acquire *persistent* cross-episode authority only after measured improvement on the triggering failure without regression beyond a fixed margin on protected cases. Replay provides matched evidence when available, forward trials provide a weaker fallback, and a corpus-level guard re-tests the accumulated active rule set. Across 16 matched Baseline and Harness runs spanning AppWorld, Terminal-Bench, and τ^2-bench, the gate rejected 383 replay-decided proposals. Of these, 211 (55%) improved their triggering failure while degrading a case that previously worked. This shows that locally beneficial self-modifications can introduce collateral regressions often enough to materially affect gate decisions, providing direct empirical motivation for external admission control. Task-completion score is higher under the Harness in all 16 pairs, with two paired bootstrap intervals excluding zero, while repeated-trial reliability is higher in 12 pairs, tied in 4, and lower in none. Because adaptation modifies the policy-inducing context while leaving model weights fixed, admitted changes remain inspectable, reversible, and compatible with closed-weight models.

Introduction

Autonomous LLM agents increasingly operate over long horizons, invoke external tools, modify environment state, and retain information across episodes. Persistent deployment therefore lets an agent alter its own future behavior through stored memories, revised operating instructions, or reusable procedures. This introduces the question of which self-generated behavioral updates should be allowed to persist. Improvement on the failure that motivated an update is insufficient by itself, since the same update may degrade behavior that previously succeeded.

We address this by separating update generation from update retention. The agent observes failures and proposes behavioral rules intended to correct them, while an external runtime decides whether each rule becomes persistent, retaining a candidate only when measured execution evidence shows improvement on the motivating failure without regression beyond a fixed margin on protected behavior. This places the persistence decision outside the adapting agent while preserving the agent's ability to generate its own corrections. The mechanism provides external oversight of which self-generated modifications are retained; we do not aim for broad value alignment.

The need for that oversight is visible directly in the agent's own proposals. Across our experiments 383 candidate rules were rejected by replay-based validation, and 211 of them (55%) improved the failure that motivated the rule while degrading a protected case that had previously succeeded. A retention policy keyed only to improvement of the triggering failure would have accepted every one of them.

We implement this as *evidence-gated self-modification* inside a self-healing harness (the Harness), a model-agnostic runtime layer around an otherwise unmodified agent that executes a **Detect**, **Notice**, **Heal**, **Validate** loop: degradation is detected from per-turn score trajectories, the evidence is recorded in an external workspace, the agent inspects it and authors a candidate rule, and validation decides retention by replaying captured failure and protected cases, or by using subsequent executions as weaker forward evidence when replay is unavailable. A corpus-level guard additionally re-tests the accumulated active set, since interactions among individually accepted updates can produce regressions no candidate-level test can see. Because adaptation modifies the policy-inducing context rather than model parameters, retained rules stay explicit, inspectable, reversible, and compatible with closed-weight models.

Deciding *when* to intervene is part of the problem, since an isolated low intermediate score may reflect incomplete progress rather than failure on a long-horizon task. The Harness therefore reads the temporal evolution of per-trace scores rather than individual values, with a per-turn barrier supplying the repeated observations that requires. We measure the operational effect through task completion and execution reliability, distinguishing first-trial success ($\mathrm{pass@1}$) from all-trials success (pass^k, the fraction of tasks completed on all k trials), because stochastic agents may succeed on one execution and fail on another. The alignment-relevant claim is narrower. A self-generated update does not become persis-

tent merely because the agent produced it. Our contributions are: (i) we formulate persistent agent adaptation as **admission control for self-modification**, where a self-authored update is retained only after demonstrating improvement on its triggering failure while satisfying a multi-metric non-regression constraint on protected behavior; (ii) we develop a model- and framework-independent **runtime architecture** integrating trajectory-based detection, agent-authored repair, an external rule workspace, replay or forward validation, and corpus-level regression testing, without modifying weights; (iii) we provide **empirical evidence of collateral regression**, since 211 of 383 replay-based rejections improved their triggering failure while degrading protected behavior, so local improvement does not imply reliable persistent adoption; and (iv) we report a **paired evaluation** over three benchmark families and four models covering task completion, repeated-execution reliability, rule validation, accumulated-rule effects, evaluator robustness, and overhead, including failure modes that persistent adaptation itself introduces.

Background and Related Work

Self-correction, memory, and verification. Reflection and self-refinement let agents revise behavior after failure (Madaan et al. 2023; Shinn et al. 2023; Gou et al. 2024), and memory and experiential-learning systems carry adaptation across episodes through long-term memory (Packer et al. 2023; Park et al. 2023) or reusable skills and lessons (Wang et al. 2023a; Zhao et al. 2024). These determine what experience to generate, retain, or retrieve. Verification methods assess individual outputs (Wang et al. 2023b; Lightman et al. 2024; Zheng et al. 2023), while guardrails and principle-based steering constrain execution under externally specified policies (Tayebati et al. 2025b,a; Rebedea et al. 2023; Darabi et al. 2026). We instead ask whether an agent-authored behavioral update should persist, and condition retention on measured evidence, so the object verified is a modification to the operating context rather than the current answer.

Persistent adaptation and oversight. Cross-episode adaptation can accumulate individually useful updates that interact and degrade previously successful behavior, analogous to catastrophic interference in continual learning (McCloskey and Cohen 1989); adapting explicit context rather than parameters keeps each update inspectable and reversible, and a corpus-level guard addresses failures emerging from the accumulated set. Test-time adaptation typically updates parameters from deployment data (Sun et al. 2020), whereas we hold π fixed and adapt an external rule set, preserving compatibility with API-served models. The mechanism relates to scalable oversight and corrigibility (Amodei et al. 2016; Bowman et al. 2022; Tayebati et al. 2026; Kumar et al. 2025), though the gate enforces non-regression only over observed metrics, which here measure task reliability rather than value alignment. Because completion varies across stochastic executions we distinguish $\mathrm{pass@1}$ from pass^k, evaluating across digital-work, terminal, and tool-calling environments (Trivedi et al. 2024; Merrill et al. 2026; Barres et al. 2025; Yao et al. 2024).

Self-modifying agents and novelty boundary. AutoManual builds and revises an environment rule book through interaction (Chen et al. 2024), Agent Workflow Memory induces reusable workflows from past trajectories (Wang et al. 2024), and A-MEM maintains an agent-curated memory store (Xu et al. 2025). Closer still, STOP retains scaffold modifications on measured meta-utility (Zelikman et al. 2024) and the Darwin Gödel Machine maintains benchmark-evaluated self-modified coding agents (Zhang et al. 2025), which establishes that empirical validation of self-modification is not by itself new. What the Harness combines is that the agent authors a *persistent behavioral update* while an external runtime controls retention; admission uses replay or subsequent execution and requires target improvement *together with* protected-case non-regression; updates follow a candidate, active, retired lifecycle with the accumulated set re-tested; and weights stay fixed. Relative to STOP and the Darwin Gödel Machine the distinctions are the protected-case constraint and runtime triggering inside an ongoing deployment rather than an offline improvement search. Table 7 compares eleven representative systems on these properties.

The Self-Healing Harness

Overview and Control Model. The Harness is a runtime envelope around an unmodified agent that lets the agent revise its own operating instructions while enforcing an evidence gate, so only validated revisions become durable: π stays constant while the rule set $\mathcal{R}$ evolves, making adaptation model-agnostic, auditable, and reversible. Four stages define the control boundary (Figure 1) and give this section its structure. Detect, Notice, and Heal sit on the proposal side of that boundary; only Validate grants persistence, and a per-turn *barrier* closes the loop inside the session in progress.

The mechanism's guarantees are interpretable only against what it governs. We assume an agent that can author natural-language rules, read and write its workspace, retrieve rules into context, and act on any rule in force, but that cannot edit the evaluator, the gate, the replay corpus, or the promotion record, all outside its tool surface. The runtime controls whether a candidate becomes persistent and whether the accumulated active set continues to satisfy non-regression on previously evaluated cases. Each promotion records the case, metric, and improvement that justified it, so every persistent change carries an auditable reason. The agent is assumed non-adversarial: we do not defend against one that circumvents the workspace, persists behavior through another channel, manipulates the evaluator, or authors a change whose harm is invisible to the metric vector, and regressions on behavior the corpus never exercised are undetectable.

Detect: A Trace-Level Trigger. An agent policy π acts over a session $s = (x_1, \ldots, x_T)$, where each x_i is a per-turn trace. We detect degradation from score trajectories rather than session-level aggregates, since a low intermediate score in a long-horizon task can reflect incomplete progress rather than failure. An evaluation service $\mathcal{E}$ scores each trace on $[0, 1]$. Metrics are tiered by cost: inexpensive progress metrics $\mathcal{M}_1$ (task completion, coherence) run continuously, while step-level $\mathcal{M}_2$ (tool-call and argument correctness), explanatory $\mathcal{M}_3$, and outcome $\mathcal{M}_0$ are evaluated only after a trajectory

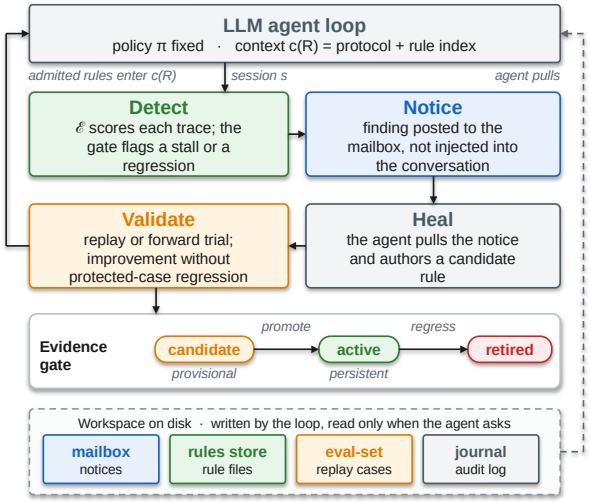

Figure 1: **Harness control boundary.** After each turn, trace-level evaluation detects a stall or regression and escalates when needed (**Detect**); the finding is written to the external workspace (**Notice**); the agent retrieves it and authors a candidate rule (**Heal**); and the candidate remains provisional until replay or forward-trial evidence supports promotion without regression (**Validate**). A per-turn barrier closes the loop before the next turn, while the pull model keeps workspace contents outside the conversation unless explicitly retrieved.

trigger (the Appendix).

Trajectory gate. Tier-1 metrics are judged by the shape of their per-trace series rather than by an absolute score. For metric j the gate $\mathcal{G}$ keeps a running peak pk_j and a counter z_j of traces since the last meaningful gain; a gain occurs when the score exceeds the peak by at least γ_{gain}, and otherwise the counter increases and the gate evaluates

$$\mathrm{stall}_j(x_i) = \mathbf{1}\left[\, z_j \geq w \;\wedge\; \mathrm{pk}_j < \theta \,\right], \tag{1}$$

$$\mathrm{regr}_j(x_i) = \mathbf{1}\left[\, m_j(x_i) < \mathrm{pk}_j - \delta_{\mathrm{drop}} \,\right]. \tag{2}$$

A gain resets z_j and updates the running peak. Consequently, improving trajectories do not trigger merely because they begin at a low score, plateaus above θ are permitted, and regressions are measured against the best observed state. After a trigger the gate opens a new window. We use w=10, θ=0.5, γ_{gain}=0.02, and δ_{drop}=0.15 (Algorithm 3, Table 6).

Escalation and evidence. Tier 1 runs on every new trace, and if nothing stalls or regresses evaluation stops there. Otherwise tier 2 scores the latest trace and asks whether the trajectory signal is corroborated by a local execution failure, meaning a step-level or outcome metric below its absolute threshold. Only a corroborated finding, a `breach`, is recorded as a replayable case for validation; an uncorroborated one is surfaced to the agent as a `trend` but never becomes promotable evidence. Corroboration is therefore required before a signal becomes evidence for a persistent change, and RQ5 reports its effect when the evaluator itself is biased.

Outcome evidence. Tier-1 and tier-2 metrics are judged proxies, so when a deployment provides a direct success signal the Harness admits it as an optional *outcome verifier* that returns a score or abstains, and validation prefers this evidence over any judged proxy. Abstention is explicit, since a verifier that scores only completed tasks cannot report mid-session.

Per-Turn Barrier. After each turn, execution waits until new traces are evaluated and any resulting notice is posted. This allows a repair authored at turn i to affect turn $i{+}1$, and supplies the repeated within-session observations the trajectory gate requires. Synchronizing only once per task would give a single observation per metric series, leaving the stall detector inoperative. Per-turn synchronization is therefore a precondition of the trigger rather than an optimization.

The External Rule Workspace. Learned rules must stay available without forcing the whole corpus into context every turn, since full inlining raises both token cost and competition for attention. The Harness therefore separates the *index* of learned rules from their *content*:

$$c(\mathcal{R}) = c_0 \oplus \iota(\mathcal{R}), \tag{3}$$

where c_0 is a fixed root document holding the standing protocol and tool interface and $\iota(\mathcal{R})$ is a compact directory of rule files, their sizes, and pending notices. No rule bodies appear in this context: rule text enters context only when the agent reads a scope or searches the workspace, so the standing context grows with the *number* of rule files rather than the volume of what has been learned, though retrieved text does occupy the conversation once pulled, which is the mechanism behind the token growth in RQ4. This is a *pull model*: notices, rules, traces, replay cases, and the audit journal live in a workspace $\mathcal{W}$, and none of it is injected into the conversation. The agent reaches them through explicit tool calls, so workspace contents remain external until retrieved.

Heal: Agent-Authored Behavioral Updates. Repair is authored by the agent and retention is controlled by the runtime. After retrieving a notice and inspecting the flagged trace, the agent writes a candidate behavioral rule r, which becomes persistent only if it passes the evidence test below. The agent therefore determines what to propose while the runtime determines what persists. The Appendix states the update operator and rule metadata; Algorithm 1 gives the per-turn procedure.

Validate: Evidence-Gated Self-Modification. The gate controls cross-episode persistence. A candidate remains provisional and may influence behavior while its effect is measured, since a candidate under trial must be in force for the trial to measure anything, but persists across episodes only after promotion. Promotion records the supporting evidence. Validation uses replay when suitable prior cases are available and a weaker forward trial otherwise (Algorithm 2, Appendix).

Replay scoring and case selection. Replay re-scores the captured session on $\mathcal{M}_1 \cup \mathcal{M}_2$, the same metrics that can produce a triggering notice, so the gate tests a candidate against the signal that motivated the repair rather than a broad aggregate; $\mathcal{M}_3$ is excluded as explanatory. The verdict is read against the most authoritative target available per case, a direct verifier outcome if there is one, else the metric the rule declares, else the metric named by the triggering signature, so abstention never blocks validation. Each round replays at

most three *failure* cases, matched to the candidate's signature $\mathrm{sig}(r)$, and at most two *protected* cases.

A protected case is a captured session paired with the metrics on which it scored above threshold. Because capture is triggered at the session level while scores are recorded per metric, a session that entered the corpus on one failing metric still contributes its passing metrics as protected behavior during replay. The non-regression condition therefore rejects a candidate that repairs its triggering metric while degrading another that previously succeeded, which makes the test a cross-case non-regression test rather than a target-only improvement test. At most two protected cases are replayed per round and we do not rank them for semantic proximity to the candidate, which is the conservative reading of the mechanism: protection is necessarily incomplete and side effects outside the sampled cases can escape detection. Protected-set construction is therefore a first-order determinant of what the gate can see, and *Limitations* lists relevance-ranked selection as an important untested variant.

The admission test. Each selected session is replayed under the candidate context $c(\mathcal{R} \cup \{r\}, \cdot)$ and re-scored. Writing $\Delta_j(s_i)$ for the replay-minus-original difference on metric j of case s_i, a candidate must improve its target metric $\hat{\jmath}$ on at least one failure case, $\exists i : \Delta_{\hat{\jmath}}(s_i) \geq \delta_{\mathrm{prom}}$, while satisfying non-regression across all replayed cases and all measured metrics, $\forall i, \forall j : \Delta_j(s_i) > -\delta_{\mathrm{reg}}$, with $\delta_{\mathrm{prom}} = \delta_{\mathrm{reg}} = 0.05$. These two predicates are the admission rule. The admission criterion gives precedence to non-regression: any non-regression violation therefore causes rejection irrespective of target improvement. Inconclusive replay falls back to the forward trial after at most three attempts. The test is a bounded empirical procedure, not a certificate of causation, since agents are stochastic and at most five sessions are replayed, so a delta near the margin can reflect sampling variation and repeated-replay agreement of verdicts is unmeasured.

Forward trial, and why it is weaker. When replay is unavailable the gate instead measures whether the candidate reduces future occurrences of its triggering failure, comparing the flagged rate over the n sessions following activation against the pre-activation rate p_0 and promoting only if the failure stops or its rate falls by at least δ_{prom} (the Appendix). The study uses $n = 3$ so a trial completes inside one run, with $p_0 = 1$ when no history exists. This path is materially weaker: three sessions are unmatched evidence, there is no protected-case component so collateral damage cannot be detected at all, and the default $p_0 = 1$ makes promotion comparatively easy. RQ3 reports how the two paths were distributed across runs.

Regression Guard for Behavioral Drift. Candidate-level validation tests individual updates against sampled evidence, while regressions can emerge as the active set accumulates. The guard therefore replays the whole evaluation corpus under the current active rules, re-scores each case on $\mathcal{M}_1 \cup \mathcal{M}_2$, and fails the corpus if any previously evaluated case drops by at least δ_{reg} on any measured metric, improvement elsewhere not offsetting a regression. Within the coverage of that corpus the guard bounds degradation of previously evaluated behavior at δ_{reg} per metric. Three limits bound the guarantee. It covers only properties in the metric vector, which here measure task performance, so it establishes non-regression in measured reliability rather than value alignment, although safety or permission signals routed through the verifier would fall under the same guard. Replay detects only regressions the corpus exercises. And it inherits the reliability of $\mathcal{E}$, including the evaluator noise of RQ5.

Experimental Setup

Protocol. We use a paired A/B protocol in which **Baseline**, the unmodified agent loop, and **Harness**, the same loop with Detect, Notice, Heal, and Validate enabled, share the agent implementation, model, prompts, task set, and execution order and differ only in self-healing. **PandaProbe** (Tayebati 2026) supplies the tracing and evaluation infrastructure with identical metric definitions, gate parameters, and thresholds in both arms. Benchmark-native graders determine $\mathrm{pass@1}$ and pass^k and supply the continuous signal behind $\bar{s}$, while trace-level evaluations drive Harness adaptation, keeping the evaluator separate from every reported outcome measure.

Benchmarks and models. We evaluate **AppWorld** (Trivedi et al. 2024) for long-horizon application and API tasks, **Terminal-Bench** (Merrill et al. 2026) for command-line tasks, and **τ^2-bench** (Barres et al. 2025) for tool-using dialogue. The four splits are AppWorld `test_normal` (168 tasks), Terminal-Bench `sample@2.0` (10), and τ^2 `airline` (50) and `retail` (114), each evaluated with GPT-5.6-TERRA, GPT-5.6-LUNA, CLAUDE-HAIKU-4.5, and CLAUDE-SONNET-5 using k=4 trials, a 100-turn budget per task, and seed 1, yielding 16 matched pairs (Table 8, Appendix). Trial-to-trial stochasticity is retained because the evaluated endpoints do not expose temperature control.

Online adaptation. The Harness remains active throughout each run, allowing earlier tasks to affect later behavior through proposed, validated, promoted, and retired rules, so matched pairs use the same execution order and all outstanding validation jobs are settled before archival. Each split is processed once as a continuous adaptation stream without a separate learning and evaluation partition.

Metrics and statistics. Each task uses the benchmark's native binary success criterion, with $\mathrm{pass@1}$ denoting first-trial success and pass^k success on all four trials. For $\bar{s}$, we compute paired per-task Harness-minus-Baseline differences and report their mean with a 95% percentile-bootstrap interval over 10,000 task-level resamples and a two-sided bootstrap p-value, resampling tasks rather than trials because repeated trials of one task are dependent (Efron 1979). Intervals containing zero are treated as directional, and formal metric and interval definitions appear in the Appendix.

Results

Scope and Organization. We report 16 matched Baseline–Harness pairs across three benchmarks, four splits, and four models, with k=4 trials per task and seed 1. Each comparison uses only task-trials completed by both arms under the same benchmark, dataset, model, seed, and k configuration. We organize the results around five questions. **RQ1**:

does gated persistent adaptation improve reliability? **RQ2**: do self-generated updates require admission control? **RQ3**: does the gate discriminate among proposals, and how does the validation path affect outcomes? **RQ4**: does accumulated adaptation introduce additional risk? **RQ5**: what are the robustness and overhead of continuous runtime oversight?

RQ1: Does gated adaptation improve reliability? We report a *task-completion score* $\bar{s}$ as the primary continuous metric: each benchmark's own continuous signal placed on one $[0, 1]$ scale, namely AppWorld's and Terminal-Bench's passing-test fraction and τ^2's per-component reward, averaged over all trials. It is the highest-resolution quantity these benchmarks expose. pass@1 and pass^k are reported alongside it. Table 1 gives the primary comparison. The clearest single result is AppWorld with GPT-TERRA, the study's largest paired evaluation at 166 tasks, where the Harness raises $\bar{s}$ from 0.701 to 0.723 ($\Delta = +0.022$) with a bootstrap 95% CI of $[+0.006, +0.040]$ that excludes zero ($p = 0.010$), and raises pass@1 from 0.018 to 0.054. The second interval that excludes zero is τ^2 retail with CLAUDE-SONNET-5, $\Delta\bar{s} = +0.032$, CI $[+0.003, +0.061]$, $p = 0.03$.

Across all 16 pairs the Harness improves $\bar{s}$, leads on pass^k in 12 and ties in 4, and leads on pass@1 in 14 with one tie and one Baseline lead. Only two $\bar{s}$ intervals exclude zero, so the remaining effects are directional. The τ^2 family shows the smallest gains, between $+0.012$ and $+0.032$, and RQ3 identifies a validator limitation specific to part of that family.

Repeatability. Figure 2 illustrates the per-task pattern on Terminal-Bench, where several intermittent Baseline successes become repeatable under the Harness. Because Terminal-Bench contains only 10 tasks, its pass^k values move in coarse steps. The gains do not generally concentrate in pass^k relative to pass@1: only 3 of 16 pairs satisfy $\Delta\text{pass}^k > \Delta\text{pass@1}$ (Table 9, Appendix), so we claim consistent improvement in repeated-trial success rather than a repeatability-specific effect.

RQ2: Do self-generated updates need admission control? Across the 16 Harness runs the agent proposed 2,306 rules, of which 1,244 were promoted and 1,062 retired (Table 10, Appendix), so roughly 46% of proposals were rejected. The more informative statistic is *why*.

Conflict prevention. Among 383 replay-decided retirements, 211 (55%) improved their triggering failure but regressed protected behavior (Table 2). Observed regressions include a 0.60 decrease in *task_completion* and a 0.25 decrease in *outcome_correct*. Replay-decided verdicts are a subset of the totals above, the remainder having been resolved by forward trial, and Terminal-Bench contributes none because replay is unavailable there (RQ3). These proposals were rejected because the non-regression condition overrode measured local improvement, so the majority of rejected self-modifications here were *locally correct and harmful to protected behavior*. Under the observed replay evidence, a retention rule based only on triggering-case improvement would have admitted all 211. This measurement does not depend on the end-task comparison, since it characterizes the agent's proposals under a fixed evidence test rather than the reliability difference in RQ1. The 211 cases are also detected

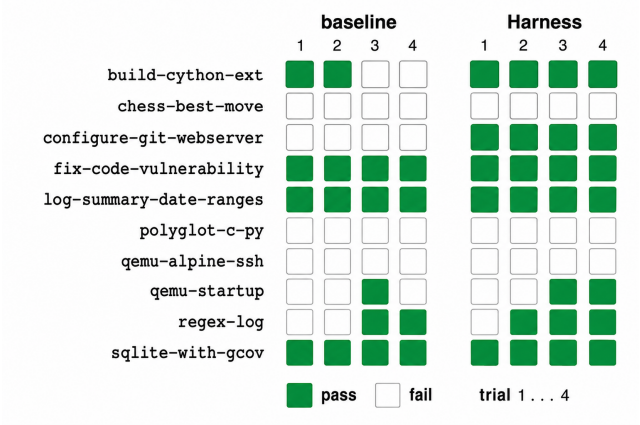

Figure 2: **Per-task trial outcomes** on Terminal-Bench with TERRA; filled cells denote successful trials. Under the Harness, `build-cython-ext` and `regex-log` become consistently successful after intermittent Baseline performance, while `configure-git-webserver` improves from zero to four successes. Overall, four tasks succeed on all four trials under the Harness versus three under Baseline.

conflicts under sparse protection, not an estimate of total collateral-regression incidence. Each validation round examines at most two unranked protected cases, so harmful effects outside that small sample remain unobserved. The result therefore establishes that collateral regression occurs frequently enough to be detected under limited coverage, while providing no guarantee that promoted rules are free of unseen regressions.

RQ3: Does the gate discriminate, and does the evidence path matter? *Selectivity.* Promotion rates vary substantially across benchmarks and validation paths, from a minority on AppWorld to 0.41–0.63 on τ^2, with the per-run rates in Table 10. Replay-decided promotions span all five evaluated metrics rather than concentrating on a single signal (Table 2).

Natural variation in validation path. Replay availability differs structurally across benchmarks (Table 3), so these differences are observational rather than an ablation. Terminal-Bench cannot replay because validating one candidate would require rebuilding and rerunning the task environment, and its four runs correspondingly show zero retirements. The τ^2 airline runs instead suffered a replay-budget unit mismatch, the budget specified in agent turns while τ^2 consumes it as message hops, which truncated many replays before the relevant behavior was exercised.

Both statistically resolved improvements occur among runs with high replay share. Replay availability is confounded by benchmark and model, however, so this association does not establish a performance advantage for replay. It does establish that part of the τ^2 corpus received weaker validation than intended and that Terminal-Bench received no replay-based regression testing at all. Replay share therefore also indicates how much of a run's validation activity received the protected-case non-regression test, which the forward path lacks.

RQ4: Does accumulated adaptation create new risk?

Table 1: Headline results for matched Baseline (B) and Harness (H) runs. $\bar{s}$ denotes task-completion score. CI_{95} and p are the paired task-level bootstrap interval and two-sided p-value for H−B in $\bar{s}$. Bold H entries indicate numerical Harness leads, while bold CI_{95}/p entries indicate intervals excluding zero. All remaining differences are directional. Rates are computed over the task-trials present in both arms, so the denominator behind pass@1 and pass^k can differ within a row.

			Task-completion score $\bar{s}$			pass@1		pass^k	
Benchmark	Model	n	B	H	CI_{95} / p	B	H	B	H
AppWorld	gpt-terra	166	0.701	**0.723**	**[+0.006,+0.040] / .01**	0.018	**0.054**	0.000	0.000
	gpt-luna	168	0.658	**0.662**	[-0.01,+0.02] / .54	0.000	**0.006**	0.000	0.000
	claude-haiku-4.5	168	0.566	**0.590**	[-0.06,+0.10] / .58	0.000	**0.143**	0.000	**0.143**
	claude-sonnet-5	166	0.714	**0.727**	[-0.005,+0.032] / .17	0.030	**0.048**	0.006	**0.012**
Terminal-Bench	gpt-terra	10	0.450	**0.525**	[-0.13,+0.33] / .65	0.444	**0.556**	0.333	**0.444**
	gpt-luna	10	0.275	**0.300**	[-0.10,+0.15] / .72	0.300	**0.400**	0.100	**0.200**
	claude-haiku-4.5	10	0.229	**0.375**	[-0.28,+0.52] / .42	0.600	0.600	0.000	**0.200**
	claude-sonnet-5	10	0.398	**0.417**	[-0.41,+0.43] / .90	0.571	0.429	0.000	**0.333**
τ^2-bench	gpt-terra (retail)	114	0.681	**0.704**	[-0.004,+0.050] / .09	0.384	**0.411**	0.125	**0.143**
	gpt-terra (airline)	50	0.724	**0.741**	[-0.019,+0.054] / .34	0.500	**0.540**	0.360	**0.400**
	gpt-luna (retail)	114	0.650	**0.662**	[-0.02,+0.04] / .47	0.339	**0.357**	0.099	**0.117**
	gpt-luna (airline)	50	0.690	**0.704**	[-0.02,+0.05] / .42	0.460	**0.480**	0.280	**0.320**
	claude-haiku-4.5 (retail)	114	0.632	**0.651**	[-0.009,+0.047] / .18	0.304	**0.321**	0.071	**0.080**
	claude-haiku-4.5 (airline)	50	0.682	**0.696**	[-0.02,+0.05] / .44	0.440	**0.460**	0.320	**0.340**
	claude-sonnet-5 (retail)	114	0.669	**0.701**	**[+0.003,+0.061] / .03**	0.366	**0.384**	0.107	0.107
	claude-sonnet-5 (airline)	50	0.710	**0.740**	[-0.006,+0.066] / .10	0.440	**0.480**	0.320	0.320

Table 2: Replay-based gate decisions, pooled over the runs in which replay resolved the verdict; Terminal-Bench contributes none, since replay is unavailable there. Promotions are grouped by the metric that improved beyond the admission margin. Among retirements, 55% improved the triggering failure but regressed protected behavior.

Decision basis	Count	Share	Verdict
`task_completion`	200	48%	
`tool_correctness`	73	17%	
`outcome_correct`	73	17%	**Promote** (420)
`coherence`	58	14%	
`argument_correctness`	15	4%	
Protected-case regression	**211**	**55%**	**Retire** (383)
No triggering-case improvement	172	45%	

Candidate-level admission tests one update against the available evidence; it does not establish whether the accumulated adaptation state remains reliable, and our data show the second question is not implied by the first. Table 4 tracks per-task $\Delta\bar{s}$ across execution quartiles alongside active-set size, each Harness task differenced against its Baseline counterpart so difficulty is controlled and execution order is not read as learning. It covers the configurations for which quartile-resolved telemetry is available. Those five rows carry three positive and two negative slopes, so the subset is not assembled around the negative result reported below. For most configurations the advantage is present early rather than emerging after accumulation: on AppWorld with GPT-TERRA it stays positive in all four quartiles, and both τ^2 datasets show positive headline effects and positive slopes as the corpus grows. AppWorld with GPT-LUNA is the informative exception, declining monotonically from $+0.017$ in Q1 to -0.016 in Q4 ($p = 0.022$) while the active set grows from 13 to 70 rules and input tokens per turn rise from $1.50\times$ to $2.52\times$ Baseline. Every rule passed the gate individually, so the harm is a property of the corpus rather than of any member.

Two mechanisms are consistent with this pattern: context dilution, as an expanding corpus competes for attention, and rule interference, when individually valid rules give conflicting guidance; the present data do not distinguish them.

Why the corpus guard did not prevent this. The guard exists for exactly this class of effect, but three scope limits mean it could not have registered this particular decline. *Coverage*: the guard replays only captured `breach` cases, whereas the decline is measured over the full task stream. *Granularity*: the guard detects per-case drops of at least $\delta_{\mathrm{reg}} = 0.05$, while the observed Q4 deficit is a smaller aggregate shift distributed across tasks. *Mechanism*: if degradation arises from retrieval dilution as the corpus grows, replaying a stored case may not reproduce the live retrieval state that caused it. The guard therefore bounds regression on replayed cases but does not control aggregate drift over uncaptured behavior. Addressing this failure mode requires controls over the accumulated adaptation state in addition to per-candidate admission. Candidate admission and corpus-level oversight therefore address different failure scales, the first at the level of individual updates and the second at the level of their accumulated interactions.

RQ5: How robust and how expensive is oversight? *Mechanism verification.* Median metric-series length ranges from 6 to 11 across runs, confirming repeated within-session ob-

Table 3: Validation path by run. Replay % is the share of validation activity resolved by replay rather than forward trial, and R/F reports the corresponding replay and forward event counts. These are per validation event and do not correspond one-to-one with the rule counts of Table 10. Terminal-Bench does not support replay and therefore has structural 0% rows. Variation in replay availability is observational rather than a controlled ablation. Bold indicates replay shares above 80%.

Benchmark	Model	Replay %	R/F	$\Delta\bar{s}$	Δpass^k
AppWorld	terra	**87%**	104/15	+0.022	+0.000
	luna	**90%**	189/21	+0.004	+0.000
	haiku-4.5	**84%**	168/32	+0.024	+0.143
	sonnet-5	**86%**	93/15	+0.013	+0.006
Terminal-Bench	terra	0%	0/206	+0.075	+0.111
	luna	0%	0/324	+0.025	+0.100
	haiku-4.5	0%	0/280	+0.146	+0.200
	sonnet-5	0%	0/230	+0.019	+0.333
τ^2 **airline**	terra	72%	118/46	+0.017	+0.040
	luna	69%	151/68	+0.014	+0.040
	haiku-4.5	64%	132/74	+0.014	+0.020
	sonnet-5	58%	59/43	+0.030	+0.000
τ^2 **retail**	terra	**89%**	246/30	+0.023	+0.018
	luna	**92%**	357/29	+0.012	+0.018
	haiku-4.5	**88%**	332/45	+0.019	+0.009
	sonnet-5	**91%**	211/21	+0.032	+0.000

Table 4: Learning dynamics for the five configurations with quartile-resolved telemetry. Q1–Q4 report paired $\Delta\bar{s}$ over execution quartiles; Rules gives the mean active-rule count in Q1 and Q4. Slope is per task in units of 10^{-3}, with permutation p-values from $5{,}000$ shuffles.

Benchmark	Model	Q1	Q2	Q3	Q4	Rules	slope (p)
AppWorld	terra	+.031	+.001	+.040	+.018	10→47	+.03 (.86)
	luna	+.017	+.009	+.006	-.016	13→70	-.27 (.02)
Term-Bench	terra	+.250	+.500	-.125	-.125	7→7	-74.2 (.09)
τ^2 airline	luna	-.010	-.010	+.063	+.012	11→81	+1.23 (.26)
τ^2 retail	luna	-.003	+.029	-.025	+.047	36→195	+.20 (.67)

servations, while tier-2 escalation rates range from 0.29 to 0.83. These are operational checks rather than ablations; full telemetry appears in Table 11 in the Appendix.

Evaluator noise. During AppWorld development, trace judges incorrectly penalized read-only Harness meta-tools, with 14% of sampled rationales blaming meta-tool calls and 31 spurious zero scores. Neutrality guidance reduced these to 0.17% and 1 over $4{,}800$ sampled rationales (Table 12, Appendix). Both arms of every reported pair are scored by the same judge prompts, so an evaluator revision cannot differentially favor either arm of a comparison. Because detection and admission are separate, such trigger noise does not directly grant persistence.

Cost and overhead. Mean context usage is $1.25\times$ Baseline and agent turns increase by 0.0 to 1.8 per trial. Task-execution spend rises from $\$177.06$ to $\$211.46$, with $\$92.90$ additional repair cost, yielding $1.72\times$ Baseline overall. The dominant overhead is latency, with wall-clock time increasing by roughly 1.7 to $7\times$, primarily from synchronous trace evaluation, which is a material deployment constraint for latency-sensitive applications. Per-run accounting appears in Tables 13 and 14 in the Appendix.

Limitations. The primary missing comparison is between evidence-gated and immediate admission of the same agent-authored rules (arm B′), with the downstream corpus guard fixed. The Baseline–Harness comparison therefore does not isolate the effect of pre-admission validation. This limits claims about the benefit of gating, but not the RQ2 observation that protected-case regressions occur directly among the agent's proposals under the fixed replay test. The no-barrier, inlined-rule, tier-3, replay-only, and forward-trial-only arms were also not run, and the protocol measures online adaptation rather than held-out transfer. Only two of 16 paired intervals exclude zero, the remaining effects are directional, and all runs use seed 1 although adaptation is path dependent. The gate is a bounded empirical test over at most five replayed sessions, with repeated-replay stability and sensitivity to δ_{prom}, δ_{reg}, and w unmeasured. Forward trials use only three unmatched sessions and no protected-case test, leaving Terminal-Bench and part of τ^2 airline supported by weaker evidence. Non-regression is further limited by protected-set coverage, since at most two unranked protected cases are replayed per round, so the 211 conflicts in RQ2 are detected conflicts rather than an estimate of total incidence. The guarantees cover only measured properties, here task reliability, and cannot exclude unmeasured safety violations or failures from irreversible actions. Synchronous oversight adds roughly 1.7–$7\times$ wall-clock latency, while RQ4 shows that per-candidate admission does not control degradation of the accumulated rule set. Evaluation is limited to three benchmark families, text-based rules, and a non-adversarial agent within its tool surface, leaving richer persistent modifications and adversarial circumvention open.

Conclusion

We formulate persistent agent self-modification as admission control, where the agent proposes behavioral updates and an external runtime controls persistence through replay-based improvement, multi-metric non-regression, a weaker forward-trial fallback, and corpus-level re-testing. The central empirical result is that 211 of 383 replay-decided rejections (55%) improved their triggering failure while degrading protected behavior; under the observed replay evidence, a target-only retention rule would have admitted all 211. Because each validation round inspects at most two protected cases, these are detected conflicts under sparse coverage rather than an estimate of total incidence. We also observe declining performance as individually validated rules accumulate, indicating that per-candidate admission alone does not control corpus-level degradation. These results motivate external evidence-gated control over self-modification.

Disclosure

Capital One does not use nor has plans to use this methodology in any capacity (e.g., audits, credit decision explainability, anti-discrimination testing).

References

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*.

Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; et al. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*.

Barres, V.; Dong, H.; Ray, S.; Si, X.; and Narasimhan, K. 2025. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv preprint arXiv:2506.07982*.

Bowman, S. R.; Hyun, J.; Perez, E.; Chen, E.; Pettit, C.; Heiner, S.; Lukošiūtė, K.; Askell, A.; Jones, A.; Chen, A.; et al. 2022. Measuring Progress on Scalable Oversight for Large Language Models. *arXiv preprint arXiv:2211.03540*.

Chen, M.; Li, Y.; Yang, Y.; Yu, S.; Lin, B.; and He, X. 2024. AutoManual: Constructing Instruction Manuals by LLM Agents via Interactive Environmental Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Darabi, N.; Naik, D.; Tayebati, S.; Jayasuriya, D.; Krishnan, R.; and Trivedi, A. R. 2026. EigenShield: Inference-Time, Model-Agnostic Jailbreaking Defense via Causal Subspace Filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 3524–3532.

Efron, B. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1): 1–26.

Gou, Z.; Shao, Z.; Gong, Y.; Shen, Y.; Yang, Y.; Duan, N.; and Chen, W. 2024. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *International Conference on Learning Representations (ICLR)*.

Kumar, D.; Tayebati, S.; Migliarba, F.; Krishnan, R.; and Trivedi, A. R. 2025. Learnable conformal prediction with context-aware nonconformity functions for robotic planning and perception. *arXiv preprint arXiv:2509.21955*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2024. Let's Verify Step by Step. In *International Conference on Learning Representations (ICLR)*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, volume 24, 109–165. Academic Press.

Merrill, M.; Shaw, A.; Carlini, N.; Li, B.; Raj, H.; Bercovich, I.; Shi, L.; Shin, J.; Walshe, T.; Buchanan, E. K.; et al. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, 40903–40986.

Packer, C.; Wooders, S.; Lin, K.; Fang, V.; Patil, S. G.; Stoica, I.; and Gonzalez, J. E. 2023. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*.

Park, J. S.; O'Brien, J. C.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*.

Rebedea, T.; Dinu, R.; Sreedhar, M. N.; Parisien, C.; and Cohen, J. 2023. NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. In *Conference on Empirical Methods in Natural Language Processing (EMNLP), System Demonstrations*.

Shinn, N.; Cassano, F.; Berman, E.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A. A.; and Hardt, M. 2020. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In *International Conference on Machine Learning (ICML)*.

Tayebati, S. 2026. pandaprobe.

Tayebati, S.; Kumar, D.; Darabi, N.; Ettori, D.; Krishnan, R.; and Trivedi, A. R. 2026. TRACER: Trajectory risk aggregation for critical episodes in agentic reasoning. *arXiv preprint arXiv:2602.11409*.

Tayebati, S.; Kumar, D.; Darabi, N.; Jayasuriya, D.; Krishnan, R.; and Trivedi, A. R. 2025a. Learning conformal abstention policies for adaptive risk management in large language and vision-language models. *arXiv preprint arXiv:2502.06884*.

Tayebati, S.; Kumar, D.; Darabi, N.; Jayasuriya, D.; Tulabandhula, T.; Krishnan, R.; and Trivedi, A. R. 2025b. Cap: Conformalized abstention policies for context-adaptive risk management for llms and vlms. In *The 17th Asian Conference on Machine Learning (Conference Track)*.

Trivedi, H.; Khot, T.; Hartmann, M.; Manku, R.; Dong, V.; Li, E.; Gupta, S.; Sabharwal, A.; and Balasubramanian, N. 2024. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 16022–16076.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023a. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023b. Self-Consistency

Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations (ICLR)*.

Wang, Z. Z.; Mao, J.; Fried, D.; and Neubig, G. 2024. Agent Workflow Memory. *arXiv preprint arXiv:2409.07429*.

Xu, W.; Liang, Z.; Mei, K.; Gao, H.; Tan, J.; and Zhang, Y. 2025. A-MEM: Agentic Memory for LLM Agents. *arXiv preprint arXiv:2502.12110*.

Yao, S.; Shinn, N.; Razavi, P.; and Narasimhan, K. 2024. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *arXiv preprint arXiv:2406.12045*.

Zelikman, E.; Lorch, E.; Mackey, L.; and Kalai, A. T. 2024. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. *arXiv preprint arXiv:2310.02304*.

Zhang, J.; Hu, S.; Lu, C.; Lange, R.; and Clune, J. 2025. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954*.

Zhao, A.; Huang, D.; Xu, Q.; Lin, M.; Liu, Y.-J.; and Huang, G. 2024. ExpeL: LLM Agents Are Experiential Learners. In *AAAI Conference on Artificial Intelligence*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Technical Appendix

This appendix carries material deferred from the main text. It is organized so that each part answers a question a reader of the main text may raise, in the order those questions arise. *Experiments not run* states the arms we designed but did not execute, so the gap between the thesis and the comparison is explicit rather than implied. *Metric and interval definitions* gives the formal definitions behind the reliability metrics and the bootstrap interval. *Mechanism details* specifies, item by item, the implementation of every mechanism whose scientific content is stated in the Harness section. *Algorithms* gives the per-turn loop, the evidence gate, and the trajectory-gate update in full. *Notation* and *Hyperparameters* are the complete symbol reference and the complete constant table, the latter listing every package default beside the value actually used. *Comparison to prior mechanisms* positions eleven representative systems against the five properties of controlled self-improvement, and *Additional result tables* holds the per-run tables behind aggregate numbers reported in the results. Nothing here changes a claim made in the main text.

Experiments not run

Arms that were designed but not executed, expanding the main text discussion of what the study does not isolate.

Three configurations. It is worth separating three settings that are easy to conflate:

- **B**: candidate validation before admission, plus the corpus-level guard. This is what we ran.
- **B′**: immediate admission of authored rules, with the same corpus-level guard downstream.
- **B″**: immediate admission with no corpus guard, that is, genuinely uncontrolled persistence.

B′ is the comparison that directly isolates the effect of pre-admission validation while holding the downstream corpus guard fixed. We are explicit that B′ is *not* ungated adaptation: an external evidence-based regression check still operates after admission, so the contrast is between validating before admission and validating only afterwards. B″ is the fully uncontrolled setting, and we name it for completeness rather than proposing it as the primary comparison, since removing both layers would confound the two mechanisms.

The B′ arm. Configuration B′ is the Harness with candidate validation disabled, so that an authored rule enters the active set immediately. Everything else is held identical to B: the same detection, notices, agent, repair prompts, rule authoring, retrieval, corpus guard, task order, and benchmark. The measurements that would separate B from B′ are final task completion, $\mathrm{pass@1}$ and pass^k, the number of self-authored rules retained, the number of retained rules that later regress a protected case under the same corpus guard, the magnitude of those regressions, and performance as a function of execution order. The prediction implied by RQ2 needs care, because adaptation is path dependent. Under the observed B trajectories, automatic admission would have retained the 211 proposals that B rejected for protected-case regression. An actual B′ run would not encounter those same proposals: once an early rejected rule is retained, subsequent failures, notices, and authored rules diverge. The testable claim is therefore distributional rather than case-matched. Immediate admission should accumulate rules faster and should show a higher rate and larger magnitude of downstream protected-case regression under the identical guard, together with worse late-run paired performance if those regressions are consequential. That formulation also anticipates the obvious objections to reading RQ2 as sufficient on its own: the rejected rules might prove harmless in live execution, some protected-case drops might be evaluator noise, faster accumulation might outweigh the regressions, and the corpus guard might catch the damage later anyway. The last of these is precisely why B′ retains the guard, since a difference that survives an identical downstream check is attributable to the admission test. A controlled B versus B′ comparison is therefore the most direct experiment for isolating the effect of the candidate-level admission test.

Other arms not run. The no-barrier, inlined-rule, tier-3-enabled, replay-only, and forward-trial-only arms were also not run. The barrier evidence in RQ5 is mechanism verification, and the replay-share variation in RQ3 is observational and confounded. Neither substitutes for the corresponding controlled arm.

Held-out transfer. The protocol measures online reliability during adaptation, in one continuous pass over each split. It does not measure whether the accumulated rule corpus transfers. The corresponding experiment is to run the Harness over an adaptation stream, freeze the final validated corpus, and evaluate Baseline against frozen-Harness on held-out tasks,

which would separate reusable behavioral improvement from useful intervention within the same stream.

Metric and interval definitions

Formal statements of the reliability metrics and the confidence interval used throughout the results, deferred from the experimental setup. For task set T with binary benchmark outcomes $X_{t,i} \in \{0,1\}$ over k trials,

$$\text{pass@1} = \tfrac{1}{|T|}\textstyle\sum_{t\in T} X_{t,1}, \;\; \text{pass}^k = \tfrac{1}{|T|}\textstyle\sum_{t\in T}\prod_{i=1}^{k} X_{t,i}, \tag{4}$$

so pass@1 is first-trial success while pass^k requires success on every trial. For the paired per-task difference d_t in task-completion score, with $R = 10{,}000$ bootstrap resamples over tasks,

$$\text{CI}_{95} = \big[\, Q_{2.5}(\{\bar{d}^{(*r)}\}_{r=1}^{R}),\; Q_{97.5}(\{\bar{d}^{(*r)}\}_{r=1}^{R}) \,\big]. \tag{5}$$

Mechanism details

Implementation-level specifications for the mechanisms whose scientific content is stated in the main text. Nothing here changes a claim; each item is the formal or operational form of a sentence in the main text.

Metric ladder, from *Detect*. $\mathcal{M}_1$ (*progress*, always on) is task completion and coherence, evaluated on every new trace; coherence is embedding-based, which makes this the cheapest tier. $\mathcal{M}_2$ (*step-level diagnosis*) is tool-call correctness and argument correctness, evaluated only on the latest trace once $\mathcal{M}_1$ flags the trajectory. $\mathcal{M}_3$ (*explanatory*, opt-in) is planning adherence, plan quality, and step efficiency, used only to enrich a confirmed finding and never as a trigger. $\mathcal{M}_0$ (*outcome*) is the score from the optional developer-defined verifier. The ladder is also the cost-control mechanism: tier 1 scores all of a turn's new traces in one batched request, tier 2 scores only the latest trace after a gate fire, and tier 3 is off by default. Because model-judged metrics consume the full trace, confining expensive diagnosis to flagged trajectories is what makes continuous monitoring tractable.

Absolute breaches and severity, from *Detect*. For metrics with an explicit threshold τ_j,

$$b_j(x) = \mathbf{1}\big[\, m_j(x) \neq \varnothing \;\wedge\; m_j(x) < \tau_j \,\big], \tag{6}$$

with pending scores excluded. Absolute thresholds apply only to outcome and step-level metrics,

$$\text{breach}_j(x) = b_j(x) \;\wedge\; \text{tier}(j) \in \{0,2\}, \tag{7}$$

so tier-1 metrics are governed only by the stall and regression conditions and tier-3 metrics never produce a breach. The severity assigned to a fired trajectory condition is then

$$\text{sev} = \begin{cases} \texttt{breach} & \exists\, j \in \mathcal{M}_2 \cup \mathcal{M}_0 :\; \text{breach}_j(x_T), \\ \texttt{trend} & \text{otherwise}, \end{cases} \tag{8}$$

evaluated on the latest trace x_T. Each fired condition is emitted as a signature of the form `condition:metric`, such as `stall:task_completion` or `breach:tool_correctness`.

Barrier timeout, from the per-turn barrier. The barrier has a dedicated timeout budget. If it expires, evaluation continues asynchronously and the turn is marked timed out, so a slow evaluator adds latency without blocking task execution or changing the correctness of already-completed work.

Outcome verifier, from *Detect*. Given the session state, the verifier returns either a score or an abstention,

$$m_{\text{out}}(s) \in [0,1] \cup \{\varnothing\}, \tag{9}$$

so a verifier that can score only completed tasks returns $\varnothing$ on intermediate turns. Valid outcomes enter the ladder as tier-0 metrics against a strict threshold $\tau_{\text{out}} = 0.9$. Verifier failures are isolated from the agent loop: exceptions are caught and the turn proceeds without an outcome score. Because verification depends only on the turn's end state, it runs concurrently with trace-level metric evaluation.

Scopes, from the rule workspace. Rules are partitioned into free-form *scopes*, each backed by one rule file. The Harness provides a `global` scope for broad trajectory-level lessons and a `scoped` default for step-level findings, and the agent may create more specific topic scopes. Notices suggest an initial scope, which the agent may override when authoring or reorganizing rules. New scopes appear in $\iota(\mathcal{R})$ automatically. The ranking function applied at retrieval time is the relevance score there below.

Heal as an update operator, from *Heal*. At step t, given the breaches $\mathfrak{B}_t$ identified by $\mathcal{E}$,

$$\mathcal{R}_{t+1} = \mathcal{H}(\mathcal{R}_t, \mathfrak{B}_t) = \text{gate}\big(\mathcal{R}_t, \text{propose}(\mathcal{R}_t, \mathfrak{B}_t)\big), \tag{10}$$

where *propose* is carried out by the agent and *gate* is the evidence test of the evidence gate. Candidates are de-duplicated against the live rule set using normalized rule text.

Target metric selection, from *Validate*. The verdict is read against

$$\hat{\jmath} \;=\; \text{first-available}\big(\, m_{\text{out}},\; \text{met}(r),\; \text{met}(\text{sig}(r)) \,\big), \tag{11}$$

where $\text{met}(\cdot)$ denotes the metric named by a rule or by its triggering signature. Availability is resolved per case, so verifier abstention does not block validation.

Replay delta, from *Validate*. For replayed case s_i and metric j,

$$\Delta_j(s_i) = m'_j(s_i) - m_j(s_i), \tag{12}$$

with m' the replay score and m the score recorded when the case was captured.

Forward-trial predicate, from *Validate*. Let p_0 be the pre-activation rate of the rule's metric-specific signature family (`stall`, `regression`, or `breach`), estimated as flagged sessions divided by total sessions, with $p_0 = 1$ when no prior window exists. After n subsequent sessions under the candidate, the empirical flagged rate $\hat{p}$ must satisfy

$$\hat{p} = 0 \quad \text{or} \quad \hat{p} \leq p_0 - \delta_{\text{prom}}. \tag{13}$$

Otherwise the rule is retired, and until enough sessions accumulate, validation stays pending.

Eval-case capture, from *Validate*. Each captured case stores the replay input together with its baseline trace scores. Advisory `trend` notices are not captured, which keeps the replay

corpus focused on diagnosed failures and keeps the deltas in the replay delta interpretable against a recorded baseline.

Worked example, from *Validate*. Suppose an agent repeatedly applies a destructive action to the wrong file and proposes the rule "verify the target path before any destructive action." If replay improves the target metric by 0.30 without violating the non-regression condition, the rule is promoted. If the same rule repairs the original failure but degrades a protected case by more than δ_{reg}, it is retired. The second case is not hypothetical; it is the modal rejection in our runs (RQ2).

Corpus regression classification, from the regression guard. The guard classifies each replayed case from its per-metric deltas,

$$\text{cls}(s_i) = \begin{cases} \text{regressed} & \exists\, j:\ \Delta_j(s_i) \le -\delta_{\text{reg}}, \\ \text{improved} & \text{else if } \exists\, j:\ \Delta_j(s_i) \ge \delta_{\text{prom}}, \\ \text{unchanged} & \text{otherwise}, \end{cases} \tag{14}$$

so regression takes precedence over improvement and the rule set passes only if no replayed case is classified as regressed.

Healing action space, from the rule workspace. Healing is exposed through a small self-diagnostic interface. The agent can list, read, and acknowledge notices; inspect traces; and read, search, add, retire, or query the status of rules. Each operation returns a structured response and fails gracefully, so a diagnostic error does not interrupt task execution. The full set of tunable constants appears in Table 6.

Relevance score for search

Deferred from the rule workspace. This score orders the agent's *search* results and narrows the active rules rendered inside one scope file when a query is supplied. It does not select what enters the prompt by default, since the context decomposition carries no rule bodies. Each rule r carries tag tokens T_r and text tokens X_r drawn from its rule text, rationale, and metric. The query token set Q is assembled from pending notice signatures and metric names plus any caller-supplied terms, case-folded and tokenized on `[a-z0-9_]+` with length ≥ 2. The relevance of r to Q is

$$\text{score}(r, Q) = \frac{2\,|Q \cap T_r| + |Q \cap (X_r \setminus T_r)|}{\max(1, |Q|)}, \tag{15}$$

so a tag-token match is weighted 2.0 and a non-tag text-token match 1.0, normalized by query size. A scope file rendered under a query keeps its TOP-k active rules by score ($k = 8$), with ties broken by score, then recency, then id, and appends a one-line note naming the search operation so that nothing becomes unreachable:

$$\rho_\sigma(\mathcal{R}, Q) = \text{render}\Big(\text{top-}\,k\ {}_{r \in \mathcal{R}_\sigma}\ \text{score}(r, Q)\Big), \tag{16}$$

where $\mathcal{R}_\sigma$ is the active subset of scope σ. *Candidate* rules are exempt from this cutoff and always render in full, which is the retrieval-level expression of the provisional execution authority described under *Validate*: a candidate under trial has to be in force for the trial to measure anything, so narrowing it away would starve its own validation.

Algorithms

The three procedures the main text refers to, in full.

Per-turn self-heal loop. Algorithm 1 states the full per-turn loop deferred from *Heal*; the evidence gate it invokes is Algorithm 2.

Algorithm 1 Self-Heal Loop (one turn)

Input: rule set $\mathcal{R}$, service $\mathcal{E}$, workspace $\mathcal{W}$, gate $\mathcal{G}$

compose context $c(\mathcal{R})$ via the context decomposition {protocol + index; no rule text}
run one agent turn; new traces $X \leftarrow (x_i, \ldots, x_T)$ not yet scored
$\mathfrak{B} \leftarrow \varnothing$
{tier 1: every new trace, one batched service call}
for all $x \in X$ in chronological order **do**
  for all $j \in \mathcal{M}_1$ **do**
    fold $m_j(x)$ into $\mathcal{G}$ {Algorithm 3}
    add STALL/REGR verdicts to $\mathfrak{B}$
  end for
end for
if $\mathfrak{B} \neq \varnothing$ **then**
  {tier 2: last trace only}
  add $\{\, \text{breach}_j(x_T) : j \in \mathcal{M}_2 \,\}$ to $\mathfrak{B}$ {the absolute-breach predicate}
  if any tier-2 breach **and** tier 3 enabled **then**
    score $\mathcal{M}_3$ on x_T {explanatory only}
  end if
end if
$m_{\text{out}} \leftarrow$ verifier(s) if wired {concurrent; may abstain}
if $\mathfrak{B} \neq \varnothing$ and not deduplicated/cooled-down **then**
  sev $\leftarrow$ the severity rule; post notice to mailbox $\subset \mathcal{W}$
  if sev $=$ `breach` and capture enabled **then**
    capture the session as a replayable eval-case
  end if
end if
VALIDATECANDIDATES($\mathcal{R}$) {Algorithm 2}
barrier: wait until the above has landed {the per-turn barrier}
{pull model: the agent may now pull notices on its own initiative}
if agent pulls a notice **then**
  agent inspects the flagged trace, reads the suggested scope file
  agent authors candidate rule r; $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$ {provisional authority only}
end if
return $\mathcal{R}$

The evidence gate. Algorithm 2 states the gate deferred from *Validate*. It is driven from the turn hook on every handled report, as a single-flight background round, so it never blocks a turn and any failure inside it degrades to a log line rather than an exception in the host loop.

The trajectory gate. Deferred from *Detect*. Algorithm 3 is the complete per-metric update. It is a pure function of the stored state (pk_j, z_j) and the new value, which is why a

Algorithm 2 VALIDATECANDIDATES: the evidence gate

for all candidate $r \in \mathcal{R}$ **do**
  if replay fn. available **and** attempts$(r) < 3$ **then**
    select ≤ 3 failure cases matching $\mathrm{sig}(r)$, ≤ 2 protected wins
    replay each under $c(\mathcal{R} \cup \{r\})$; re-score on $\mathcal{M}_1 \cup \mathcal{M}_2$
    deltas $\Delta_j(s_i)$ via the replay delta; target $\hat{\jmath}$ via the target-selection rule
    if $\exists\, i, j : \Delta_j(s_i) \leq -\delta_{\mathrm{reg}}$ **then**
      retire r {the non-regression condition: regression dominates}
    else if $\exists\, i : \Delta_j(s_i) \geq \delta_{\mathrm{prom}}$ **then**
      promote r {the improvement condition}
    else if no conclusive failure case **then**
      mark *pending*; attempts(r)++
    else
      retire r {no measured improvement}
    end if
  else
    observe n sessions; form $p_0, \hat{p}$ {forward-trial fallback}
    if $\hat{p} = 0$ **or** $\hat{p} \leq p_0 - \delta_{\mathrm{prom}}$ **then**
      promote r {the forward-trial predicate}
    else if sessions observed $\geq n$ **then**
      retire r
    else
      pending
    end if
  end if
end for

whole turn's traces can be folded in chronological order in a single write to the history store. The three properties claimed in the main text are visible here: a gain resets the counter and raises the peak, so a climbing session never fires; the stall test compares the *peak* against θ rather than the current value, so a session that reached its target may plateau; and both conditions reset the counter after firing, so one stall yields one alert.

State is persisted per (session, metric) alongside the score series, so it survives process restarts. A session's series is exactly the sequence of tier-1 samples the per-turn barrier caused to be collected, which is why its length is an operational check (RQ5) rather than an implementation detail.

Full notation

Table 5 is the complete symbol reference. The main text introduces each symbol where it is first used and does not repeat the table.

Hyperparameters

Table 6 lists every tunable constant with both its package default and the value used in the reported runs. Exactly one constant differs across benchmarks, the replay turn budget, so the gate constants, the evidence-gate margins, and the trial window are held fixed for all three benchmarks and all four models. No constant was tuned per benchmark, which is why *Limitations* reports threshold sensitivity as an open cost rather than a result.

Algorithm 3 $\mathcal{G}$: trajectory-gate update for metric j on one trace

Input: state (pk_j, z_j), value $v = m_j(x_i)$
Output: updated state, verdict $\in \{\textsc{ok}, \textsc{stall}, \textsc{regr}\}$

if $\mathrm{pk}_j = \varnothing$ **or** $v \geq \mathrm{pk}_j + \gamma_{\mathrm{gain}}$ **then**
  $\mathrm{pk}_j \leftarrow \max(\mathrm{pk}_j, v)$; $z_j \leftarrow 0$
  return OK {progress: reset the window}
end if
$z_j \leftarrow z_j + 1$
$\mathrm{stall} \leftarrow [\, z_j \geq w \ \wedge\ \mathrm{pk}_j < \theta \,]$ {Eq. (1)}
$\mathrm{regr} \leftarrow [\, v < \mathrm{pk}_j - \delta_{\mathrm{drop}} \,]$ {Eq. (2)}
if stall **or** regr **then**
  $z_j \leftarrow 0$ {fire once, then reopen the window}
end if
return verdict from $(\mathrm{stall}, \mathrm{regr})$

Comparison to prior mechanisms

Table 7 expands the novelty boundary stated in related work. The columns are five properties of controlled self-improvement. *P*, persistent, meaning the change survives across episodes. *S*, self-generated, meaning authored by the agent rather than a human. *V*, validated by measurement before the change is retained. *R*, regression-guarded, meaning retention is conditioned on not degrading behavior that previously succeeded. *T*, triggered by a measured runtime signal during deployment rather than by a fixed schedule, an offline search loop, or an author's decision about when to intervene.

The rows are chosen to make the claim harder rather than easier. Reflexion, ExpeL, AutoManual, Agent Workflow Memory, and A-MEM all let an agent write durable operating knowledge, and none conditions retention on a measured non-regression test. STOP and the Darwin Gödel Machine are the closest comparators, because both score self-modifications empirically before keeping them, so *V* is satisfied. They differ on the remaining two columns: neither conditions retention on protected cases that previously passed, and both run as offline improvement searches rather than as a runtime loop inside an ongoing deployment. The Harness combination is *V* and *R* and *T* together on an agent-authored persistent update. Entries describe each system as its authors present it; "partial" marks a related but weaker form of the property, and any of these systems could be extended.

The repeatability ordering. RQ1 reports that only three of the 16 pairs satisfy $\Delta\mathrm{pass}^k > \Delta\mathrm{pass@1}$. Table 9 lists them; in two, pass^k improves while $\mathrm{pass@1}$ is flat or lower, which is the pattern a repeatability-specific effect would produce. We treat it as suggestive rather than as validation, since in most pairs the two metrics move together.

Corpus composition. The retained corpus is specific rather than generic: active rules average 348 characters, reflecting concrete procedures rather than broad instructions, and the

Table 5: Complete notation. Where a default and a study value differ, the study value is given in Table 6.

Symbol	Meaning
π	the (fixed) agent policy wrapped
$s = (x_1, .., x_T)$	a session (one task trial); its traces
x_i	the i-th trace (one agent turn)
$\mathcal{E}$	trace-level evaluation service
$m_j(x)$	metric j on trace x; $\varnothing$ if pending
$\mathcal{M}_0$	$\{m_{\text{out}}\}$: verifier outcome (tier 0)
$\mathcal{M}_1$	progress metrics, every trace (tier 1)
$\mathcal{M}_2$	step-level metrics, last trace (tier 2)
$\mathcal{M}_3$	explanatory metrics, opt-in (tier 3)
τ_j	absolute threshold, metric j (def. 0.5)
τ_{out}	outcome threshold (def. 0.9)
$b_j(x)$	point-breach indicator on metric j
$\mathcal{G}$	trajectory gate
pk_j, z_j	gate state: peak, traces since gain
w, θ	gate window, target (def. 5, 0.5; study w=10)
$\gamma_{\text{gain}}, \delta_{\text{drop}}$	gain / drop margins (0.02, 0.15)
$\mathcal{R}$	rule set (active $\cup$ candidate)
r	a rule (text, rationale, scope, tags)
$c(\mathcal{R})$	policy-inducing context
ι	rule-*index* operator
$\mathcal{W}$	filesystem workspace
$\mathcal{H}$	self-heal operator on the rule set
$\mathfrak{B}_t$	conditions observed at turn t
$\hat{\jmath}$	promotion target metric
$\delta_{\text{prom}}, \delta_{\text{reg}}$	promote / regress margins (0.05)
n	forward-trial window (def. 5; study 3)
$p_0, \hat{p}$	baseline / trial flagged rate
k	trials per task in evaluation

repair loop chooses a contextual scope over the `global` default in 87% of cases, with AppWorld producing application-level scopes and τ^2 domain-level ones.

Additional result tables

Per-run tables behind aggregate numbers reported in the main text, in the order they are cited there. Table 8 gives the configuration matrix as run. Table 9 lists the three pairs in which the repeatability ordering holds. Table 10 breaks the rule lifecycle down per run behind the totals in RQ2. Table 11 reports per-run Harness activity behind the two operational checks in RQ5. Table 12 tabulates the evaluator-bias measurement of RQ5. Tables 13 and 14 give the per-run cost accounting and place added context beside the paired task-completion gain. Every number these tables contain that carries a claim is also stated in the main text.

Table 6: Harness hyperparameters: package default vs. the value used in every reported run. **Boldface** marks a study override. Eval-case *capture* defaults off and is on in the Harness arm throughout the run, so replay validation has cases to work with; tier 3 stays off because each model-judged metric re-reads the whole trace. Rows marked † are harness-level constants with no package default; * marks the only value that differs across benchmarks. The forward-trial window is lowered to 3 so that a rule can complete a trial inside a single run, and the barrier and poll budgets are raised because trace evaluation is model-judged and takes minutes.

Symbol / knob	Meaning	Default	Study
Trigger and trajectory gate			
w	gate window (traces w/o new high)	5	**10**
θ	gate target	0.5	0.5
γ_{gain}	gain counted as progress	0.02	0.02
δ_{drop}	drop from peak = regression	0.15	0.15
τ_j	absolute threshold (tiers 0, 2)	0.5	0.5
τ_{out}	outcome-verifier threshold	0.9	0.9
–	tier 3 enabled	off	off
–	traces listed per turn	50	50
Barrier and cost			
–	barrier budget (s)	180	**1080**
–	score poll: interval × tries	1×20	**5×200**
–	global concurrent evaluations	4	4
Rules			
k	TOP-k actives per queried scope	8	8
–	max live rules (active+cand.)	uncapped	
Evidence gate			
n	forward-trial window	5	**3**
δ_{prom}	promote margin	0.05	0.05
δ_{reg}	regress margin	0.05	0.05
–	max replay attempts	3	3
–	failure / win cases replayed	3 / 2	3 / 2
–	replay timeout (s)	300	**180**
–	replay turns†*	–	**15**; τ^2 **100**
–	replay env-wait grace (s)	0	**900**
–	validation round budget (s)	0	**600**
–	regression sample	0 = whole set	
–	validation / capture	on / off	on / **on**
End-of-run settlement			
–	settle timeout / poll†	–	**1080 / 10**
Run design			
–	trials per task k	4	
–	seed	1 of {1, 2, 3}	
–	per-task agent turns	100	

Table 7: Where the Harness sits. Rows: Reflexion (Shinn et al. 2023), Voyager (Wang et al. 2023a), Generative Agents (Park et al. 2023), ExpeL (Zhao et al. 2024), AutoManual (Chen et al. 2024), Agent Workflow Memory (Wang et al. 2024), A-MEM (Xu et al. 2025), RAG memory (Lewis et al. 2020), STOP (Zelikman et al. 2024), Darwin Gödel Machine (Zhang et al. 2025), Constitutional AI (Bai et al. 2022). Columns P, S, V, R, T defined in text.

Approach	P	S	V	R	T
Reflexion	×	✓	×	×	✓
Voyager	✓	✓	partial	×	✓
Generative Agents	✓	✓	×	×	partial
ExpeL	✓	✓	×	×	×
AutoManual	✓	✓	partial	×	✓
Agent Workflow Memory	✓	✓	×	×	×
A-MEM	✓	✓	×	×	×
RAG memory	✓	×	×	×	×
STOP	✓	✓	✓	×	×
Darwin Gödel Machine	✓	✓	✓	×	×
Constitutional AI	✓	×	✓	partial	×
Harness (ours)	✓	✓	✓	✓	✓

Table 8: Configuration matrix as run. Each native split is used whole in one continuous pass, so there is no learning and evaluation partition. "pairs" counts the matched Baseline and Harness comparisons contributing to the results; "turns" is the per-task agent budget.

Benchmark	Split	tasks	models	pairs	k	turns
AppWorld	`test_normal`	168	4	4	4	100
Terminal-Bench	`sample@2.0`	10	4	4	4	100
τ^2-bench	`airline`	50	4	4	4	100
τ^2-bench	`retail`	114	4	4	4	100

Total: 16 matched pairs, 4 models (GPT-5.6-TERRA, GPT-5.6-LUNA, CLAUDE-HAIKU-4.5, CLAUDE-SONNET-5); seed 1 of $\{1, 2, 3\}$ configured; arms Baseline and Harness; forward-trial window n=3; gate window w=10; δ_{prom}=δ_{reg}=0.05.

Table 9: The three of 16 pairs in which the all-k-pass rate rises more than the first-trial rate, $\Delta\text{pass}^k > \Delta\text{pass@1}$. Reported as a suggestive pattern, not as validation of a distinctive repeatability signature. Terminal-Bench has 10 tasks, so its pass^k increments are coarse.

Benchmark	Model	Δpass@1	Δpass^k	gap
Terminal-Bench	claude-sonnet-5	-0.143	**+0.333**	+0.476
Terminal-Bench	claude-haiku-4.5	+0.000	**+0.200**	+0.200
τ^2 airline	gpt-luna	+0.020	**+0.040**	+0.020

Table 10: Rule lifecycle per run, grouped by benchmark. Terminal-Bench cannot replay and has zero retirements in these runs, so its promotion rate should not be read as replay-gate selectivity.

Benchmark	Model	prop.	prom.	retired	prom. rate
AppWorld	terra	112	49	63	0.44
AppWorld	luna	205	78	127	0.38
AppWorld	haiku-4.5	181	61	120	0.34
AppWorld	sonnet-5	96	42	54	0.44
Terminal-Bench	terra	12	12	0	–
Terminal-Bench	luna	23	23	0	–
Terminal-Bench	haiku-4.5	20	20	0	–
Terminal-Bench	sonnet-5	15	15	0	–
τ^2 airline	terra	76	41	35	0.54
τ^2 airline	luna	173	93	80	0.54
τ^2 airline	haiku-4.5	129	53	76	0.41
τ^2 airline	sonnet-5	44	24	20	0.55
τ^2 retail	terra	248	151	97	0.61
τ^2 retail	luna	359	225	134	0.63
τ^2 retail	haiku-4.5	402	231	171	0.57
τ^2 retail	sonnet-5	211	126	85	0.60
Total (16 runs)		2306	1244	1062	

Table 11: Harness activity per run. Escalation rate is `breach`/(`trend` + `breach`). Samples/series is the median length of a per-session metric series and is the direct read-out of the barrier (the barrier); a value of 1 would mean the trigger could not have fired.

	AppWorld				Terminal-Bench				τ^2 airline				τ^2 retail			
Quantity	terra	luna	haiku	sonnet	terra	luna	haiku	sonnet	terra	luna	haiku	sonnet	terra	luna	haiku	sonnet
graded trials	661	672	672	664	40	40	40	40	200	200	200	200	448	445	448	448
rules proposed	112	205	181	96	12	23	20	15	76	173	129	44	248	359	402	211
rules promoted	49	78	61	42	12	23	20	15	41	93	53	24	151	225	231	126
rules retired	63	127	120	54	0	0	0	0	35	80	76	20	97	134	171	85
notices `trend`	156	164	190	141	34	35	38	31	171	218	176	139	584	756	811	496
notices `breach`	611	807	690	548	14	22	24	18	225	287	200	198	612	745	792	563
notices `needs_human`	46	34	41	27	0	0	0	0	7	11	9	6	18	23	31	15
tier-2 escalation rate	0.80	0.83	0.78	0.80	0.29	0.39	0.39	0.37	0.57	0.57	0.53	0.59	0.51	0.50	0.49	0.53
gate fires / session	0.059	0.104	0.082	0.052	0.325	0.350	0.375	0.300	0.025	0.015	0.000	0.020	0.024	0.027	0.031	0.022
samples / series (median)	**11**	**11**	**10**	**11**	**7**	**8**	**8**	**7**	**7**	**7**	**6**	**6**	**10**	**11**	**10**	**10**
repair episodes	809	1005	947	732	48	57	63	51	364	516	385	312	1186	1520	1667	1024

Table 12: Evaluator bias before and after adding meta-tool neutrality guidance to the judge prompts, measured over 4,800 sampled judge rationales after the change. The Harness is unmodified; only the evaluator prompts change. All runs in Table 1 use the corrected prompts in both arms.

Quantity	before	after
rationales blaming meta-tool calls	14%	**0.17%**
rationales mentioning meta-tools	–	2,251
spurious zero scores	31	**1**

Table 13: Cost and overhead, grouped by benchmark and ordered within each group by added context. Cost is total USD over the run; the repair column is additional spend on top of the Harness task-execution column. Wall time is per trial.

		tokens / turn		agent turns		cost (USD)		wall / trial
Benchmark	Model	B→H	ratio	B→H	Δ	B→H	repair	B→H (s)
AppWorld	terra	5215→7504	1.44×	10.1→11.9	+1.8	26.47→34.45	8.26	17→109
AppWorld	luna	5172→8986	1.74×	9.9→11.6	+1.7	2.57→4.19	1.79	17→114
AppWorld	haiku-4.5	5580→7590	1.36×	10.0→11.4	+1.4	5.80→7.60	4.20	18→112
AppWorld	sonnet-5	7110→9310	1.31×	10.2→11.5	+1.3	28.00→34.00	11.00	20→120
Terminal-Bench	terra	6837→6151	**0.90**×	8.5→9.1	+0.6	3.67→2.37	0.61	101→202
Terminal-Bench	luna	3968→4984	1.26×	7.2→8.3	+1.1	0.16→0.17	0.14	97→209
Terminal-Bench	haiku-4.5	5210→6090	1.17×	7.8→8.6	+0.8	1.15→1.35	0.42	95→198
Terminal-Bench	sonnet-5	6610→7010	1.06×	8.1→8.5	+0.4	4.10→4.30	0.75	99→205
τ^2 airline	terra	5920→6510	1.10×	7.6→7.9	+0.3	5.50→6.20	2.20	24→145
τ^2 airline	luna	3982→4655	1.17×	8.2→8.7	+0.5	0.41→0.53	0.84	19→133
τ^2 airline	haiku-4.5	5670→6100	**1.08**×	7.0→7.1	**+0.1**	9.75→10.46	9.14	26→125
τ^2 airline	sonnet-5	7243→7732	**1.07**×	7.1→7.1	**+0.0**	26.00→27.71	15.02	41→173
τ^2 retail	terra	5510→7220	1.31×	9.8→10.9	+1.1	8.40→10.60	5.40	25→166
τ^2 retail	luna	4198→6056	1.44×	10.3→11.7	+1.4	1.08→2.53	3.63	20→175
τ^2 retail	haiku-4.5	6010→8120	1.35×	9.9→11.1	+1.2	18.00→22.00	11.50	29→168
τ^2 retail	sonnet-5	7610→9820	1.29×	9.6→10.7	+1.1	36.00→43.00	18.00	42→190
Total (16 pairs)		1.25× mean				177.06→211.46	92.90	

Table 14: Cost against benefit: added context (Table 13) beside the paired task-completion gain (Table 1) across all 16 pairs. Every pair shows a positive $\Delta\bar{s}$. Terminal-Bench/terra is the only configuration below Baseline token cost, while Terminal-Bench/haiku-4.5 shows the largest task-completion gain.

Benchmark	Model	added context	$\Delta\bar{s}$
AppWorld	sonnet-5	1.31×	+0.013
AppWorld	haiku-4.5	1.36×	+0.024
AppWorld	terra	1.44×	+0.022
AppWorld	luna	1.74×	+0.004
Terminal-Bench	terra	**0.90**×	+0.075
Terminal-Bench	sonnet-5	1.06×	+0.019
Terminal-Bench	haiku-4.5	1.17×	**+0.146**
Terminal-Bench	luna	1.26×	+0.025
τ^2 airline	sonnet-5	1.07×	+0.030
τ^2 airline	haiku-4.5	1.08×	+0.014
τ^2 airline	terra	1.10×	+0.017
τ^2 airline	luna	1.17×	+0.014
τ^2 retail	sonnet-5	1.29×	+0.032
τ^2 retail	terra	1.31×	+0.023
τ^2 retail	haiku-4.5	1.35×	+0.019
τ^2 retail	luna	1.44×	+0.012