FROM FAILURES TO HARNESS: ADMISSION-GATED RUNTIME PATCH SELECTION FOR HARD HARNESS DEFECTS IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents rely on a runtime harness for tool calls, execution budgets, and recovery. Evaluating repairs to this layer requires distinguishing task success from recovery under an exercised defect. We introduce an injected-defect setting and an evaluation protocol that measures repair under recorded trigger exposure, same-family transfer, and clean-task regression. TARPS (Trigger-Aware Replay for Patch Selection) implements candidate search and admission using replay and reference-repair screening. Across seven frozen domain–defect cohorts on τ^2-bench, reward-only selection raises mean repair from 10% for immediate first-candidate selection to 57%; adding clean-task checks reaches 76%, matching full TARPS. TARPS attains 76% transfer versus 71% for reward plus clean checks; the incremental effect of its remaining checks is unresolved at this sample size. Separate admission analyses show that reference-mechanism inclusion, refusal, and transfer measure distinct properties, and coverage changes which failures enter evaluation. Together, these findings identify execution checking as the main source of the observed gain and clarify what admission and trigger-aware evaluation measure.

1 INTRODUCTION

Large language model (LLM) agents complete tasks through repeated model calls and tool interactions (Yao et al., 2023). Their runtime control layer—the *harness*—dispatches tools, assembles context, executes recovery logic, and manages execution budgets. A defect in this shared layer can cause repeated failures across tasks, making the harness itself a target for repair.

Consider a harness that both cuts the execution budget and returns incorrect tool results. Restoring the budget fixes only one component. Yet a test task may now succeed if it never triggers the result error, crediting an incomplete repair. More generally, a successful execution does not establish that a silent or compound defect has been repaired. Repair evaluation must therefore establish which defect was exercised, while patch selection must also check recovery and clean-task behavior.

Agent self-improvement methods turn experience into reflective memories (Shinn et al., 2023), reusable insights (Zhao et al., 2024), or induced workflows (Wang et al., 2025). We target executable repairs to tool dispatch, result handling, and budget control. Our central question is what execution evidence supports selecting such a repair when several patches plausibly address the same failure symptom.

We study this question through four injected defect families in the airline and retail domains of τ^2-bench (Barres et al., 2025): unavailable tools, substituted tool results, reduced execution budgets, and compound defects. Known mechanisms let us measure repair under trigger exposure, distinguish mechanism inclusion from task recovery, and compare selection rules on the same failed instances. TARPS (Trigger-Aware Replay for Patch Selection) provides a concrete implementation: it enumerates typed operators from required task actions, screens replay probes with reference repairs, and tests candidates on affected and clean tasks (Figure 1). Here, trigger awareness specifies the defect exposure used by the protocol; its incremental value at admission is an empirical question.

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.

The primary evaluation raises mean repair from 10% to 71% and same-family transfer from 14% to 76% over immediate first-candidate selection. A separate comparison on seven frozen cohorts examines how much simple execution feedback already achieves: reward-only selection reaches 57% repair, and adding clean-task checks reaches 76%, matching full TARPS (Section 6.2). Thus, much of the observed improvement is already obtained by simple checks; the incremental contribution of the full rule remains uncertain. Episode counts differ, and an exploratory replacement for reference-based screening improves recovery while using more executions. These results motivate treating information access and execution cost explicitly when comparing patch selectors.

Our contributions are threefold.

(1) A repair setting and exposure-aware evaluation protocol. Four defect families expose explicit, silent, compound, and ambiguous failures; the protocol measures repair under defect exposure and transfer to unseen same-family tasks (Sections 3–5).

(2) Empirical findings on execution-based selection. Frozen-cohort comparisons quantify the recovery achieved by simple execution checks. Admission and coverage analyses distinguish reference-mechanism inclusion, refusal, transfer, and failure confirmation, so that these outcomes can be assessed separately (Section 6).

(3) A reproducible implementation of checked selection. TARPS combines typed candidate enumeration, replay, and clean-task checks, returning a jointly tested patch set under known defect triggers and reference repairs (Section 4).

2 Related Work

Reasoning, experience reuse, and skill learning. Chain-of-thought prompting (Wei et al., 2022) and ReAct (Yao et al., 2023) improve reasoning and its interaction with actions. Reflexion (Shinn et al., 2023) and ExpeL (Zhao et al., 2024) reuse feedback as reflections or experience, while Agent Workflow Memory (Wang et al., 2025) extracts reusable workflows, including a variant that exposes them as callable actions. Voyager (Wang et al., 2023) builds a library of executable skills, and DSPy (Khattab et al., 2024) optimizes language-model pipelines against a specified metric. TARPS studies a more specific problem: admitting compensating patches for defects in the runtime control layer, with checks that distinguish task recovery from execution that fails to exercise the defect.

Agent infrastructure and executable repair. AutoGen (Wu et al., 2023), AgentVerse (Chen et al., 2024), and AgentScope (Gao et al., 2024) provide infrastructure for composing agents and coordinating their execution. Automated agent design (Hu et al., 2025) searches over agent implementations in code. Program synthesis and repair likewise produce executable changes, from generated solutions (Chen et al., 2021; Austin et al., 2021) to repository patches evaluated on software issues (Jimenez et al., 2024). Harness repair shares this executable repair surface. Our focus is the acceptance decision for a bounded set of runtime operators when repeated agent executions can yield different outcomes.

Trace-guided harness repair. HarnessFix (Chen et al., 2026) directly addresses harness flaws: it aligns failed trajectories with implementation artifacts, diagnoses responsible components, generates scoped patches, and validates them using improvement and regression checks. TARPS shares the use of executable patches and regression validation. We examine admission under controlled silent and multi-layer defects, using recorded trigger events, reference repairs to screen replay instances, and repeated execution to evaluate candidates. Known injected mechanisms also let us measure whether admitted patch sets contain the corresponding repair operators. The comparison therefore concerns the evidence used for admission and the evaluation protocol.

Tool-use benchmarks and controlled defects. τ^2-bench (Barres et al., 2025) evaluates agents interacting with users and shared state, while API-Bank (Li et al., 2023) and ToolLLM (Qin et al., 2023) evaluate tool-use capabilities. Task completion in these settings supplies useful outcome measures for repair evaluation. Our protocol adds controlled harness-defect injections and trigger logs, so that an unsuccessful task can be checked for defect exposure and recovered reward can be interpreted alongside coverage. Disjoint task pools measure same-family transfer after admission,

and clean-task probes check for regressions. The resulting comparisons concern the specified defect and operator families.

3 HARNESS DEFECTS AND REPAIR

Agents, harnesses, and patches. We study tool-using LLM agents that conduct task dialogues, call tools in an executable environment, and receive an episode reward $r \in [0, 1]$. The harness controls context, tool dispatch, verification, recovery, and execution budgets: $\mathcal{H} = (\pi_{\mathrm{ctx}}, \pi_{\mathrm{tool}}, \pi_{\mathrm{ver}}, \pi_{\mathrm{rec}}, \pi_{\mathrm{budget}})$. Repair changes this control layer while keeping model weights, task specifications, and tool implementations fixed. For task i, a defect instance d_i produces the defective harness $\mathcal{H}^{d_i}$; its parameters may depend on the task. A patch δ is a compensating operator that restores a binding or budget, or adds a guard, without disabling the defect injection. $\mathrm{Apply}(\mathcal{H}, \delta)$ denotes application of a patch to a harness. A method returns an ordered patch list P, which may be empty. We evaluate these repairs on tool-use benchmarks (Section 5).

Defect exposure and confirmed failures. An episode under $\mathcal{H}^{d_i}$ with patches P produces a transcript τ and an execution record ρ, containing its reward, observed events $\mathcal{E}(\rho)$, and a summary of tool calls, errors, and termination. Let e_t denote an attempt to invoke the affected tool t, and e_B denote reaching the reduced execution budget. The events required to exercise a defect are

$$E(d) \;=\; \begin{cases} \{e_t\} & \text{tool unavailability or result substitution on } t, \\ \{e_B\} & \text{budget reduction,} \\ E(d_1) \cup E(d_2) & \text{compound defect } d = (d_1, d_2). \end{cases} \tag{1}$$

An attempted call counts as exposure even when a repair routes it around the faulty dispatch logic. An episode is *covered* when $E(d) \subseteq \mathcal{E}(\rho)$, and is a *confirmed failure* when it is covered and $r(\rho) < \frac{1}{2}$. Appendix A specifies how these events are recorded.

Hard harness defects. We call a defect *hard* when the observed symptoms do not uniquely identify the affected harness component. Tool unavailability, which gives an explicit error, serves as a symptomatic comparison. The harder cases include result substitution, which silently returns another entity's record through a successful tool call, and budget reduction, which truncates execution with a cap-termination indicator. Multi-layer compound defects combine these two mechanisms. The reduced budget can end an episode before the affected tool call, leaving the swap unexercised and the compound defect uncovered. Symptoms can also have competing explanations: a tool error may indicate an outage, and an incomplete dialogue may reflect the agent's action choices. These cases motivate checking defect exposure separately from task reward.

Evaluation sets and repair metrics. A cell pairs a domain with a defect family. Within a cell, the *diagnosis set* $\mathcal{D}$ contains confirmed failures under the unpatched defective harness. The *transfer set* $\mathcal{T}$ is collected in the same way from a disjoint pool of tasks in the same family. Clean held-out probes $\mathcal{P}$ are screened for majority success under the unpatched clean harness and used to check regressions. An instance is *rescuable* when the reference repair $o(d_i)$ achieves a majority of covered passes over repeated episodes. This screen selects replay instances; all instances remain in $\mathcal{D}$ for scoring, including those that fail the screen. We report $\mathrm{Rep@D}$, the fraction of diagnosis instances with a strict majority of passing votes among covered episodes under P; $\mathrm{Tr@T}$ applies the same rule to the transfer set. Held indicates whether every clean probe retains majority success with P applied. Pool sizes and vote counts are specified in Section 5; admission is defined in Section 4.

Why reward recovery is insufficient. A task can succeed without reaching the affected tool or budget limit. Such an episode provides no evidence about repair of that mechanism. For a compound defect, a patch can also appear successful when execution exercises only the component it repairs. Reward-only validation cannot distinguish these outcomes from recovery under full defect exposure. Coverage-restricted scoring excludes uncovered episodes. TARPS also checks coverage during admission; the frozen-cohort comparison in Section 6.2 examines this choice separately from the common scoring rule.

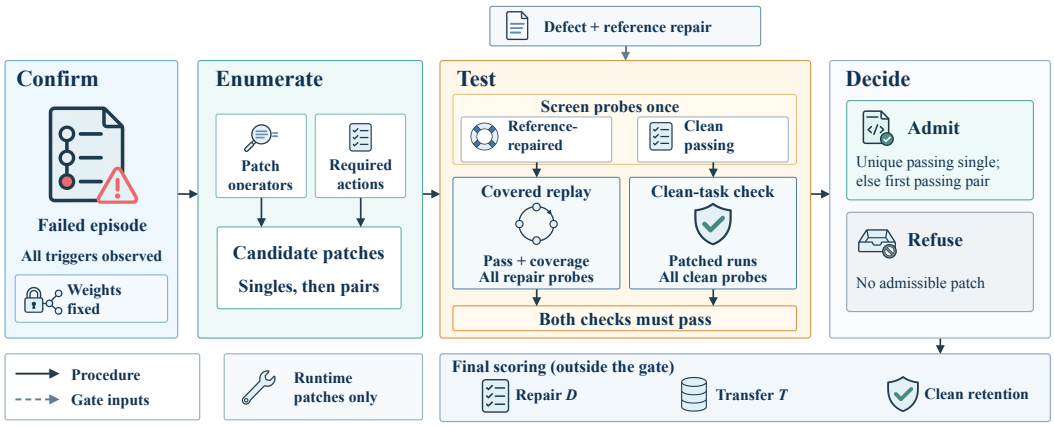


Figure 1: TARPS separates runtime patch generation from admission. Candidates come from required task actions. Reference-repair and clean-task screens select probes; coverage-aware replay and clean-task regression test single operators or candidate pairs (Algorithm 1). The returned patch list is scored on diagnosis, transfer, and clean held-out instances.

4 TARPS: Candidate Search and Patch Admission

Given a confirmed failure (i, ρ_i) and its task specification, TARPS enumerates single operators from required task actions and returns a passing singleton only when unique, else the first jointly passing pair or $\emptyset$; every returned set passes the gate as a composition of at most two operators (Figure 1; Algorithm 1). Only the gate accesses the injected defect specification and reference repair: candidate generation uses task actions, while admission assumes these controlled inputs. The selection comparisons in Section 6.2 vary the checks and information used at admission while retaining the evaluation protocol.

Algorithm 1 TARPS candidate selection: return the unique passing single operator, otherwise the first jointly passing pair.

Require: confirmed failure (i, ρ_i); candidates $\Delta(i)$; probes $\mathcal{D}_{\text{resc}}, \mathcal{P}$; defects $\{d_j\}$, votes v_g, v_h
Ensure: returned patch list P^* or $\emptyset$

if $\mathcal{D}_{\text{resc}} = \emptyset$ or $\mathcal{P} = \emptyset$ **then**
  return $\emptyset$ {empty probes; no candidate evaluation}
end if
$S \leftarrow [\,]$
for each operator $\delta \in \Delta(i)$ in order **do**
  evaluate $g_{\text{rep}}([\delta])$ and $g_{\text{held}}([\delta])$ using Eqs. 2–3
  if $g_{\text{rep}}([\delta]) \wedge g_{\text{held}}([\delta])$ **then**
    append δ to S
  end if
end for
if $|S| = 1$ **then**
  return S {unique passing singleton}
end if
for $P \in \binom{\Delta(i)_{1:6}}{2}$ in order **do**
  evaluate $g_{\text{rep}}(P)$ and $g_{\text{held}}(P)$
  if $g_{\text{rep}}(P) \wedge g_{\text{held}}(P)$ **then**
    return P
  end if
end for
return $\emptyset$ {no candidate admitted: $\mathcal{H}$ unchanged}

Candidate search. The patch DSL contains dispatch restoration, targeted and global result checks, and budget restoration (Table 3, Appendix A). These operators compensate for defects without disabling injection. For each required action, $\Delta(i)$ includes dispatch restoration for its tool and targeted checks for entity-identity arguments (user, reservation, order, and product identifiers), followed by a global check and budget restoration. Search tests all single operators in order. If exactly one passes, it is returned; if zero or more than one pass, the search falls through to pairs drawn from the first six candidates, evaluated as compositions under the same gate, stopping at the first passing pair. The six-candidate limit bounds the number of pair tests. Every returned set is therefore gate-tested jointly and contains at most two operators. Section 6.3 distinguishes the one nonconforming historical run from this selection rule.

Probe selection. Disjoint task pools supply confirmed failures $\mathcal{D}$ and $\mathcal{T}$, with defect parameters recorded for replay (Section 3). A diagnosis pool without a confirmed failure is unevaluable. Screening runs once per cell: $i \in \mathcal{D}$ enters $\mathcal{D}_{\text{resc}}$ if the reference repair achieves a majority of covered passes over v_s episodes; a clean task enters $\mathcal{P}$ if a majority of v_s unpatched episodes passes. This screen selects replay probes without removing instances from final scoring. If either probe set is empty, the gate returns $\emptyset$ without evaluating candidates.

Replay and coverage. Let $\text{Run}(i, d, P)$ denote an episode with defect d and patches P ($d = 0$ denotes a clean harness). For each $j \in \mathcal{D}_{\text{resc}}$, v_g episodes with rewards r_v and event sets $\mathcal{E}_v$ must satisfy

$$\text{rep}_j(P) \;:\quad \#\{v : r_v \geq \tfrac{1}{2}\} \;>\; \tfrac{v_g}{2} \quad\wedge\quad \#\{v : E(d_j) \subseteq \mathcal{E}_v\} \;\geq\; 1, \tag{2}$$

The gate requires $g_{\text{rep}}(P) = \bigwedge_j \text{rep}_j(P)$, stopping replay at the first failing instance; with the reported $v_g = 2$ this means both episodes pass and at least one is fully covered.

Clean-task checks and admission. Each $h \in \mathcal{P}$ is run for v_h episodes under the clean harness with P applied. The regression check is

$$g_{\text{held}}(P) \;=\; \bigwedge_{h \in \mathcal{P}} \mathbf{1}\Big[\#\{v : r_v \geq \tfrac{1}{2}\} > \tfrac{v_h}{2}\Big] \tag{3}$$

A tested candidate passes when $g_{\text{rep}} \wedge g_{\text{held}}$ holds; Eq. 3 uses strict majority. Within each run, $\mathcal{T}$ is used only for final scoring, not candidate selection.

Cost and output. One candidate uses at most $v_g|\mathcal{D}_{\text{resc}}| + v_h|\mathcal{P}|$ episodes; early termination reduces replay but the clean checks still run. Search makes at most $|\Delta| + \binom{6}{2}$ gate calls, plus $v_s(|\mathcal{D}| + |\mathcal{P}_{\text{pool}}|)$ screening episodes. Failure collection and final scoring are counted separately. The returned list, including an empty list after refusal, is scored on all of $\mathcal{D}$ and $\mathcal{T}$ under coverage-restricted majority (Section 3); vote budgets are specified in Section 5.

A candidate's gate pass establishes passage of these probes and thresholds, not of an untested composition; it requires observable triggers and reference repairs, and an empty probe set or failed reference screen does not prove all possible repairs fail. Table 8 (Appendix C) records the failures motivating the checks.

5 Experimental Design and Comparisons

Benchmark and defects. The primary matrix uses the airline and retail domains of τ^2-bench (Barres et al., 2025), where agents and simulated users interact with shared state. Evaluation uses task rewards, including required database changes and, for airline tasks, communication. Each domain–defect pair is a cell (Section 3); tool-specific defects fix the affected tool within the cell, so transfer uses new tasks with that tool. The four primary families are tool unavailability (removing the target tool from dispatch), result substitution (another entity's record), budget reduction (step cap 40 to 5), and compound (the budget cut plus result swap).

Comparators. All methods return patches in the same DSL. KEYWORD maps explicit error strings to operators by if–then rules. NOGATE applies the first candidate in TARPS's enumeration without testing it. HARNESSFIX generates candidates from error text in the trace digest and evaluates

them with TARPS's gate; it is our independent heuristic under this injected-defect protocol, not a reproduction of the external system (Chen et al., 2026). LLM-DIAG asks the agent model to propose DSL patches from the trace digest and applies them without admission. ORACLE applies the known repair for the injected mechanism and supplies the rescuability-screening reference; its measured success rate is not a deterministic upper bound. TARPS tests action-derived single operators and, unless exactly one passes, jointly gated pairs, returning at most two operators (Section 4).

Task sets and admission. In each primary cell, disjoint scan pools of ten and eight tasks supply $n_{\mathcal{D}} = n_{\mathcal{T}} = 3$ confirmed failures for diagnosis and same-family transfer. Three screening episodes per instance assess reference-repair recovery; qualifying diagnosis instances enter $\mathcal{D}_{\text{resc}}$. One clean probe is selected from four tasks by majority success over three clean episodes (Sections 3 and 4). The first confirmed diagnosis failure supplies the task and trace: TARPS uses required actions, while trace-based methods use its digest. Gate candidates must pass both replay episodes on every screened diagnosis instance, with at least one fully covered episode per instance, and majority success over three candidate-alone episodes on each clean probe. Empty probe sets cause refusal. Final scoring uses three episodes per instance and covered votes under the defect and returned patch list for $\mathrm{Rep@D}$ and $\mathrm{Tr@T}$. The primary agent and LLM-DIAG proposer use `deepseek-v4-flash`.

Comparison boundaries. Methods share scan-pool definitions and voting rules within a matrix, but stochastic collection can yield different confirmed tasks. Gate-using methods spend additional replay and clean-check episodes; we report episode counts because the NOGATE comparison tests checked search with these extra evaluations, not equal-cost selection. The diagnosis pilot uses $n_{\mathcal{D}} = n_{\mathcal{T}} = 2$ and different confirmed task sets from the primary matrix. The coverage-disabled arm also changes failure collection and scoring, so its task sets differ; it does not isolate admission-time coverage (Section 6.5).

Frozen-cohort selection comparison (v7). A separate study freezes diagnosis, transfer, and clean-probe membership within each of the seven evaluable cells before comparing selection rules. All conditions share task-derived candidates and final scoring; Section 6.2 compares reward, clean-task, coverage, and reference-screening variants. An oracle-free variant infers admission triggers from traces and replaces the reference screen; evaluation still uses known defect triggers. We report actual episode counts: post-selection top-ups impose a minimum expenditure without contributing to decisions, so this is not an equal-budget search comparison. Appendix G gives the condition definitions, paired analysis, and exploratory revision.

Extensions and reporting. Appendix E reports local QWEN3.6-35B-A3B-NVFP4 as a second agent model; τ-bench v1 (Yao et al., 2025) under both agent models; the τ^2-bench telecom domain; and three additional defect families. Agent and user-simulator configurations are listed per arm; all extensions retain the vote counts and scoring rules. An enlarged scan-pool arm re-collects six cells at $n_{\mathcal{D}} = n_{\mathcal{T}} = 5$. Reported patch sets are classified against known repair mechanisms, including additional KEYWORD, HARNESSFIX, and LLM-DIAG runs on τ-bench v1 (Appendix F, Table 13). Per cell, we report $\mathrm{Rep@D}$, $\mathrm{Tr@T}$, Held, replay-set size $n_{\text{rescuable}}$, and total episodes. Cells without confirmed failures are unevaluable rather than assigned zero repair (retail$\times$compound); gate refusals are scored with an empty patch list. Table 8 (Appendix C) links each admission check to the observed failure that motivated it.

6 REPAIR OUTCOMES AND SELECTION EVIDENCE

Figures 2 and 3 summarize repair and transfer outcomes; Table 2 reports admission quality. The primary matrix uses 1952 episodes, including construction of the unevaluable eighth cell. Exact aggregate and per-cell rates are in Appendix B (Tables 4 and 5).

6.1 CHECKED SEARCH RAISES REPAIR AND TRANSFER

TARPS reaches 71% mean repair and 76% transfer across seven cells, versus 10% and 14% for NoGate (Figure 2a). The paired cell-level gain is 62 points for each measure (95% cell-bootstrap interval $[19, 95]$; two-level interval $[19, 100]$). TARPS is higher in five cells, lower in one, and tied

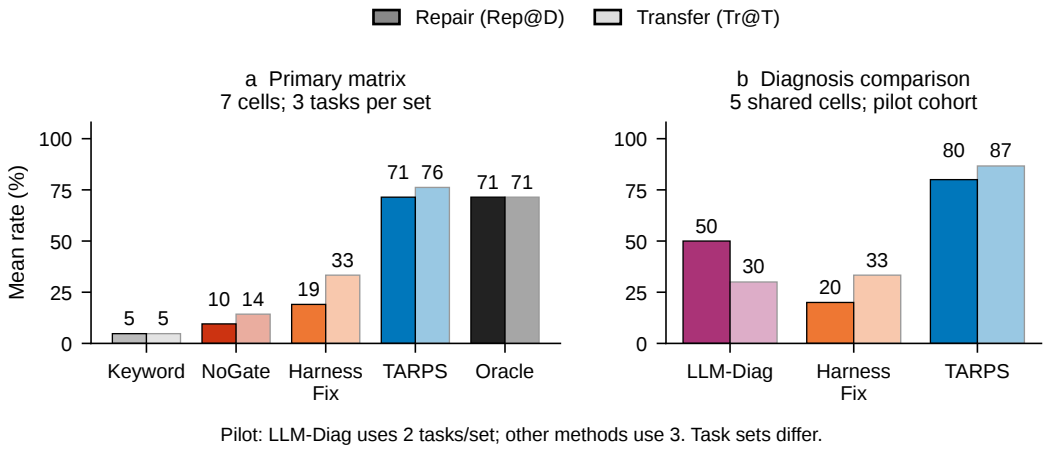


Figure 2: TARPS reaches 71% repair and 76% transfer in the primary matrix and 87% transfer in the diagnosis comparison. **(a)** Unweighted mean rates across seven evaluable domain–defect cells, with three instances per set; Oracle is the reference repair, not a performance upper bound. **(b)** Means over five shared cell names: LLM-Diag uses two instances per set (320-episode pilot), versus three for the other methods. Confirmed tasks differ, so this is not a paired or equal-budget comparison. HarnessFix denotes our internal heuristic, not the external system. Bar labels are rounded percentages; exact rates are in Table 4.

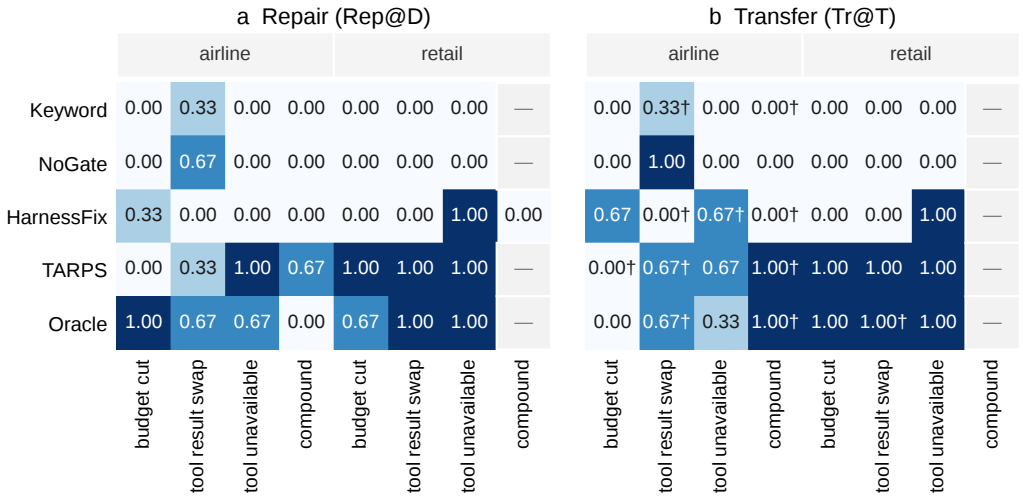


Figure 3: Aggregate gains span several defects, while TARPS refuses the airline budget repair. Primary-matrix repair (a) and transfer (b) rates use three instances per set. Grey dashes denote unevaluable retail compound; our internal HarnessFix heuristic alone has one diagnosis instance there (repair 0, no transfer), excluded from the seven-cell means. † A failed final clean held-out check, including checks on empty outputs; it does not by itself imply patch-induced harm. Exact rates: Table 5.

in one (exact cell sign-flip $p = 0.0625$; Appendix B). Its mean repair equals the reference repair's, although it refuses the airline budget cell and admits the reference combination for airline compound. The reference supplies the rescuability screen and is not an independent comparator. Retail compound is unevaluable because no confirmed failure exercises both triggers.

Table 1: Selection rules on seven frozen cohorts (v7), with three instances per diagnosis and transfer set. Rates are unweighted cell means. Episodes include post-selection top-ups and differ across conditions; these are not equal-budget search results. † Exploratory screen revision after the initial oracle-free results. Oracle is a reference repair, not an upper bound.

Selection rule	Repair (%)	Transfer (%)	Episodes
NoGate	9.5	14.3	210
Reward only	57.1	57.1	879
Reward + clean check	76.2	71.4	957
+ coverage (no reference screen)	71.4	61.9	957
Full TARPS	76.2	76.2	716
Oracle-free: first-candidate screen	23.8	23.8	716
Oracle-free: any-candidate screen†	61.9	52.4	1478
Oracle reference	90.5	85.7	210

Table 2: Reference-mechanism inclusion across reported runs, excluding v7. Correct means containing the reference repair mechanism, not successful transfer. † Historical merge-path violation; excluding it gives 18/18 conforming nonempty TARPS outputs (Appendix F).

Method	Nonempty	Correct	Incorrect	Precision
NoGate	35	9	26	25.7%
TARPS	19	18	1†	94.7%

6.2 What Simple Execution Checks Already Achieve

The primary comparison combines candidate testing with additional execution cost. We therefore compare selection rules on seven frozen cohorts (Table 1), keeping task membership, candidates, and scoring fixed. This separate study is not pooled with the primary matrix. Reward-only selection reaches 57% repair versus 10% for NoGate; adding clean-task checks reaches 76%, matching full TARPS. Thus, simple execution checks already recover much of the observed improvement.

Full TARPS reaches 76% transfer versus 71% for reward plus clean checks. Relative to reward-only selection, its paired gain is 19 percentage points for both repair and transfer, with two-level 95% bootstrap intervals $[-29, 62]$ and $[-29, 67]$, respectively. Against reward plus clean checks, the differences are 0 and 5 points, with intervals $[-38, 33]$ and $[-38, 43]$. The additional checks show no resolved improvement (Appendix G).

Replacing known admission triggers with trace-inferred events and screening with the first candidate yields 24% repair and transfer. After inspecting this outcome, we evaluated an any-candidate screen: it reaches 62% repair and 52% transfer at 2.06 times full TARPS's recorded episode count. This exploratory revision changes both screening and cost, so it suggests a direction for reducing reference information rather than establishing a budget–information exchange. All conditions retain known triggers for evaluation. Top-up episodes run after selection and scoring and cannot improve either decision; Appendix G documents this accounting.

6.3 Mechanism Inclusion, Refusal, and Transfer Differ

Across the 37 reported cell runs in the main and extended evaluations, nonempty TARPS outputs contain the reference mechanism in 18/19 cases, versus 9/35 for NoGate (Table 2). Excluding one historical merge-path violation, all 18 outputs conforming to the joint-gate rule contain the reference mechanism (Wilson 95% interval $[82\%, 100\%]$, versus $[14\%, 42\%]$ for NoGate). These rates describe selection precision among nonempty outputs, alongside 14 refusals despite reference-repair recovery evidence. The historical exception and corrected rerun are detailed in Appendix F.

On τ-bench v1 (Yao et al., 2025), TARPS with DeepSeek admits reference-containing repairs in $6/8$ cells, yet two of the three admitted airline cells have zero transfer. Local Qwen refuses $5/8$ cells (Appendix E). Thus, inclusion, admission, and transfer describe different properties of a repair system.

6.4 Ungated Model Patches in the Diagnosis Pilot

The five-cell diagnosis pilot compares proposal generation as well as admission, with two instances per set for LLM-Diag versus three for TARPS and different confirmed tasks. TARPS achieves 87% transfer versus 30% for LLM-Diag (Figure 2b). LLM-Diag selects the correct dispatch repair for both unavailable-tool cells, but its targeted result check fails to transfer on airline result substitution, and budget restoration alone repairs neither diagnosis nor transfer tasks for compound. These examples illustrate incomplete repairs; the unpaired, unequal-cost pilot does not isolate adding a gate to model-generated proposals.

6.5 Coverage Changes Failure Confirmation

The coverage-off rerun changes confirmed-failure membership in eleven cells. Nine reproduce their coverage-on outcome, and two change to refusal; no rejected patch is re-admitted. In the two divergent cells, no sampled instance passes reference screening, leaving an empty replay set. Across the 36-cell conforming cohort, nonempty precision is $16/16$ coverage-off versus $18/18$ coverage-on, while refusals with empty replay sets increase from 4 to 9 (Appendix F, Table 14). Coverage therefore changes which failures enter evaluation and how refusals are classified. Because this arm also changes collection and scoring, it does not isolate the admission-time coverage term; the frozen-cohort comparison keeps those evaluation choices fixed.

Earlier protocol failures. Appendix D and Tables 8–9 document the observed failures that motivated successive admission checks.

7 Implications and Limitations

Separate execution feedback from additional gate checks. The frozen-cohort comparison shows that reward-based selection already improves substantially over committing the first candidate; adding clean-task checks matches full TARPS's mean repair. This makes simple execution checks an essential comparator for runtime repair methods. The present intervals leave the extra value of coverage and reference screening unresolved. Their utility should be evaluated jointly with refusal and transfer, since precise mechanism inclusion can coexist with frequent refusals or poor task recovery.

Construct informative replay probes. Trigger coverage identifies which failures are defect-exposed, while reference screening identifies probes on which the known repair can recover reward. The exploratory any-candidate screen recovers part of the performance lost by the initial oracle-free variant, at higher execution cost. Because both the screening rule and cost change, this result motivates a controlled information–budget study rather than establishing a substitution rate between them.

The broader evaluation spans two agent models and injected defects, with repairs in the specified operator space and new tasks sharing the affected tool. The primary and frozen-cohort studies use $n_{\mathcal{D}} = n_{\mathcal{T}} = 3$ per cell, with finite-vote checks; this sample and design do not resolve individual component contributions. Pairing controls task membership, but model executions remain stochastic, costs differ, and transfer pools were reused during protocol development. The reported intervals describe recorded task outcomes, not repeated full selector runs. The revised oracle-free screen is exploratory. Future work should use fresh confirmation pools and fixed decision budgets, then extend the protocol to natural defects with independently observable exposure criteria and replay probes.

8 Conclusion

Runtime harness repair requires evidence about defect exposure, recovery, and clean-task behavior. Our injected-defect protocol and TARPS implementation make these quantities explicit. Frozen-cohort comparisons show that simple execution checks recover much of the gain over immediate candidate selection, making them essential comparators for richer admission rules. Mechanism inclusion, refusal, and transfer reveal different aspects of repair quality. Measuring them separately provides a basis for comparing repair systems and studying informative replay beyond known injected mechanisms.

AI USE STATEMENT

We used generative AI tools to assist with method and experiment-code implementation, analysis of experiment records, and manuscript revision. We reviewed the AI-assisted work and take responsibility for the final content, including text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

All defects, repair operators, the admission gate, and the evaluation protocol are described in Sections 4 and 5 and implemented in the anonymous supplementary code. Per-cell admission metadata (rescuable-probe counts, committed patch-set sizes, episodes spent) and the run identifiers behind every reported table are given in Appendix B; the versioned chain that produced the admission checks is recorded in Appendix C; per-episode records are included in the supplementary material.

The frozen-cohort comparison is recorded separately under `tau2_v7_ident`. Its frozen task sets, per-instance outcomes, analysis script, and condition-level episode counts accompany Appendix G; both oracle-free screen variants are retained. Post-selection top-ups and the exploratory screen revision are documented separately from the primary matrix.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable LLM agents: Diagnosing and repairing harness flaws, 2026. URL https://arxiv.org/abs/2606.06324. preprint.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, 2024.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Zeyu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into

state-of-the-art pipelines. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155.

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*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024.

A Appendix: Defect and repair operator details

Typed patch DSL. The prose names dispatch restoration, targeted result check, global result check, and budget restoration correspond respectively to `restore_dispatch`, `result_guard`, `integrity_guard`, and `restore_budget`. Table 3 defines these four families; the DSL also admits `restore_policy` and list composition—the oracle for `compound` concatenates the component oracle repairs. All operators are compensating: applied on top of the defective harness, they restore bindings and budgets or add guards and never toggle the defect off.

Table 3: Patch operator families: parameters, semantics, and the defect mechanisms each addresses. "Oracle for" marks the defect family whose oracle repair $o(d)$ emits that operator.

Family	Params	Semantics and addresses	Oracle for
Dispatch restoration	tool t	rebinds calls to t to the pristine dispatch captured before the first defect wrap, bypassing whatever the defect installed; idempotent when no wrap exists	`tool_unavailable`, `tool_result_swap`
Targeted result check	tool t, field a	raises when the returned entity field a differs from the requested argument—a mitigation, not a repair: converts silent wrong-entity results (`tool_result_swap`) into visible errors but cannot correct the data	—
Global result check	—	the same entity-consistency check applied to *every* assistant call; payload binds no instance, so one patch guards any corrupted entity lookup	—
Budget restoration	budget b	resets the orchestrator's step budget to b; repairs episodes truncated by a reduced cap	`budget_cut`

Table 4: Mean repair and transfer rates in two comparison cohorts. **(a)** Primary matrix, with three instances per set. **(b)** Diagnosis comparison on five shared domain–defect cells, with two instances per set for LLM-Diag and three otherwise. Task instances differ across methods; the two blocks are not a paired comparison.

	(a) Primary matrix (7 cells)		(b) Diagnosis comparison (5 cells)	
Method	Repair	Transfer	Repair	Transfer
Keyword	0.05	0.05	—	—
NoGate	0.10	0.14	—	—
HarnessFix	0.19	0.33	0.20	0.33
LLM-Diag	—	—	0.50	0.30
Oracle	0.71	0.71	—	—
TARPS	**0.71**	**0.76**	**0.80**	**0.87**

Required trigger events. In Eq. 1, e_t corresponds to the logged event `call:`t, and e_B to `budget_cut:cap_hit`. The observed event set $\mathcal{E}(\rho)$ is stored in the record's `ev` field; its trace summary records `calls`, `errors`, and `terminated_at_cap`. Each defect family defines the required trigger events $E(d)$ an episode must contain to count as covered (Eq. 1): `tool_unavailable` and `tool_result_swap` on tool t require `call:`t—the assistant attempted the defected call, logged at the outermost dispatch layer so it registers even when a repair bypasses the defect's code path; `budget_cut` requires `budget_cut:cap_hit`—the episode reached the defect's step cap, by terminating at it or exhausting the capped assistant-step count; `compound` $d = (d_1, d_2)$ requires the union $E(d_1) \cup E(d_2)$: *every* component's triggers must fire, so an episode where one component is masked upstream covers nothing.

B PER-CELL REPORTS

Table 4 records the exact aggregate rates shown in Figure 2. Table 5 gives the primary per-cell rates visualized in Figure 3; Figure 4 also includes the separate LLM-Diag pilot. The primary matrix's 1,952 episodes sum `episodes_spent` across five reported methods in the four v6 airline cells (1,037) and four clean v6 retail cells (915), including the unevaluable cell; the packaged `reports.json` files give the per-cell counts.

Table 6 lists per-run admission metadata from the `reports.json` of each cell run (v6 gate matrix and `llm_diag_pilot` artifacts) for every cell whose run exercised a non-trivial admission decision: the rescuable replay-set size n_{resc}, the committed patch-set size n_{rep}, and total episodes N_{eps}

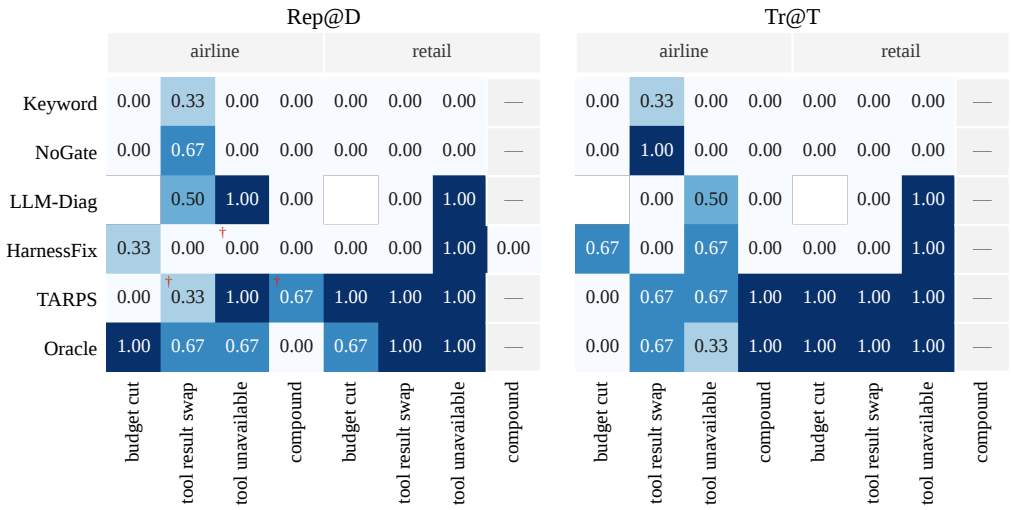


Figure 4: Full per-cell Rep@D and Tr@T matrix, including the LLM-Diag pilot (two instances per set, versus three otherwise; confirmed tasks differ). Exact primary rates are in Table 5, and pilot rates in Table 7. Daggers mark admitted patches that fail clean held-out probes; grey dashes mark the cell with no evaluable construction under coverage; blanks mark combinations not run.

Table 5: Per-cell Rep@D/Tr@T on τ^2-bench ($n_{\mathcal{D}}$=$n_{\mathcal{T}}$=3, coverage scoring), methods ordered by admission quality; the visual summary is Figure 3. TARPS matches the oracle Rep@D mean over evaluable cells and slightly exceeds its Tr@T mean; sub-1.0 cells reflect instance-level noise that the held-out probes also exhibit (the oracle itself fails Held on three cells). †final held-out validation failed under the committed patch set (Held=F); cells with n_{rep}=0 reflect clean-probe noise rather than a patch-induced regression. ‡unevaluable under coverage: the 5-step cap ends episodes before the swapped call is attempted, so no confirmed failure exists for the gate-scored methods; only HARNESSFIX surfaces a single confirmed diagnosis instance, which it fails to repair and for which no transfer is measured—reported as —, not 0.

	KEYWORD		NOGATE		HARNESSFIX		TARPS		ORACLE	
Cell	Rep@D	Tr@T	Rep@D	Tr@T	Rep@D	Tr@T	Rep@D	Tr@T	Rep@D	Tr@T
airline										
`budget_cut`	0	0	0	0	0.33	0.67	0	0†	1.0	0.0
`tool_result_swap`	0.33	0.33†	0.67	1.0	0	0†	0.33	0.67†	0.67	0.67†
`tool_unavailable`	0	0	0	0	0	0.67†	1.0	0.67	0.67	0.33
`compound`	0	0†	0	0	0	0†	0.67	1.0†	0	1.0†
retail										
`budget_cut`	0	0	0	0	0	0	1.0	1.0	0.67	1.0
`tool_result_swap`	0	0	0	0	0	0	1.0	1.0	1.0	1.0†
`tool_unavailable`	0	0	0	0	1.0	1.0	1.0	1.0	1.0	1.0
`compound`‡	—	—	—	—	0.0	—	—	—	—	—
Mean (7 evaluable cells)	0.05	0.05	0.10	0.14	0.19	0.33	**0.71**	**0.76**	0.71	0.71

(screening, admission, and scoring). Dashes mark `retail compound`, which yields no confirmed diagnosis-pool failure under coverage and is therefore unevaluable; repair and transfer rates for the same cells are in Tables 5 and 7.

Task-set differences in the pilots. For airline result substitution, LLM-Diag uses diagnosis tasks $\{6, 7\}$ and transfer tasks $\{32, 33\}$, while primary TARPS uses $\{3, 6, 7\}$ and $\{32, 33, 36\}$. For airline

Table 6: Per-cell admission metadata: rescuable replay-set size, committed patch-set size, and episodes spent (screening, admission, scoring) for every cell with a non-trivial admission decision.

Cell	n_{resc}	n_{rep}	N_{eps}
TARPS *(admission gate)*			
`airline budget_cut`	3	0	61
`airline tool_result_swap`	2	2	79
`airline tool_unavailable`	3	2	85
`airline compound`	2	2	103
`retail budget_cut`	3	1	98
`retail tool_result_swap`	3	1	100
`retail tool_unavailable`	3	2	103
`retail compound`	—	0	21
HARNESSFIX *(same gate)*			
`airline budget_cut`	3	1	51
`airline tool_result_swap`	1	0	44
`airline tool_unavailable`	3	1	48
`airline compound`	2	0	51
`retail budget_cut`	3	0	46
`retail tool_result_swap`	2	0	41
`retail tool_unavailable`	3	1	48
`retail compound`	1	0	35
LLM-DIAG *(gate bypassed, pilot)*			
`airline tool_result_swap`	1	1	35
`airline tool_unavailable`	2	1	28
`airline compound`	2	1	32
`retail tool_result_swap`	2	1	30
`retail tool_unavailable`	2	1	28
`retail compound`	—	0	21
TARPS, *coverage off (`ignore_coverage`, pilot)*			
`airline compound`	2	1	67
`airline tool_result_swap`	2	2	79

Table 7: Per-cell Rep@D/Tr@T for diagnosis without admission and the coverage-disabled pilot. **(a)** LLM-DIAG (320 episodes, two instances per set) and main-matrix methods (three per set) use five matching domain–defect cells but different confirmed tasks. **(b)** `ignore_coverage` changes failure collection, admission, and scoring; the two-instance pilot returns an incomplete budget-only patch on `airline compound` (1.0/0.0). This earlier pilot predates the matched-protocol coverage-off rerun (Section 6.5), which supersedes it as the ablation estimate: under matched budgets the same cell refuses rather than re-admitting the partial patch.

Cell	Patch proposed / admitted	Rep@D	Tr@T
(a) LLM-DIAG*: model diagnosis, no admission*			
`airline tool_unavailable`	`restore_dispatch(get_reservation_details)`	1.0	0.5
`airline tool_result_swap`	`result_guard(get_reservation_details)`	0.5	0.0
`airline compound`	`restore_budget` only	0.0	0.0
`retail tool_unavailable`	`restore_dispatch(get_order_details)`	1.0	1.0
`retail tool_result_swap`	`result_guard(get_order_details)`	0.0	0.0
mean over the five cells		0.50	0.30
TARPS (same cells)		0.80	0.87
HARNESSFIX (same cells)		0.20	0.33
(b) TARPS *without the coverage term*			
`airline compound`	`restore_budget` alone re-admitted	1.0	0.0
`airline tool_result_swap`	`restore_dispatch` pair	1.0	1.0

compound, the coverage-disabled pilot uses $\{1, 2\}$ and $\{32, 33\}$, while coverage-enabled TARPS uses $\{3, 6, 7\}$ and $\{33, 36, 40\}$. Thus matching cell names do not imply paired task samples.

Table 8: Admission-protocol design choices and the observed failure each blocks; each was added only after that failure was measured in an earlier protocol version (versioned chain and artifacts: Table 9; quantitative evidence: Section 6). Reported runs use two replay votes per instance, so per-instance majority requires both to pass; the residual single-draw brittleness at that budget is the probe-noise limitation of Section 7.

Design choice	Failure it blocks
Cross-instance replay	a partial patch passes where only part of a `compound` defect fired
Coverage-aware admission	reward recovery certified although the defect mechanism never fired
Per-instance majority replay	unanimity lets one stochastic fail reject a correct patch
Majority held-out regression	a single clean-task flake vetoes a correct patch
Rescuability screening	an instance no patch can save vetoes every candidate
Empty-probe short-circuit	empty probe sets admit vacuously and waste episodes

Table 9: Protocol failure→fix chain. The right column names the run in which the motivating failure was measured; the mechanism at left was added in response and shipped in the next protocol version.

Mechanism added	Measured failure	Run (episodes)
Cross-instance replay (v1→v2)	`compound` partial-accept: a budget-only patch passes replay on the diagnosed instance where only the budget half fired (Rep@D 1.0, Tr@T 0.0)	`transfer_matrix` (365)
Coverage-aware admission (v1→v2)	reward recovery is certified although the defect mechanism never fired; the trigger monitor sits at the outermost layer because a correct `restore_dispatch` bypasses the defect wrapper	`transfer_matrix` (365); verified in `compound_cov` (122)
Per-instance majority replay (v2→v3)	unanimity false-rejects a correct patch on one stochastic draw among covered votes (retail `tool_unavailable`, HARNESSFIX); 97 episodes spent admitting nothing (airline `tool_result_swap`)	`transfer_matrix_v2` (1072)
Majority held-out regression (v3→v4)	a single-shot clean-task flake vetoes a correct patch (retail `budget_cut`: $6/6$ covered replay votes, one held-out fail)	`transfer_matrix_v3`
Rescuability screening (v4→v5)	an unrescuable $\mathcal{D}$ instance vetoes every candidate (airline task 7: the oracle itself scores 0 on it)	`transfer_matrix_v3`
Empty-probe short-circuit (v4→v5)	empty probe sets admit vacuously and waste episodes	`transfer_matrix_v5` (907)

Uncertainty of the primary comparison. Rep@D and Tr@T are fractions of three instances, so each cell yields instance-level pass indicators. For the seven evaluable primary cells we compute the paired difference TARPS−NoGate per cell and average it (62 points for both measures). A cell bootstrap ($B = 20{,}000$) gives the 95% interval $[19, 95]$; a two-level bootstrap that resamples cells and then instances within each cell gives $[19, 100]$. Per-cell repair differences are $+100, -33, 0, +67, +100, +100, +100$ points (airline unavailable, swap, budget, compound; retail unavailable, swap, budget), and the exact sign-flip test over cells gives $p = 0.0625$. Mechanism inclusion uses Wilson intervals: $18/18$ gives $[82.4\%, 100\%]$ and $9/35$ gives $[14.2\%, 42.1\%]$. The two counts come from different run cohorts, so the Fisher exact $p < 10^{-6}$ is only a descriptive contrast. The script and its output are in the supplementary material (`artifacts/intervals/`).

C PROTOCOL FAILURE–FIX CHAIN

Table 8 summarizes the six admission mechanisms and the observed failure each blocks. Each mechanism entered the protocol only after that failure was measured in an earlier protocol version; Table 9 gives the versioned chain, and all artifacts are in the released code bundle under `artifacts/experiments/` (the `tau2_` prefix is omitted).

The v5 run (`transfer_matrix_v5`, 907 episodes) is historical development evidence. The current headline matrix uses v6 airline and the clean v6 retail rerun. Transfer tasks were reused

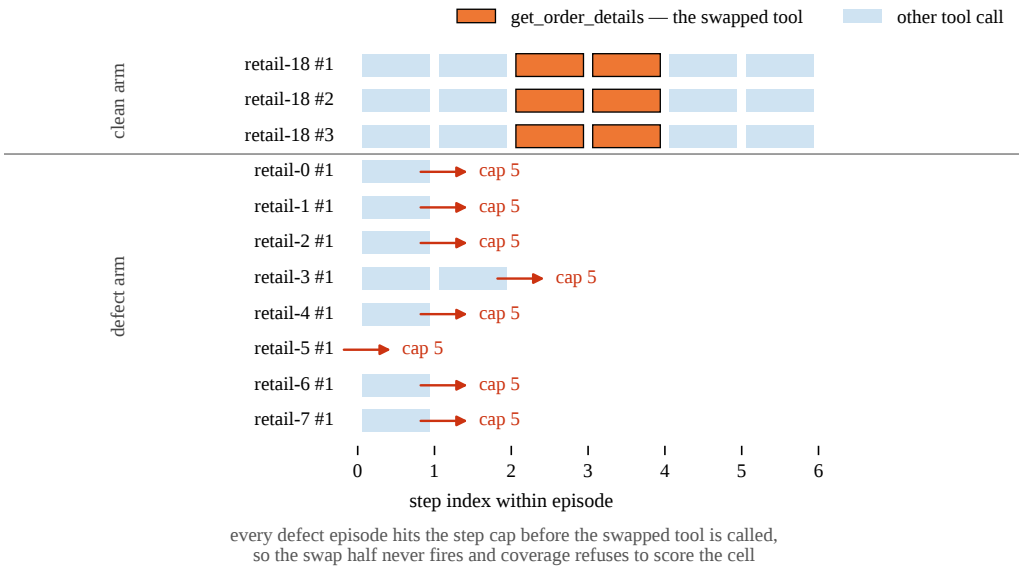


Figure 5: Coverage refuses an unexercised mechanism. Clean episodes reach the swapped tool within the first few steps; every episode under the compound defect instead hits the step cap before that tool is called, so the swap half never fires and the protocol declines to score the cell rather than attributing reward recovery to an unexercised mechanism.

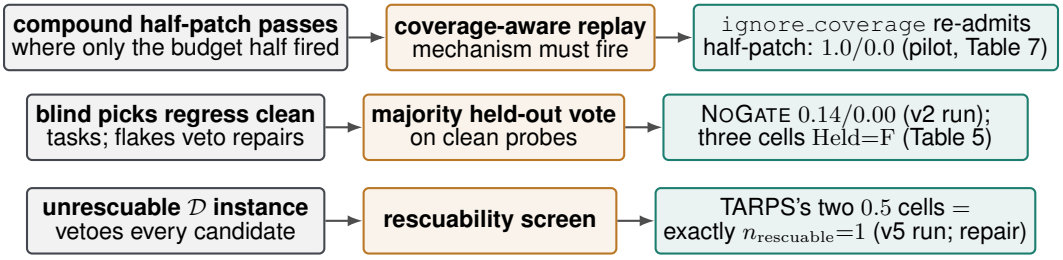


Figure 6: Visual summary of the same failure→fix chain tabulated in Table 8 (versions and runs: Table 9): each admission-gate check (center) was added after the measured failure at left; the right column gives observations from the named runs. The coverage-disabled pilot also changes task collection and scoring, so it is not an isolated intervention on the gate's coverage condition.

across development versions; within-run separation from candidate selection does not establish an untouched confirmation set after protocol design.

Figure 5 shows the coverage refusal that motivated the outermost-layer trigger monitor: in the `retail compound` cell every defect episode hits the step cap before the swapped tool is called, so the swap half of the compound defect never fires and no confirmed failure exists.

D Admission-gate evidence map

Figure 6 summarizes the admission checks. Figure 7 shows the clean-task check refusing a candidate that passed every replay vote; Figure 8 compares successful unpatched and failed patched episodes on clean task airline-18. Its conditional means are 24.2 and 28.5 calls; across all six unpatched and three patched episodes the means are 24.17 and 27.0. The selected failures contain more repeated arithmetic. This descriptive comparison does not isolate a patch-induced regression from execution variability.

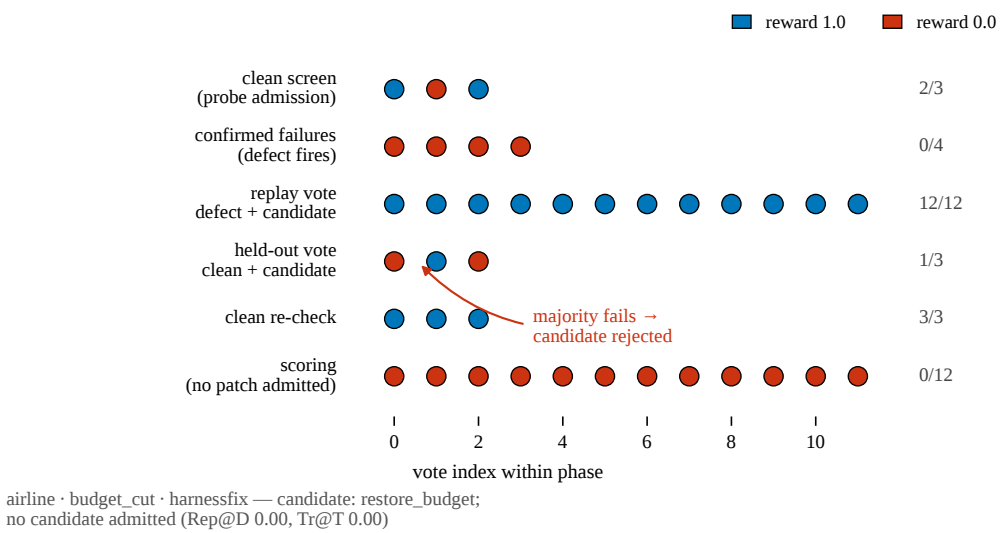


Figure 7: Why admission rejects a candidate that repairs every confirmed failure: in the airline budget-cut cell the candidate passes all replay votes under the defect but fails the majority of clean held-out votes, so nothing is committed and the cell is then scored with no patch admitted. Drawn from the per-episode records of that run.

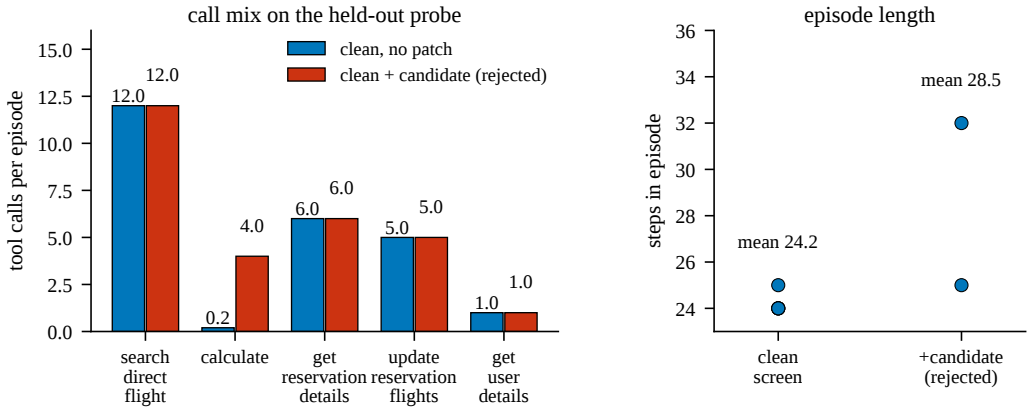


Figure 8: Outcome-conditioned trajectories on a clean probe. The plot selects five successful unpatched episodes and two failed patched episodes on airline-18; one failed unpatched and one successful patched episode are excluded. Values are means within these selected groups. Repeated arithmetic characterizes the selected failures, without establishing that the patch caused the difference.

E SECOND-MODEL AND SECOND-BENCHMARK TRANSFER PILOTS

Two pilots probe whether admission is a property of the mechanism or of the primary setting. Both use the same protocol (coverage, rescuability screening, per-instance majority replay, clean-probe regression, $n_{\mathcal{D}}{=}n_{\mathcal{T}}{=}3$); the agent model is a local QWEN3.6-35B-A3B-NVFP4 served by vLLM with thinking disabled.

Table 10: Transfer pilots. Rates are $\mathrm{Rep@D/Tr@T}$. **(a)** Second model on τ^2-bench (6 cells, 521 episodes; fully local, agent and user simulator both the local Qwen). **(b)** Second benchmark τ-bench v1, third-party harness (16 cells, 1281 episodes; fully local). "refused" = the gate admitted no candidate; † marks the cell where even the oracle repair scores 0, so no replay set exists and refusal is structural.

Cell	TARPS	NoGate	Note
(a) τ^2-bench, second model			
`retail tool_unavailable`	0.67/0.67	0.0/0.0	repair returned
`airline tool_result_swap`	0.0/0.0	0.67/0.33	refused; NoGate first-candidate luck
`airline compound`	0.0/0.0	0.0/0.0	refused
(b) τ-bench v1, third-party harness			
`retail tool_result_swap`	1.0/1.0	0.0/0.33	admitted repair transfers
`retail tool_unavailable`	0.67/0.67	0.0/0.67	admitted repair transfers
`retail budget_cut`	0.0/0.0	0.0/0.0	refused
`retail compound`	0.0/0.0	0.0/0.0	refused
`airline tool_result_swap`	0.0/0.0	0.33/1.0	refused†; NoGate luck
`airline tool_unavailable`	0.33/0.0	0.33/0.67	admitted, transfer failed
`airline budget_cut`	0.0/0.0	0.0/0.0	refused
`airline compound`	0.0/0.0	0.0/0.0	refused

Table 11: Extension matrices (local Qwen, fully local user simulator). Rates are $\mathrm{Rep@D/Tr@T}$. **(a)** τ-bench v1 new defect families (12 cells, 860 episodes). **(b)** τ^2 `telecom` domain (8 cells, 382 episodes; ‡ non-evaluable: no confirmed defect failure).

Cell	TARPS	NoGate	Note
(a) τ-bench v1, new families			
`retail tool_output_corrupt`	0.67/1.0	0.33/0.33	admitted repair transfers
`retail policy_truncate`	0.33/1.0	1.0/0.67	TARPS's one false admission
`retail schema_param_rename`	0.0/0.0	0.33/0.33	refused
`airline tool_output_corrupt`	0.0/0.33	0.33/0.0	admitted, repair failed
`airline policy_truncate`	0.33/0.33	0.0/0.0	refused
`airline schema_param_rename`	0.0/0.0	0.33/0.67	refused; NoGate luck
(b) τ^2 telecom, third domain			
`telecom tool_unavailable`	0.0/0.0	0.33/0.0	refused
`telecom tool_result_swap`	0.0/0.0	0.0/0.33	refused
`telecom budget_cut`	0.0/0.0	0.0/0.0	refused
`telecom compound`	—/—	—/—	non-evaluable‡

Pilot A: second model on τ^2-bench. Six cells of the main matrix re-run with the local model as agent (fully local: agent and user simulator both the local model; 521 episodes): `airline tool_result_swap`, `airline compound`, and `retail tool_unavailable`, each under TARPS and NoGate.

Pilot B: second benchmark, third-party harness. τ-bench v1 (Sierra's official harness, published at ICLR 2025) is a genuinely different stack: dispatch is `Env.step` → `tools_map[name].invoke`, the step budget lives on the agent loop, and reward is a deterministic database-hash check with no LLM judge. A small adapter exposes the same defect contract (single dispatch point, step budget, task set, reward), so the defect and repair operators run unchanged; the full matrix (2 domains × 4 families × {TARPS, NoGate}) ran fully locally (agent and user simulator both the local model), 1281 episodes.

Pilot C: wider defect families and a third domain. Two extension matrices probe breadth on the same local model (Table 11). (i) τ-bench v1 gains three new families — `tool_output_corrupt`, `schema_param_rename`, `policy_truncate` — 2 domains × 3 families × {TARPS, NoGate}, 860 episodes. (ii) τ^2 gains a third domain, `telecom` (114 tasks), × the four original families × both methods, 382 episodes; the `compound` cell is non-evaluable because the injected defect never produced a confirmed failure in the scan pool, and is reported rather than dropped.

Table 12: τ-bench v1 under the strong model (8 cells, 1219 episodes; agent = deepseek-v4-flash, user simulator = local Qwen). Pilot B uses the same harness, user simulator, and protocol settings; changing the agent can also change the collected failures and probes.

Cell	TARPS Rep@D/Tr@T	NoGate Rep@D/Tr@T	note
retail tool_result_swap	1.0/1.0	0.0/0.0	true admission
retail tool_unavailable	1.0/1.0	0.0/0.67	true admission
retail budget_cut	1.0/1.0	0.0/0.0	true admission
retail compound	0.0/0.0	0.0/0.0	refused (cons. miss)
airline budget_cut	0.67/0.33	0.0/0.0	true admission
airline tool_result_swap	0.67/0.0	0.67/0.0	true admission
airline tool_unavailable	0.67/0.0	0.67/0.0	true admission
airline compound	0.0/0.0	0.0/0.0	refused (cons. miss)

On the local model these extensions are dominated by refusals and near-zero rates. They add one nonempty TARPS output missing the reference mechanism (retail policy_truncate), versus five such outputs from NoGate (Table 13). In a densified $n{=}5$ rerun, TARPS's decisions remain unchanged in five of six cells; one changes from a nonempty output to refusal. This checks task-count sensitivity, not larger vote budgets or independent confirmation after protocol design.

Pilot D: strong model on the published benchmark. The same τ-bench v1 matrix re-run with deepseek-v4-flash as agent (local-model user simulator; 1219 episodes) is the mirror image of Pilot B: the gate admits in six of eight cells, *every* admission contains the oracle mechanism, and TARPS reaches $1.0/1.0$ on the three non-compound retail families — the admit→repair→transfer pattern of the main matrix, reproduced on a third-party harness.

Reading. Across Pilots A, B and D, every nonempty TARPS output contains the reference mechanism, but several have zero transfer repair. Across the 37 cell runs in Table 13, one of the 19 nonempty outputs lacks that mechanism. These are observed set inclusion rates, not guarantees of task repair, composition safety, or unchanged performance across settings. Qwen produces more refusals than DeepSeek in the reported τ-bench comparison. Model changes also affect failure and probe collection, so these runs do not isolate an effect of agent competence. NoGate's successes depend on candidate order: when the pinned tool is the first required action, its first candidate matches the reference repair. Two of the 1281 local τ-bench episodes ended in ContextWindowExceededError; they are counted as failures.

F Admission calibration against defect ground truth

For each injected defect, we compare the returned operator set with the reference repair set. A nonempty output is labeled *true* iff it contains the reference set, and *false* otherwise. Additional operators are counted as *extra*; their inclusion does not establish safety or joint gate passage. A refusal is labeled *principled* when no sampled instance passes the reference screen, and *conservative* otherwise. These are operational labels for the sampled runs, not claims about all possible repairs.

Table 13 uses the listed runs only. The main airline row includes only airline records from tau2_transfer_matrix_v6; its void retail block is excluded. Retail uses the clean tau2_transfer_matrix_v6_retail rerun. Later cross_proposer_tau2, outofdsl_refusal, and tau2_v7_ident runs are outside this table. The resulting 37 cell runs span repeated domain–defect settings under different models and benchmarks; they are not 37 independent defect families.

The inclusion rates are $18/19$ for TARPS and $9/35$ for NoGate, with 14 conservative and 4 principled TARPS refusals. On retail policy_truncate, three targeted guards and a budget restoration passed individually and were combined. The four-operator list first appears in final clean-probe and $\mathcal{D}/\mathcal{T}$ scoring, without joint replay admission. This result cannot be interpreted as a jointly tested composition passing the gate; it exposes the singleton-composition limitation of that earlier implementation, which Algorithm 1 avoids by joint testing.

Table 13: Reference-mechanism inclusion in the listed valid runs. TARPS returns nonempty sets in 19 of 37 cell runs; 18 contain the reference set. The policy-truncation output combines four individually passing operators without a joint gate check. NoGate returns nonempty sets in 35 runs; 9 contain the reference set.

Run	method	cells	true	false	cons. miss	principled
τ^2 main (airline)	TARPS	4	3	0	1	0
	NoGate	4	1	3	0	0
	oracle	4	4	0	0	0
τ^2 main (retail)	TARPS	4	3	0	0	1
	NoGate	4	0	3	0	1
	oracle	4	3	0	0	1
τ^2, second model	TARPS	3	1	0	2	0
	NoGate	3	0	3	0	0
τ1, third-party harness	TARPS	8	3	0	4	1
	NoGate	8	2	6	0	0
τ1, new families	TARPS	6	2	1	2	1
	NoGate	6	2	4	0	0
τ^2 telecom	TARPS	4	0	0	3	1
	NoGate	4	2	1	0	1
τ1 baselines	HarnessFix	8	0	0	8	0
	Keyword	8	2	0	5	1
	LLM-Diag	8	0	0	6	2
τ1, strong model	TARPS	8	6	0	2	0
	NoGate	8	2	6	0	0
Total	TARPS	37	18	**1**	14	4
	NoGate	37	9	**26**	0	2

Table 14: Coverage-off summary for the eleven cells with changed confirmed task membership. Identical means the same admission decision and Rep@D/Tr@T. The two divergent coverage-on outputs already had failed clean held-out checks and zero diagnosis repair.

Outcome	Cells	Direction
Identical outcome	9	2 admit / 7 refuse
Divergent	2	both $\rightarrow$ refuse
Nonempty precision (36-cell cohort)	16/16 off vs. 18/18 on	

Quarantine and joint-gate correction. A run-level audit of all reported cells found that multiple passing singletons occurred only here; every other cell admits at most one, so the earlier merge semantics and the corrected joint-gate semantics coincide on the remaining cells. We preserve the `policy_truncate` run for provenance, exclude it from the conforming cohort, and reran the cell under Algorithm 1 in both coverage arms: both reruns refuse (no jointly admissible subset), so the conforming coverage-on cohort is $18/18$ over 36 cell runs.

Coverage-off arm. Under `ignore_coverage`, confirmed-failure membership changed in 11 of the 36 cells; those cells were rerun with identical budgets and the remainder reuse the reported runs. The coverage-off cohort records 16 true admissions, 0 false, 11 conservative misses, and 9 principled refusals: precision is unchanged ($16/16$), while refusals shift toward the principled class because unrescuable instances enter the diagnosis pool and empty the replay set (Section 6.5). Table 15 lists the reruns; two timeout-contaminated telecom attempts were quarantined and rerun with a raised episode cap, and three episodes in the telecom cells hit the context-window limit during confirmation probes — none changes a refusal outcome.

Screen-off check. As a consistency check we reran the fourteen local cells in which the rescuability screen removed at least one instance, with the screen disabled (`--no-screen`, coverage retained). All fourteen refuse and none produces a false admission: three reach the gate and are vetoed by the unrescuable instances now left in the replay set, while eleven exit at failure collection because resampled pools confirm no defect-attributable failure. The arm is consistent with the screen acting

as a downstream guard, but collection resampling dominates its outcomes and it does not isolate the screen's contribution.

Table 15: Matched coverage-off reruns (identical protocol and budgets; n_R = returned operators). Only the eleven cells whose confirmed $\mathcal{D}/\mathcal{T}$ membership changes under `ignore_coverage` were rerun. On-arm scores are recomputed under coverage; off-arm scores credit uncovered episodes.

Run	Cell	On arm	Off arm	Outcome
τ1 deepseek	`retail/tool_unavailable`	$1.0/1.0$ (n_R=2)	$1.0/1.0$ (n_R=2)	identical
τ1 deepseek	`retail/compound`	$0.0/0.0$	$0.0/0.0$	identical
τ1 deepseek	`airline/tool_unavailable`	$0.67/0.0$ (n_R=1)	$0.67/0.0$ (n_R=1)	identical
τ^2 v6	`airline/compound`	$0.67/1.0$ (n_R=2)	$0.0/0.0$	$\rightarrow$ refuse
τ^2 telecom	`telecom/tool_unavailable`	$0.0/0.0$	$0.0/0.0$	identical
τ^2 telecom	`telecom/tool_result_swap`	$0.0/0.0$	$0.0/0.0$	identical
τ^2 v6-qwen	`airline/compound`	$0.0/0.0$	$0.0/0.0$	identical
τ1 qwen-ext	`retail/schema_param_rename`	$0.0/0.0$	$0.0/0.0$	identical
τ1 qwen-ext	`airline/tool_output_corrupt`	$0.0/0.33$ (n_R=2)	$0.0/0.0$	$\rightarrow$ refuse
τ1 qwen-v2	`retail/compound`	$0.0/0.0$	$0.0/0.0$	identical
τ1 qwen-v2	`airline/tool_result_swap`	$0.0/0.0$	$0.0/0.0$	identical

The τ1-baselines rows (1046 episodes) complete the third-party-harness comparison. KEYWORD's two commits are both true admissions — `tool_unavailable` prints `tool unavailable: X` in the trace, so a symptomatic defect is genuinely keyword-solvable, which is exactly the easy-defect regime the paper motivates away from; the silent families leave it blind. LLM-DIAG never committed on τ1 (the weak local model's diagnoses were empty or out-of-DSL), and HARNESSFIX refused every cell — trace-evidence heuristics find nothing to match on the silent families that dominate this matrix. None of the three baselines committed a wrong mechanism, which is consistent with their near-total refusal profile. The HarnessFix rows describe only our internal heuristic under injected defects; they do not evaluate the external system. Reference-set inclusion is measurable here because the injected mechanisms and their reference operators are known.

G FROZEN-COHORT SELECTION COMPARISON

Purpose and relation to the primary matrix. The v7 study compares selection rules after fixing the task instances used for evaluation. It uses the seven evaluable airline/retail cells from the primary domain–defect grid, with three diagnosis and three transfer instances per cell and one clean probe. These are separate runs from the primary matrix: full TARPS has mean Rep@D/Tr@T of $0.762/0.762$ here, versus $0.714/0.762$ in the primary matrix. The two matrices are not pooled. The agent is `deepseek-v4-flash`, with the same DSL, vote counts, and coverage-restricted final scoring. Frozen membership pairs task identities across conditions; executions and candidate-test outcomes remain stochastic. Reuse of task pools across protocol development also means these are not fresh, independent confirmation tasks.

Selection rules. All checked-selection conditions use the task-action candidate enumeration and the singleton/pair selection structure in Algorithm 1; NoGate commits its first candidate, and Oracle applies the reference repair directly. The reward-only condition removes the reference screen, the coverage requirement during replay, and candidate clean-task checks. Reward plus clean checks restores Eq. 3, while the no-reference-screen condition additionally restores the coverage term in Eq. 2. Full TARPS also screens replay instances with the reference repair. Consequently, the first three checked conditions use all confirmed diagnosis instances for replay, whereas full TARPS may use a subset. The frozen clean-probe membership is retained even when a condition omits its candidate-level regression check. Final clean-probe scoring remains separate from admission.

The initial oracle-free condition substitutes trace-inferred events for $E(d)$ at admission and screens rescuability using only the first task-derived candidate. The event heuristic prioritizes an errored tool, then a budget-cap flag, then the last tool call; with no inferred event, the empty requirement passes coverage vacuously. It does not perform a clean-versus-defective execution comparison. The subsequent any-candidate condition screens each diagnosis instance with all task-derived single candidates,

Table 16: Per-cell Rep@D/Tr@T in the frozen-cohort comparison (percentages, rounded to integers). N: NoGate; R: reward only; RC: reward plus clean checks; RCC: RC plus coverage; Full: TARPS; OF1/OFA: oracle-free first/any-candidate screen. OFA is exploratory. Costs are given separately in Table 17.

Cell	N	R	RC	RCC	Full	OF1	OFA	Oracle
airline budget	0/0	100/100	100/100	100/100	0/0	0/0	100/100	100/100
airline compound	0/0	0/0	0/0	0/0	67/100	0/0	0/0	67/100
airline swap	67/67	0/33	33/33	0/0	67/67	67/100	33/0	67/67
airline unavailable	0/33	100/67	100/67	100/67	100/67	100/67	100/67	100/33
retail budget	0/0	0/0	100/100	100/67	100/100	0/0	100/100	100/100
retail swap	0/0	100/100	100/100	100/100	100/100	0/0	0/0	100/100
retail unavailable	0/0	100/100	100/100	100/100	100/100	0/0	100/100	100/100

retaining it when any candidate passes a reward majority. Both use clean-task regression and inferred coverage for admission. The latter screen was evaluated after inspecting the initial condition's poor recovery; it is an exploratory implementation revision, not an independent confirmation of a fixed screening rule. "Oracle-free" refers only to these admission inputs: failure collection and final scoring still use the known injected mechanisms and trigger events. The candidate space is also designed for the injected defect families.

Execution accounting. Frozen cohorts are collected once and reused (147 episodes across seven cells); this collection cost is excluded from each condition's reported episodes. The comparison drivers disable replay early termination, but the number of candidates and probes actually tested still depends on screening and admission. For designated conditions, the driver reads full TARPS's episode count and sets it as a minimum expenditure. After selecting and scoring the returned patch, it fills any shortfall with repeated executions of that patch on a replay or clean probe. These top-ups do not affect selection or scoring. Conditions can exceed the minimum, so the totals in Table 1 are neither equal decision budgets nor equal total costs. Episode counts also do not equate token usage. In particular, the any-candidate condition uses 1478 episodes versus 716 for full TARPS ($2.06\times$); changing both screen and expenditure does not identify the effect of additional budget alone.

Paired summaries and uncertainty. We average the per-cell instance success rates with equal weight over seven cells. For each contrast, the analysis resamples cells and then paired task outcomes within each sampled cell, using 20,000 bootstrap resamples with analysis seed 20260924. The resulting intervals describe the recorded task outcomes, not repeated full selection experiments. We also enumerate all 2^7 cell-level sign flips. These are descriptive small-sample sensitivity summaries: cells can share task identities, and neither independent random sampling of defect families nor a randomized assignment of selection rules is established. We additionally compare full TARPS with reward plus clean checks using the same analysis function. We report all recorded condition contrasts rather than treating the exploratory sequence as a preregistered confirmatory test.

Artifact selection and provenance. The supplementary `tau2_v7_ident` directory retains frozen cohorts, reports, and per-episode records. For each cohort-backed cell, `analyze_v7_ladder.py` reads reports from the base directory, then `__resume`, then lexically ordered `__arm_*` directories, with later entries replacing earlier reports of the same method. It verifies common diagnosis and transfer membership. The two `__topup` directories without completed reports remain provenance and do not supply table entries. Internal `ag_rpo` identifiers denote TARPS. The packaged analysis and verification record specify the selected reports so that interrupted attempts cannot be mistaken for additional independent observations.

Table 17: Recorded episodes by condition and cell, including decision-independent top-ups. The totals exclude shared cohort construction and incomplete attempts; they describe the selected completed reports. Abbreviations follow Table 16.

Cell	N	R	RC	RCC	Full	OF1	OFA	Oracle
airline budget	30	57	57	57	57	57	87	30
airline compound	30	147	210	210	107	107	107	30
airline swap	30	147	210	210	72	72	249	30
airline unavailable	30	84	84	84	84	84	135	30
retail budget	30	177	129	129	129	129	255	30
retail swap	30	129	129	129	129	129	390	30
retail unavailable	30	138	138	138	138	138	255	30
Total	210	879	957	957	716	716	1478	210

Table 18: Recorded paired contrasts on seven frozen cells. Differences and two-level bootstrap intervals are in percentage points; p is the exact two-sided cell sign-flip summary. OFA contrasts are exploratory. Intervals and sign flips quantify different resampling assumptions and have limited resolution in seven cells.

Contrast	Outcome	Difference	95% interval	p
OF1 − R	Repair	-33.3	$[-81.0, +14.3]$	0.2500
OF1 − R	Transfer	-33.3	$[-76.2, +14.3]$	0.2500
Full − RC	Repair	0.0	$[-38.1, +33.3]$	1.0000
Full − RC	Transfer	$+4.8$	$[-38.1, +42.9]$	1.0000
Full − R	Repair	$+19.0$	$[-28.6, +61.9]$	0.6250
Full − R	Transfer	$+19.0$	$[-28.6, +66.7]$	0.6250
RC − R	Repair	$+19.0$	$[+0.0, +47.6]$	0.5000
RC − R	Transfer	$+14.3$	$[+0.0, +42.9]$	1.0000
RCC − RC	Repair	-4.8	$[-19.0, +0.0]$	1.0000
RCC − RC	Transfer	-9.5	$[-28.6, +0.0]$	0.5000
Full − RCC	Repair	$+4.8$	$[-38.1, +42.9]$	1.0000
Full − RCC	Transfer	$+14.3$	$[-28.6, +57.1]$	0.7500
Full − OF1	Repair	$+52.4$	$[+14.3, +85.7]$	0.1250
Full − OF1	Transfer	$+52.4$	$[+9.5, +85.7]$	0.1250
OFA − R	Repair	$+4.8$	$[-38.1, +42.9]$	1.0000
OFA − R	Transfer	-4.8	$[-42.9, +38.1]$	1.0000
Full − OFA	Repair	$+14.3$	$[-28.6, +57.1]$	0.7500
Full − OFA	Transfer	$+23.8$	$[-28.6, +71.4]$	0.6250
Full − N	Repair	$+66.7$	$[+28.6, +100.0]$	0.0625
Full − N	Transfer	$+61.9$	$[+28.6, +95.2]$	0.0625