INVOCATION-LEVEL RELIABILITY OF TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

An early tool-call error can corrupt every downstream input even when an agent subsequently applies the right local rule. We isolate this effect with matched evaluations under a correct baseline history and the model's own free-running history. For tasks of depth d, their pooled invocation rates $\bar{p}_d$ and $\bar{g}_d$ define a fit-free relative propagation loss $L_d = 1 - \bar{g}_d/\bar{p}_d$. Across five open-weight models on controlled routing tasks, the two 7–8B models lose about 68% of baseline capability by depth 6; stronger models mostly remain at ceiling. Our main result is a construct-validity condition for trajectory scoring. Let Y_k^* be the canonical next target after a divergence and H_k the agent-observable history. If Y_k^* has conditional guessing probability at most ϵ_k given H_k, then any fixed-gold scorer credits the agent with probability at most ϵ_k, regardless of its local competence. Gold-agreement severity is therefore driven to a scorer-imposed boundary, and observed reconvergence cannot measure behavioral recovery. **Fixed reference alone is not sufficient**: the result applies when the canonical target becomes unreachable or unpredictable, and excludes reconstructible targets and evaluators that accept alternative states or milestone paths. Our hidden-constant tasks instantiate the condition: 0 of 869 corrupted-context calls match gold, and 0 of 580 opportunities reconverge. Conditional-on-state replay instead asks whether a call correctly continues from the state the agent actually holds; it yields interior severity estimates of 0.149 and 0.316 and requires no new model queries. Alternative absolute-gap and error-amplification summaries give the same qualitative depth trend.

1 INTRODUCTION

Language model agents interact with external systems by emitting structured *tool calls*: each call names a function (e.g. `get_weather`, `query_database`) and fills its required arguments. A call can fail by selecting the wrong function (*selection error*) or by providing incorrect argument values (*argument error*). In a multi-step task where each call's output feeds into the next, a single wrong call early in the chain corrupts the inputs to every downstream call, even if the model reasons correctly from that point on.

Many benchmarks report task success or agreement with a canonical call, without separately measuring selection, arguments, and the downstream effect of an earlier error. State-based benchmarks allow alternative solutions, and process benchmarks annotate individual steps, but neither directly supplies the matched baseline/free-running contrast studied here. These distinctions matter because local incompetence, state corruption, and ordinary later-step difficulty require different interventions.

We introduce a protocol that makes this distinction. A call is *correct* when it selects the appropriate tool and supplies the arguments required by the scoring regime. We measure it under two matched conditions: a *baseline* condition with a guaranteed-correct history and a *free-running* condition with the model's own prior outputs. Their gap measures the loss associated with accumulated state corruption while controlling for the prompts and later-step difficulty present in the baseline arm.

The study began as an empirical comparison and exposed a construct-validity problem. When divergence makes the canonical next target unpredictable from everything the agent observes, gold agreement measures target reachability rather than the competence of the next call. This is not a property of every fixed-reference benchmark: it depends on an explicit unreachable-target condition.

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

We state that condition, give a counterexample, and separate the resulting scorer-imposed boundary from the model's behavior.

Contributions.

1. A disaggregated measurement protocol that separately scores tool selection and argument correctness under matched baseline and free-running conditions, with a routing-task design where selection errors, not just argument errors, propagate.
2. A formal unreachable-target condition under which canonical gold agreement is bounded by the target's conditional guessing probability, making gold-based severity and reconvergence scorer-dominated rather than behavioral.
3. A remedy, conditional-on-state scoring, where each call is judged against what would be correct given the value the model actually holds: implemented and applied retrospectively to cached completions at zero marginal cost.
4. A controlled error-injection check that isolates the causal effect of state corruption from the confounds present in the observational baseline/free-running contrast, finding the observational conditional-severity estimate is inflated by roughly $2.9\times$ and that severity is confined to the selection channel.
5. Robustness summaries beyond L_d (absolute excess error and error amplification), plus per-model diagnostics for rule discrimination and error position.

2 Related Work

Toolformer (Schick et al., 2023), ReAct (Yao et al., 2023), and Gorilla (Patil et al., 2023) established tool-augmented agents. ToolLLM (Qin et al., 2024), StableToolBench (Guo et al., 2024), and API-Bank (Li et al., 2023) emphasize task success or agreement with reference calls. RoTBench (Ye et al., 2024) and MTU-Bench (Wang et al., 2025) separate selection from argument accuracy; FuncBenchGen (Maekawa et al., 2026) generates fresh dependency-graph tasks and reports stale-value failures. Our distinct estimand is the matched change in invocation correctness when gold history is replaced by the same model's history.

Not all agent evaluation uses one canonical trajectory. τ-bench (Yao et al., 2025) grades final environment state; AppWorld (Trivedi et al., 2024) uses state-based unit tests that permit multiple solutions; ToolSandbox (Lu et al., 2024) evaluates intermediate and final milestone DAGs over arbitrary trajectories. BFCL's multi-turn split (Patil et al., 2025) belongs to this group as well, notwithstanding that it distributes one reference call sequence per task: its checker compares final environment state and accepts the reference execution results as an unordered subset of the model's, so both reordering and additional calls pass, and its trajectory-order check (`method_invoke_order_checker`) is commented out at the call site and by its own docstring inspects method names but not arguments. These methods avoid the unreachable-canonical-target failure but primarily measure outcome or milestone completion, not local post-divergence competence. Section 5.3 states the narrower condition under which canonical-call scoring becomes scorer-dominated.

Process-oriented work is complementary. AgentProcessBench (Fan et al., 2026) provides human step labels and an explicit propagation rule, while FALAT (Rafi et al., 2026) traces dependencies to separate decisive errors from inherited downstream failures. Relign (Xu et al., 2025), Hammer (Lin et al., 2024), and representation-level detection (Healy et al., 2026) instead reduce or detect the production of an incorrect call. We intervene on history to quantify propagation and compare gold-agreement with conditional-on-state scoring.

3 Method

3.1 Definitions

A task of dependency depth d contains calls indexed by position $k \in \{1, \ldots, d\}$. For task i, let $B_{i,k}$ and $F_{i,k}$ indicate invocation correctness in the baseline and free-running arms. With N_d tasks at

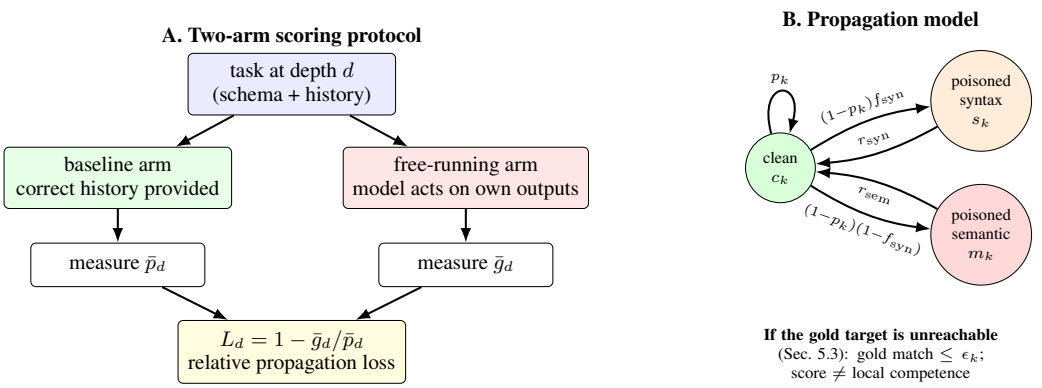

Figure 1: (A) Matched evaluation at task depth d: pooled baseline and free-running invocation rates define L_d. (B) A separate step-indexed propagation model. Under the unreachable-target condition of Section 5.3, canonical gold agreement is bounded by target predictability rather than local competence.

depth d, the reported depth-level estimands are

$$\bar{p}_d = \frac{1}{N_d d} \sum_{i=1}^{N_d} \sum_{k=1}^{d} B_{i,k}, \qquad \bar{g}_d = \frac{1}{N_d d} \sum_{i=1}^{N_d} \sum_{k=1}^{d} F_{i,k}. \tag{1}$$

- **Pooled baseline correctness** $\bar{p}_d$: invocation accuracy with a guaranteed-correct prior history.
- **Pooled free-running correctness** $\bar{g}_d$: invocation accuracy with the model's own prior outputs.
- **Relative propagation loss** $L_d := 1 - \bar{g}_d/\bar{p}_d$. We also report the absolute gap $A_d = \bar{p}_d - \bar{g}_d$ and error amplification $E_d = (1 - \bar{g}_d)/(1 - \bar{p}_d)$.

L_d is descriptive and invocation-pooled; it is not the stepwise quantity in the Markov recurrence below. Calls from one task are dependent, so all reported intervals use paired task-level bootstrap resampling: each of 2,000 draws samples N_d tasks with replacement, includes all positions from each selected task in both arms, recomputes $\bar{p}_d$, $\bar{g}_d$, and L_d within the replicate, and pairs baseline/free-running observations by task identity. The 89% level (5.5th and 94.5th percentiles) matches PyMC's default highest-density convention used elsewhere in this work. When a replicate draws only tasks with $\bar{p}_d = 0$, L_d is undefined and that replicate is excluded; this occurred in fewer than 0.1% of replicates at any depth.

3.2 A PROPAGATION MODEL WITH RECOVERY

For interpretation only, we model step-indexed context as clean, poisoned-by-syntax, or poisoned-by-semantics (Figure 1B). Let p_k be correctness at position k under clean history and $f_{\text{syn}}(k)$ the syntactic share of new errors. With origin-specific recovery rates $r_{\text{syn}}, r_{\text{sem}}$ and poisoned-state correctness $\rho_k = (1 - \pi)p_k$, occupancies evolve as

$$c_{k+1} = c_k p_k + s_k r_{\text{syn}} + m_k r_{\text{sem}}, \quad s_{k+1} = c_k(1 - p_k) f_{\text{syn}}(k) + s_k(1 - r_{\text{syn}}), \tag{2}$$

$$m_{k+1} = c_k(1 - p_k)(1 - f_{\text{syn}}(k)) + m_k(1 - r_{\text{sem}}). \tag{3}$$

The model-implied step accuracy is $g_k^{\text{model}} = c_k p_k + (s_k + m_k)\rho_k = p_k(1 - \pi x_k)$ for poisoned mass $x_k = s_k + m_k$. We fit $\pi, r_{\text{syn}}, r_{\text{sem}}$ hierarchically (PyMC, NUTS) as a cautionary diagnostic, not as the estimator of L_d. Section 5.3 explains why gold-agreement data do not identify the intended behavioral interpretation of these parameters under an unreachable target.

3.3 TASK DESIGN

Each task is a hidden dependency graph over deterministic executable functions, giving exact ground truth per call and direct control of depth ($\{1, 2, 4, 6, 8\}$). Tool outputs lie in a size-M space ($M = 100{,}000$) and take the form $(a \cdot \arg + b) \bmod M$, with per-tool constants omitted from the

schema. Section 5.3 does not infer unpredictability from secrecy alone; it states the conditional target-predictability assumption required for the bound.

Routing tasks (primary): the correct next tool is a parity rule applied to the incoming value, not a pre-announced order, so a wrong choice changes what is correct downstream. Selection is scored *conditionally*, against the tool correct given the value the model actually holds, with divergence from gold recorded separately, so applying the rule correctly to a corrupted input is not misclassified as selection failure. **Linear tasks (control)**: order announced, argument copied verbatim. Pilot runs were degenerate ($\bar{p}_d = \bar{g}_d = 1.000$ on two models), and the full null arm was not executed (Section 7).

3.4 Protocol and suite

We score selection against the tool appropriate for the value in the model's actual state and, for the primary gold-agreement analysis, score arguments against the canonical trajectory. Digit-only JSON strings are coerced to integers, so formatting is not a value error. Errors are syntactic (parse/schema failure) or semantic (executes but is wrong). Both arms use the same within-step retry budget: if a response fails to parse as a valid tool call (malformed JSON, missing required fields), the model is re-prompted with a structured error message up to two additional times; only the final attempt is scored as that step's invocation, and earlier failed attempts are recorded as syntactic errors but do not count toward $\bar{p}_d$ or $\bar{g}_d$. If all retries fail, the step is scored as a syntactic error and the free-running arm continues with the last parseable state. This policy is identical in both arms. Conditional-on-state replay replaces the argument target with the argument implied by the model's actual state, making the selection and argument criteria symmetric.

We evaluated five open-weight models available through one hosted API: `llama-3.1-8b-instant`, `allam-2-7b`, `qwen3.6-27b`, `gpt-oss-120b`, and `llama-3.3-70b-versatile`. Greedy decoding, fixed seeds, and caching by (model, calling mode, exact prompt) make each benchmark prompt deterministic. Provider constraints produced unequal nested sample sizes and prevented balanced family, proprietary-model, and calling-mode comparisons; we draw no claims along those axes.

Models run **nested prefixes** of one task suite, sized per model to that model's own measured provider-imposed token-rate allowance. Task identity depends only on $(\mathrm{depth}, k, \mathrm{seed})$, never on how many tasks were requested, so cross-model contrasts on the shared prefix remain exactly paired while each model's own estimate uses its full n.

4 Data Collected

All figures come from a dataset frozen at 14:09 on 2026-08-17; no missing model–depth cell is extrapolated.

Table 1: Per-model sample sizes at the freeze.

Model	tasks	d_1	d_2	d_4	d_6	d_8	status
`llama-3.1-8b-instant`	**273**	60	60	65	65	23	only model at depth 8
`allam-2-7b`	**225**	60	60	65	40	–	usable
`gpt-oss-120b`	**108**	26	26	28	28	–	usable
`qwen3.6-27b`	**98**	26	26	28	18	–	usable
`llama-3.3-70b-versatile`	**49**	12	12	13	12	–	usable, thin
`gpt-oss-20b`	**0**	–	–	–	–	–	**excluded**

`gpt-oss-20b` completed zero tasks under this protocol: an August pilot exhausted the provider's daily request cap before the primary run began, and a later re-attempt found the provider frequently, though not universally, rejects this model's native tool calls under our text-only protocol with an HTTP 400 error even on a correct answer (confirmed against a returned `failed_generation` payload), at a rate too high to assemble a scoreable run. The same task succeeds cleanly once tools are declared on the request; a native-mode probe run on this basis is reported in Appendix G.

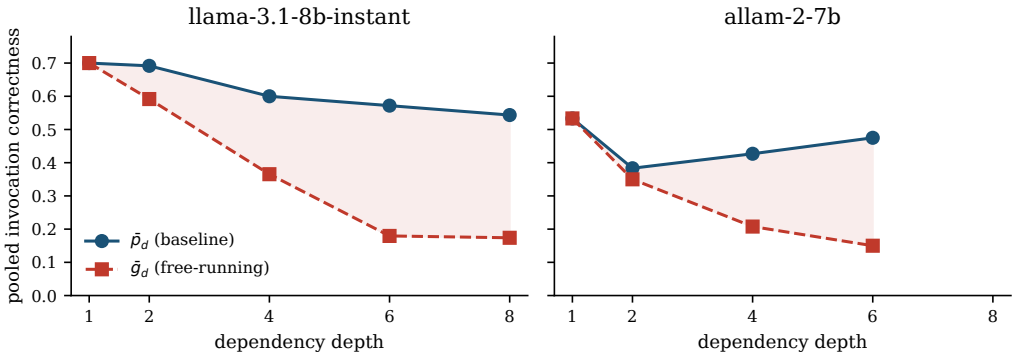

Figure 2: Pooled baseline correctness $\bar{p}_d$ versus pooled free-running correctness $\bar{g}_d$. The absolute gap widens through depth 6 and then plateaus for `llama-3.1-8b`; depth 1 is identical by construction.

The model remains excluded from every pooled claim. The original configuration's Qwen2.5 and Llama-3.1-instruct identifiers were already stale, retired by the provider before the first run.

5 Results

5.1 Propagation grows through depth 6 and exceeds baseline decay

Figure 2 and Table 2 show the core measurement. On `llama-3.1-8b`, $\bar{p}_d$ declines from 0.700 to 0.543 (22% relative), while $\bar{g}_d$ declines to 0.174 (75%). L_d rises sharply through depth 6 and is essentially unchanged at depth 8 (0.686 vs. 0.680); we therefore claim growth followed by a plateau, not strict monotonicity or significant adjacent-depth differences.

Table 2: Invocation-pooled depth estimands for the two high-n models. Intervals are the reported 89% intervals from the frozen paired analysis.

depth d	$\bar{p}_d$	$\bar{g}_d$	L_d [89% interval]	calls
`llama-3.1-8b-instant`				
1	0.700	0.700	**0.000** [0.000, 0.000]	60
2	0.692	0.592	0.145 [0.082, 0.222]	120
4	0.600	0.365	0.391 [0.311, 0.474]	260
6	0.572	0.179	**0.686** [0.607, 0.766]	390
8	0.543	0.174	**0.680** [0.574, 0.772]	184
`allam-2-7b`				
1	0.533	0.533	**0.000** [0.000, 0.000]	60
2	0.383	0.350	0.087 [0.022, 0.160]	120
4	0.427	0.208	0.514 [0.431, 0.596]	260
6	0.475	0.150	**0.684** [0.624, 0.745]	240

$L_1 = 0$ **exactly on every model**: both arms build an identical depth-1 prompt and share one cached response, providing an implementation invariant. At depth 6, `llama-3.1-8b` has identical baseline/free error counts at the first position (21/21), after which free-running errors rise while baseline errors remain approximately flat. The absolute gap and error-amplification summaries show the same through-depth-6 increase without dividing by baseline accuracy.

The lag association $\Delta_1 = P(k{+}1 \text{ correct} \mid k \text{ correct}) - P(k{+}1 \text{ correct} \mid k \text{ wrong})$ is positive for every model with errors ($+0.32$ to $+1.00$). It is descriptive rather than causal; its second term is zero under canonical gold agreement because of the unreachable-target mechanism below.

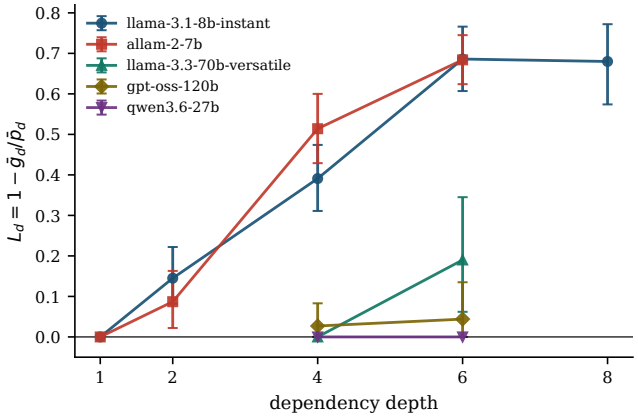

Figure 3: L_d versus depth for all usable models (89% intervals). Three models remain near ceiling; `llama-3.3-70b-versatile` shows a positive depth-6 estimate.

5.2 Ceiling models, and where in a chain a model errs

Three models sit at or near ceiling ($\bar{p}_d = 0.865$–1.000; Figure 3), so $L_d \approx 0$ is primarily a statement about task difficulty. On `llama-3.3-70b-versatile` at depth 4, 6 of 7 errors occur at a terminal position, leaving no downstream opportunity. At depth 6 its estimate is $L_6 = 0.190$ $[0.062, 0.345]$, but the sample is thin. `gpt-oss-120b` has one fresh error, and `qwen3.6-27b` has zero errors across 298 free-running calls; neither supports a robustness claim.

An error at the terminal position cannot propagate, so a fixed-depth aggregate mixes error frequency with the remaining opportunity for downstream damage; $L_1 = 0$ is the limiting case. For `llama-3.1-8b`, terminal-error shares are 0.714 at $d = 2$ and 0.197 at $d = 6$, versus uniform-position baselines of 0.500 and 0.167. For `llama-3.3-70b-versatile`, the 6/7 terminal share at $d = 4$ is suggestive but extremely uncertain. Terminal-error share should therefore accompany L_d, and cross-model comparisons require comparable position distributions.

5.3 When canonical gold agreement ceases to measure behavior

Let H_k denote everything available to the agent before a post-divergence call, and let Y_k^* be the canonical gold target used by the scorer. Define its conditional guessing probability

$$\epsilon_k(H_k) = \max_y P(Y_k^* = y \mid H_k). \tag{4}$$

Proposition (unreachable-target bound). For any possibly randomized agent whose prediction $\widehat{Y}_k$ depends only on H_k,

$$P(\widehat{Y}_k = Y_k^* \mid H_k) \le \epsilon_k(H_k). \tag{5}$$

Consequently, if q_k^{gold} is correctness in corrupted context and $p_k > 0$ is clean-context correctness, gold-defined severity satisfies $\pi_k^{\text{gold}} = 1 - q_k^{\text{gold}}/p_k \ge 1 - \mathbb{E}[\epsilon_k]/p_k$. Across recovery opportunities $\mathcal{K}$, the probability of any chance reconvergence is at most $\sum_{k\in\mathcal{K}} \mathbb{E}[\epsilon_k]$.

Proof. Conditional on H_k, sum over candidate outputs: $P(\widehat{Y}_k = Y_k^* \mid H_k) = \sum_y P(\widehat{Y}_k = y \mid H_k) P(Y_k^* = y \mid H_k) \le \max_y P(Y_k^* = y \mid H_k)$. The recovery statement is the union bound. □

The proposition is about *construct validity*: when ϵ_k is negligible, gold agreement cannot distinguish a locally competent continuation from arbitrary behavior. It does not say the operational statistic is numerically unidentified; rather, the statistic is pinned near a boundary that no longer represents the intended behavioral severity.

Instantiation in this suite. After divergence, canonical arguments depend on the hidden-constant output associated with the gold state. Under the generator assumption that this target is conditionally

uniform over $M = 100{,}000$ values, $\epsilon_k = 10^{-5}$. We observe 0 gold matches among 869 corrupted-context calls and 0 canonical reconvergences among 580 opportunities. Even a pure-guess process has expected reconvergences $580/M = 0.0058$, with probability of any such event at most 0.0058. Zero therefore carries essentially no information about whether the agent can apply the local routing/copy rule in its actual state.

If one nevertheless substitutes the scorer-implied boundary $\pi = 1$ and zero canonical recovery into the step model, it collapses to $g_k^{\text{model}} = c_k p_k$ and $c_k = \prod_{j<k} p_j$. A hierarchical fit to pooled aggregates returned severity posteriors of 0.92 and 0.73 despite zero observed post-divergence gold matches; $\hat{R} \leq 1.0005$, ESS $\geq 5{,}823$, and zero divergences show only that the misspecified posterior was explored successfully: a targeted posterior predictive check finds 0 of 6,000 replicates reproducing the observed zero canonical matches among corrupted-context calls, against a predicted mean of 67.7.

Counterexample and scope. Fixed reference is not sufficient for the proposition. If Y_k^* is reconstructible from the observable history, independent of the corrupted value, or reachable through reset/re-query actions, ϵ_k may be large and canonical recovery can be meaningful. Final-state unit tests and milestone-DAG evaluators can also accept alternative valid trajectories. Our claim is therefore restricted to canonical scoring under an empirically or structurally small ϵ_k; it is not a blanket claim about execution match, AST match, or fixed references.

5.4 A REMEDY: CONDITIONAL-ON-STATE SCORING

Credit a call when it *correctly continues from the value the model actually holds*, not when it matches the canonical state. Routing selection was already computed conditionally; replay adds the argument-side counterpart. Because every raw completion was cached, this required 881 cache reads and no new model queries.

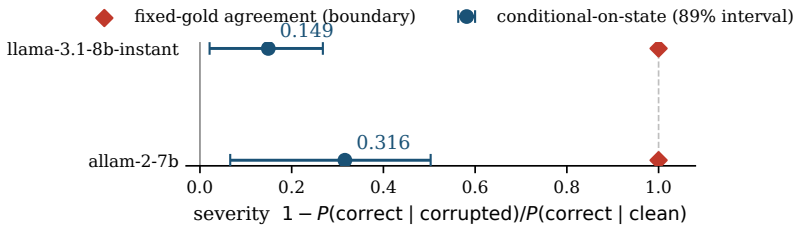

Figure 4: Gold-agreement severity is pinned at its boundary, whereas conditional-on-state replay yields interior estimates (reported 89% intervals). The contrast is a change of estimand, not an improved fit to the same target.

Clean-context calls score identically under both rules (0.658 vs. 0.658), an implementation invariant. Conditional severity is 0.149 $[0.021, 0.268]$ for `llama-3.1-8b` and 0.316 $[0.066, 0.503]$ for `allam-2-7b`. Thus state corruption is associated with genuine loss of local rule application, but far less than the boundary value assigned by canonical gold agreement. Evidence is limited to two models and a weak copy-argument competence test. A controlled error-injection check on `allam-2-7b` alone finds a smaller causal estimate, $\hat{\pi} = +0.108$ $[+0.006, +0.215]$ for selection-channel corruption (argument-channel corruption showed no effect), indicating the observational estimate above is inflated by roughly $2.9\times$: the free-running arm lets the model choose which state to corrupt, while the injection arm assigns it, and only the latter isolates a causal effect.

Trade-off. Conditional scoring requires an executable transition model for the agent's actual state and does not imply end-task success. Canonical gold, conditional-on-state, milestone, and final-state scores answer different questions; none should be presented as a universal replacement.

5.5 RULE-FOLLOWING IS MEASURABLE DIRECTLY, AND AGGREGATES HIDE IT

Define **discrimination** as $P(\text{first-listed} \mid \text{ref even}) - P(\text{first-listed} \mid \text{ref odd})$: 0 means the rule is ignored, $+1$ perfect application, negative means *anti*-correlated. Overall first-listed rates (0.478–0.497, four of five models) are indistinguishable and consistent with no position bias whatever; the

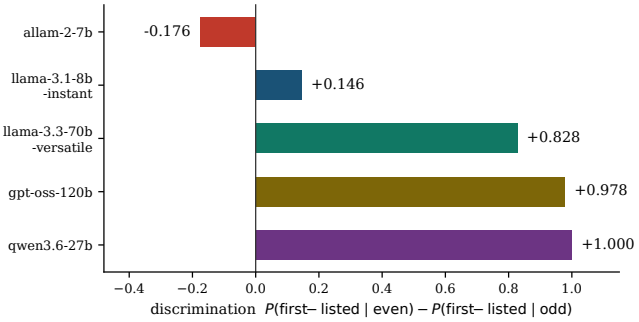

Figure 5: Discrimination, defined as $P(\text{first-listed} \mid \text{even}) - P(\text{first-listed} \mid \text{odd})$, per model. Order matches the reliability ranking of Section 5.1 (both computed from the same routing data).

same data, conditioned on parity, resolves discrimination across the full range (Figure 5), and the order matches reliability established independently in Section 5.1. **Discrimination is a per-model diagnostic of whether a model performs the task at all, invisible to any metric aggregated over the conditioning variable.**

Three qualitatively different relationships, not one spectrum: `qwen3.6-27b` *perfectly follows* the rule ($+1.000$, zero errors in 298 steps); `llama-3.1-8b` *weakly follows* it ($+0.146$, z=4.7); `allam-2-7b` is *anti-correlated* (-0.176, $z = -5.0$): it picks the first-listed tool *more* on odd refs (0.774) than even (0.597), where its accuracy is worst (0.223 vs. 0.595). `allam-2-7b`'s propagation loss is therefore not "a weak model that errs more" but a model *not performing the routing task*.

5.6 Two negative results

Untestable, not underpowered: a hypothesised error-composition shift along the chain cannot be tested here because **389 of 389** clean-context errors are selection errors: on a copy-argument task there are only two ways to fail cleanly, mis-apply the parity rule (hard) or mis-transcribe a verbatim number (models never do). **Unresolved by design**: the suite offers two 2-point scale contrasts rather than 3–4, not comparable in kind (`llama` 8B→70B dense, $8.75\times$; `gpt-oss` 20B→120B, $6\times$ total but $\approx 1.4\times$ active params). We report both axes and draw no scale conclusion, since the larger models sit at ceiling on these tasks.

6 Pipeline Validation, and What Simulation Cannot Establish

Before spending API budget we validated the full pipeline against six simulated policies with known ground truth, eliminating seven harness artifacts (retry-budget asymmetry, selection errors mis-attributed under poisoning, a contaminated syntactic-share estimator, among others). All seven were discovered and fixed before any real-model data was collected; 14 regression-test modules lock the corrected behavior and pass on the frozen dataset. Two further artifacts were not coding errors, which is what makes them instructive: comparing carried value against *expected argument* rather than *previous gold output* is correct on a copy-argument task and silently wrong under any transform; and the simulated policies recovered by reading ground truth directly, out of band, through a channel no real model has. The second artifact is why the recovery channel was never estimable from observable data in any version of this pipeline. The 81% parser-failure finding for `qwen3.6-27b` was discovered during post-collection cache replay; the parser was fixed, cached completions were re-scored (not re-collected), and no completion changed its parsed value (the failure mode was clean breakage, not silent corruption).

Three principles follow, nested by scope: **(1) simulated validation** certifies only the channels it exercises and silently passes the ones it does not (our mock backend read state from stashed context rather than the message history, so an agent loop that never threaded observations into the conversation passed the entire simulated suite); **(2) a parser or scorer can silently flip the *direction* of a result**, not just its precision: replaying cached completions through our pre-fix extractor, **319/394 (81.0%)** of `qwen3.6-27b`'s responses would have failed to parse; **(3) every constant estimated from**

one model or assumed for convenience should be verified per-model: a shared output-token constant hid `qwen`'s true cost (6.6× higher); an assumption that parity was incidental hid `allam`'s anti-correlation; per-depth pooling hid where each model's errors cluster.

This establishes that the estimation and aggregation pipeline recovers configured parameters. It does **not** establish the prompt-construction and observation-passing path, and no out-of-band simulation can.

7 Limitations and Scope

This study uses synthetic integer-argument tasks, giving exact state transitions at the cost of ecological validity. The strongest empirical findings are carried by two 7–8B models; three stronger models are largely at ceiling. The unreachable-target proposition is general only through its explicit ϵ_k condition, while the empirical claim that our generator makes ϵ_k negligible depends on its hidden-state construction. Validation on natural-language arguments and stateful environments such as ToolSandbox or AppWorld is required before claiming prevalence in deployed agents. We began that validation on BFCL's multi-turn split and found one structural obstacle in the remedy itself: conditional-on-state scoring needs a rule mapping the agent's actual state to the correct call, and a task whose gold trajectory encodes user intent rather than a decision rule supplies no such mapping. Section 5.4 therefore does not transfer to that setting without an additional model of intent, and we do not report conditional severity there.

Three designed conditions were not executed: transformed arguments, randomized presentation order, and a calling-mode ablation. A linear-task control (Appendix D) and a recovery-enabled variant (Appendix F) were each run on `allam-2-7b` alone. The linear control ($L_6 = 0.156$ vs. 0.684 on routing) confirms the baseline/free-running contrast captures more than selection-error propagation alone. The recovery variant shows restoring canonical state measurably raises downstream gold agreement ($+0.126$ at depth 6) but not completion of the original task ($+0.054$, interval includes zero), and the gain decays with steps remaining. `allam-2-7b`'s anti-correlation may depend on parity/order presentation, and copy arguments do not test semantic construction. Three of the five original models (`llama-3.1-8b-instant`, `llama-3.3-70b-versatile`, `qwen3.6-27b`) were retired from the hosting provider during this study, blocking further ablations against their baselines: the fourth unannounced infrastructure failure here, counting the earlier TPD caps (Section 4). A local-quantized replication of `llama-3.1-8b` was tested for equivalence against its frozen provider-hosted arm on matched tasks, under a rule fixed before those data existed: agreement within 0.05 and an 89% paired interval containing zero, at every shared depth and run mode. It failed in five of ten cells (depths 6 and 8 under teacher forcing; depths 4, 6 and 8 free-running), and the local arm exceeded the frozen arm in all ten cells with the gap widening in depth, indicating a systematic offset rather than sampling variation; pooling the two would have moved L_6 from 0.687 to 0.518. The cause cannot be isolated: the retirement preceded the comparison, so quantization and the provider's serving stack are confounded.

8 Conclusion

Matched baseline/free-running evaluation reveals large propagation on two small models: at depth 6, free-running correctness is about 68% below each model's baseline rate. Absolute excess error and amplification show the same pattern.

The broader result is conditional. When the canonical next target has negligible predictability from the post-divergence history, any gold scorer credits local behavior only at that guessing probability, so gold-defined severity approaches a scorer-imposed boundary and canonical reconvergence does not measure behavioral recovery. Conditional-on-state replay instead asks about local continuation quality, yielding interior estimates in this suite.

The practical rule is narrower: **before interpreting post-divergence gold agreement, estimate whether the gold target remains reconstructible from what the agent can observe.** If not, report state-conditional competence separately from milestone or final-state success.

AI USE STATEMENT

We used generative AI tools (Claude, via Claude Code and the Claude chat interface) in several capacities. For writing: revising and polishing manuscript text across drafting rounds, and drafting full sections and passages, including portions of the appendices, that were reviewed and edited by the authors before inclusion. For retrieval and discovery: searching for and verifying related work, including confirming citation details against original sources and reading a cited benchmark's scoring implementation to correct a claim in Related Work. For research ideation and execution: designing and implementing several of the experimental ablations reported here (including the error-injection design, the recovery-enabled task variant's mechanism, and the local-versus-hosted equivalence check), writing the corresponding analysis and validation code, and running the resulting experiments. The overall research direction, the central formal result, and the interpretation of all findings are the authors' own; all experimental designs and results produced with AI assistance were reviewed, tested, and verified by the authors before being reported.

ETHICS STATEMENT

This study evaluates publicly available open-weight language models on synthetic arithmetic tasks via a hosted inference API, within the provider's terms of service. It involves no human subjects, no personal or sensitive data, and no dataset release beyond the synthetic task generator described in Section 3.1. We are not aware of a plausible harmful use of this paper's central finding, that certain scoring regimes conflate target unpredictability with model incompetence; the primary implication is methodological, toward more accurate evaluation of agentic tool use.

REPRODUCIBILITY STATEMENT

The task generator, scoring pipeline, bootstrap procedure, and hierarchical model are fully specified in Section 3.1 through Section 5.4, including exact model identifiers, sampling parameters, the retry policy, and the resampling scheme used for every reported interval (Section 3.1). The frozen dataset, cached model completions, task-generation and analysis code, and the regression-test suite that locks the pipeline's scoring behavior will be released at an anonymized URL upon acceptance; Section 6 documents the pipeline-validation process and Section 7 documents every condition that was not executed. The proof of the unreachable-target proposition is given in full in Section 5.3, with its assumptions and scope stated alongside it.

REFERENCES

T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv:2305.15334*, 2023.

Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *ICLR*, 2024.

Z. Guo, S. Cheng, H. Wang, S. Liang, Y. Qin, Y. Li, B. Liu, Z. Liu, and M. Sun. StableToolBench: Towards stable large-scale benchmarking of tool learning. In *Findings of ACL*, 2024.

M. Li, Y. Zhao, B. Yu, F. Song, H. Li, H. Yu, Z. Li, F. Huang, and Y. Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *EMNLP*, 2023.

S. G. Patil, H. Mao, F. Yan, et al. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation. In *ICML*, 2025.

S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *ICLR*, 2025.

J. Ye, S. Li, G. Li, C. Huang, S. Gao, Y. Wu, Q. Zhang, T. Gui, and X. Huang. RoTBench: A multi-level benchmark for evaluating the robustness of large language models in tool learning. In *EMNLP*, 2024.

P. Wang, S. Liu, F. Meng, W. Chen, S. Wang, and J. Zhou. MTU-Bench: A multi-granularity tool-use benchmark for large language models. In *ICLR*, 2025.

S. Maekawa, J. Hassell, P. Pezeshkpour, T. Mitchell, and E. Hruschka. Towards reliable benchmarking: A contamination free, controllable evaluation framework for multi-step LLM function calling. In *ICLR*, 2026.

H. Xu, Z. Zhu, L. Pan, Z. Wang, S. Zhu, D. Ma, R. Cao, L. Chen, and K. Yu. Reducing tool hallucination via reliability alignment. In *ICML*, 2025.

Q. Lin, M. Wen, Q. Peng, G. Nie, J. Liao, J. Wang, X. Mo, J. Zhou, C. Cheng, Y. Zhao, J. Wang, and W. Zhang. Hammer: Robust function-calling for on-device language models via function masking. *arXiv:2410.04587*, 2024.

K. Healy, B. Srinivasan, V. Madathil, and J. Wu. Internal representations as indicators of hallucinations in agent tool selection. *arXiv:2601.05214*, 2026. Presented at the AAAI 2026 TrustAgent Workshop.

H. Trivedi, T. Khot, M. Hartmann, R. Manku, V. Dong, E. Li, S. Gupta, A. Sabharwal, and N. Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *ACL*, 2024.

J. Lu, T. Holleis, Y. Zhang, B. Aumayer, F. Nan, F. Bai, S. Ma, S. Ma, M. Li, G. Yin, Z. Wang, and R. Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv:2408.04682*, 2024.

S. Fan, X. Ye, Y. Huo, Z.-Y. Chen, Y. Guo, S. Yang, W. Yang, S. Ye, J. Chen, H. Chen, X. Cong, and Y. Lin. AgentProcessBench: Diagnosing step-level process quality in tool-using agents. *arXiv:2603.14465*, 2026.

M. N. Rafi, M. Ahasanuzzaman, D. J. Kim, Z. Wang, and T.-H. Chen. FALAT: Tracing failures in LLM agent trajectories via dependency-guided search. *arXiv:2606.00765*, 2026.

A AGGREGATE-DATA SENSITIVITY CHECKS

Table 3 reports the aggregate correct/total counts underlying Table 2, together with absolute excess error A_d and error amplification E_d. Task-cluster confidence intervals require the per-task records in the release artifact; the deterministic summaries below do not.

Table 3: Aggregate counts and alternative fit-free propagation summaries. $A_d = \bar{p}_d - \bar{g}_d$; $E_d = (1 - \bar{g}_d)/(1 - \bar{p}_d)$.

Model	d	baseline	free-running	A_d	E_d
`llama-3.1-8b`	1	42/60	42/60	0.000	1.000
	2	83/120	71/120	0.100	1.324
	4	156/260	95/260	0.235	1.587
	6	223/390	70/390	0.392	1.916
	8	100/184	32/184	0.370	1.810
`allam-2-7b`	1	32/60	32/60	0.000	1.000
	2	46/120	42/120	0.033	1.054
	4	111/260	54/260	0.219	1.383
	6	114/240	36/240	0.325	1.619

B WHAT EACH SCORING REGIME IDENTIFIES

C POSTERIOR PREDICTIVE CHECK FOR THE STEP-MODEL FIT

Section 5.3 states that the hierarchical fit is misspecified rather than merely uncertain. This appendix shows the check behind that claim, and the reason a conventional diagnostic does not detect it.

The fitted model implies that a call issued on a corrupted context succeeds with probability $(1 - \pi)p_k$, which is strictly positive for any $\pi < 1$. Summed over the corrupted-context calls

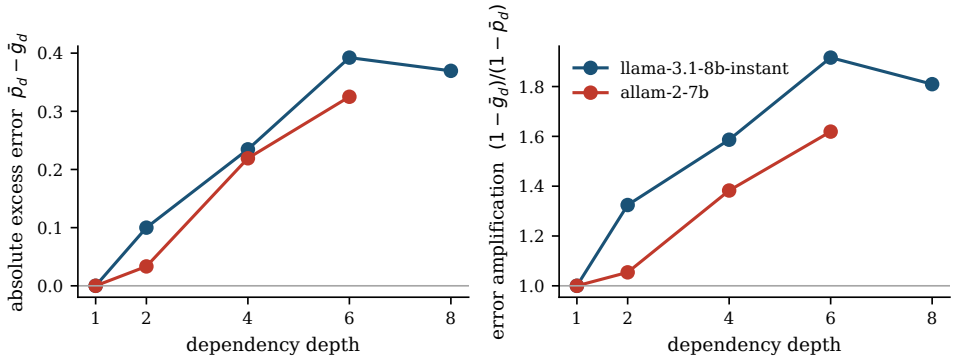

Figure 6: The propagation pattern is not specific to $L_d = 1 - \bar{g}_d/\bar{p}_d$. Absolute excess error grows to 0.392 and 0.325 by depth 6; free-running errors are amplified by factors of 1.916 and 1.619 relative to baseline errors. Both metrics plateau or terminate after depth 6 rather than supporting strict monotonicity.

Table 4: Scoring regimes answer complementary questions.

Scorer	Alternative paths	Primary estimand	Post-divergence limitation
Canonical call/trajectory match	Usually no	Agreement with one reference	Local competence is confounded when the target has small ϵ_k
Conditional-on-state (ours)	Yes, one state at a time	Correct local continuation	Does not imply final task success
Milestone DAG (ToolSandbox)	Yes	Intermediate/final progress	Attribution depends on milestone coverage
Final-state tests (AppWorld, τ-bench)	Yes	End-task state and side effects	Usually lacks per-call diagnosis

in the suite, it therefore predicts a positive number of canonical matches. The observed number is zero. We rebuilt the fitted model with one additional observed node for that subset and drew from `pm.sample_posterior_predictive` against the existing trace; no new model queries were required.

Table 5: Posterior predictive checks on two fits. The targeted check evaluates canonical matches among corrupted-context calls only; the aggregate check evaluates total free-running successes, which is the quantity such a fit is ordinarily judged on. Both fits pass the aggregate check and fail the targeted one.

	Real-model fit	Simulated-suite fit (control)
Corrupted-context calls	869	9,624
Observed canonical matches	**0**	1,648
Predicted mean	67.7	3,173
Predicted 89% interval	[39, 98]	[2,899, 3,427]
Minimum over 6,000 replicates	10	2,427
Targeted check, $\Pr(\text{rep} \leq \text{obs})$	0.000	0.000
Aggregate check, Bayesian p	**0.420**	**0.282**

Two points follow. First, **the conventional check would have certified this fit as adequate**. Replicated total free-running successes match the observation closely ($p = 0.420$), because the marginal rate g_k averages over clean and corrupted contexts and the clean ones dominate the counts. A posterior predictive check aimed at the aggregate is structurally blind to a failure confined to the corrupted subset, so passing it is not evidence that the step model describes post-divergence behavior. Reporting an aggregate predictive check alongside boundary-valued severity posteriors is therefore not a safeguard.

Second, the same targeted check fails on the simulated suite, where the generating policies *were* configured to recover and the observed count is a substantial 1,648. The fit over-predicts there too, by roughly a factor of two. The over-prediction is thus a property of the fit, which under-estimates

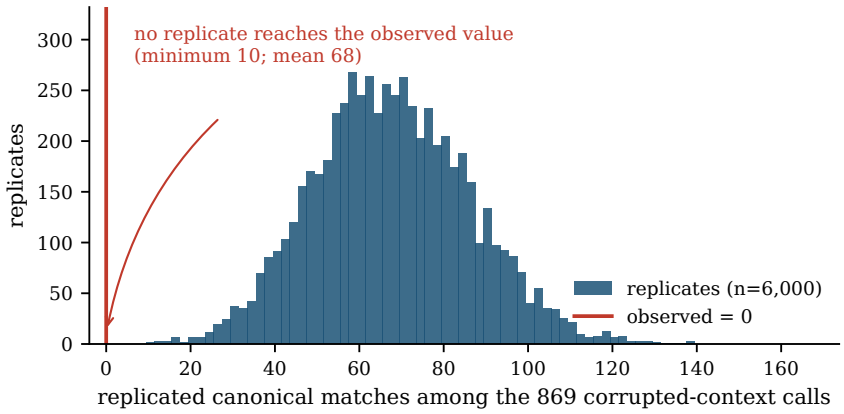

Figure 7: Posterior predictive distribution of canonical matches among the 869 corrupted-context calls in the real-model fit, against the observed value. No replicate reaches the observation; the smallest of 6,000 is 10.

severity so $(1 - \pi)$ is too large, and not an artifact peculiar to real data sitting at a scorer-imposed boundary. This is what distinguishes the finding from a claim that real models are simply harder to fit.

D LINEAR-TASK CONTROL ARM

The linear variant announces the tool order in the prompt, so the correct tool at each step is fixed in advance and cannot depend on the value carried in. What it retains is argument-value copying: each argument is a verbatim copy of the previous observation, so a wrong value still propagates forward. The arm was executed on `allam-2-7b` at the routing arm's depths and sample sizes. Depth 6 stopped when the hosting provider's daily token-rate cap was reached at 31 of 40 planned tasks; the achieved counts are a nested prefix and are used as such.

Table 6: Invocation-pooled depth estimands for `allam-2-7b` on both task families. Intervals are 89% paired bootstrap intervals over tasks, the same estimator and convention used for the routing arm.

Linear (selection dependency removed)				
depth d	$\bar{p}_d$	$\bar{g}_d$	L_d [89% interval]	calls
1	0.950	0.950	**0.000** $[0.000, 0.000]$	60
2	0.992	0.983	0.008 $[0.000, 0.026]$	120
4	0.962	0.908	**0.056** $[0.028, 0.086]$	260
6	0.930	0.785	**0.156** $[0.089, 0.229]$	186
Routing (selection dependency present)				
1	0.533	0.533	**0.000** $[0.000, 0.000]$	60
2	0.383	0.350	0.087 $[0.022, 0.160]$	120
4	0.427	0.208	0.514 $[0.431, 0.596]$	260
6	0.475	0.150	**0.684** $[0.624, 0.745]$	240

Linear tasks remove the selection dependency but retain argument-value copying, so the two arms differ in whether an upstream error can change which tool is correct downstream, not in whether a wrong value is carried forward. At depth 6 the routing loss is $4.4\times$ the linear one (0.684 against 0.156), which is consistent with selection-error propagation accounting for the majority of total loss. The linear intervals at depths 4 and 6 both exclude zero, consistent with the baseline/free-running contrast capturing effects beyond selection-error propagation, though the specific mechanism is not resolved by this control alone; the linear arm is therefore a bound on the selection dependency's

contribution rather than a null. The two families share the harness, the scorer and the prompt length at each depth, so neither of those explains the gap between them.

E ERROR-INJECTION ABLATION

To test whether the observational propagation estimate reflects a causal effect of state corruption rather than an arm-construction confound, we injected controlled errors into otherwise identical trajectories for `allam-2-7b` (52 of 60 planned tasks, 316 pairs). At a specified position $j \in \{1, 3, 5\}$, two branches diverge from the same prefix: a *corrupted* branch where one tool output is replaced with a wrong value, and a *corrected* branch where the original value is preserved. Both branches then continue through the same model. Because both share an identical prefix and differ in exactly one message, the contrast isolates state corruption from prompt-style, retry-history, and demonstration-quality differences.

Validity check. The corrected branch reproduces the baseline arm's accuracy exactly across all 40 overlapping tasks at every position tested ($+0.000$ at $j = 1, 3, 5$), confirming the injection mechanism introduces no construction artifact.

Two injection modes. `parity_flip` changes the carried value's parity, altering which tool the routing rule selects (a selection-channel corruption). `parity_preserving` changes the value without changing its parity, so the correct tool is unchanged but the argument is wrong (an argument-channel corruption).

Table 7: Causal severity from controlled error injection on `allam-2-7b`, scored conditional-on-state. Intervals are 89% paired bootstrap intervals resampling pairs. Discordant counts are the pairs in which exactly one branch was correct: b favours the corrected branch, c the corrupted one.

condition	corrected	corrupted	$\hat{\pi}$ [89% interval]	b/c
`parity_flip` (selection)	0.411	0.304	$+0.108$ $[+0.006, +0.215]$	65/48
`parity_preserving` (argument)	0.411	0.443	-0.032 $[-0.057, -0.013]$	0/5
observational (routing free arm)	n/a	n/a	$+0.316$ $[+0.066, +0.503]$	n/a

Three findings follow. First, the observational estimate is biased upward by approximately $2.9\times$ ($+0.316$ against $+0.108$), and the observational value lies outside the causal interval. In the free-running arm the model selects its own corruptions, so whatever produced the upstream error remains present downstream; the injected contrast removes that dependence by assigning corruption externally. Second, the effect is confined to the selection channel: `parity_preserving` corruption produced no degradation at any position, with zero pairs favouring the corrected branch, so a wrong argument value alone does not measurably impair this model's next call. Third, severity is not constant along the chain. The position profile is $+0.132$ at $j = 1$, $+0.226$ at $j = 3$ and -0.038 at $j = 5$, of which only $j = 3$ excludes zero. The step model of Section 3.2 treats π as position-independent, and these estimates do not support that assumption.

These results are from a single model at one depth and a truncated allocation. The pooled selection-channel interval only narrowly excludes zero and one of three positions carries the effect, so the ablation establishes the direction of the observational bias more firmly than it establishes the magnitude of the causal severity. The eight tasks not collected would not change that. We report it as a single-model causal check on the estimand, not as suite-wide validation.

F RECOVERY-ENABLED TASK VARIANT

Section 5.3 shows that after a divergence the canonical target is unreachable, so gold agreement cannot express recovery; Section 5.4 scores calls conditional on the state actually held instead. Neither distinguishes an agent that continues competently from a wrong state from one that returns to the right one, because the routing task offers no route back. This variant adds one.

Mechanism. A single additional tool, `check_state`, returns the canonical reference value for the step the agent is on. It consumes a call, does not advance the task, and is usable once per task. The one-shot budget makes usage binary, and excludes the degenerate policy of resynchronising before every step, which would convert the free-running arm into the teacher-forced arm. After a successful repair the agent again holds the canonical value and can rejoin the gold trajectory, which is what makes genuine recovery possible rather than merely tidy continuation.

Elected repair measures prompt position, not state. Two pilots on `allam-2-7b` (16 tasks each, 8 at depth 4 and 8 at depth 6, identical seeds and gold trajectories) offered the tool and let the model choose when to use it. With the budget rule stated as the final instruction before the first step, the model called `check_state` in 14 of 16 tasks and in every case at step 0, where the held value equals the canonical value by construction, so each call returned the number already in the prompt. With the same rule stated inside the task description and the output format stated last, it called the tool in 0 of 16. No repair landed on a diverged step under either prompt. The mechanism was not at fault: an always-repair oracle recovers the original trajectory in 100% of tasks, and a never-repair policy is step-for-step identical to the free-running arm. The task supplies no divergence signal, so an agent's only coherent policies are to ask always or never, and prompt salience selects between them. Raising the budget would select the former and measure nothing. We therefore assign the repair rather than eliciting it.

Construction. As in Appendix E, a value is corrupted at a fixed position ($j = 1$, `parity_flip`). The model then runs free, so the trajectory diverges under its own behaviour. At a fixed later position ($r = 3$) both branches receive the same `check_state` exchange: the repaired branch is handed the canonical value, the unrepaired branch the value it is already carrying, so its repair is a no-op by construction and the two message histories differ in exactly one number. The segment before the fork is executed once and inherited by both branches, so the pair is identical by construction rather than by reproducibility. Both branches then run free to the end of the task.

Scoring is at completion, not at the repaired call. Appendix E already establishes that restoring the true value repairs the *next* call ($+0.000$ against baseline at $j = 1, 3, 5$); re-measuring that would report a known result. The verdict here is `completed_original`: the branch ran every remaining step *and* finished holding the task's true final output. The constraint $r \leq d - 2$ is enforced in the harness so that at least two steps remain to be scored, which is what makes the verdict a trajectory property. A validity test fixes the distinction: a policy that makes the repaired call correctly and then drifts is scored as not recovered, where a next-call-only check would score it a success.

Table 8: Assigned repair on `allam-2-7b`: $n = 37$ pairs at each of depths 6 and 8, both complete at their allocated n. Intervals are 89% paired bootstrap intervals resampling pairs; the two branches share an inherited prefix, so the pair is the independent unit. Downstream rates cover the steps from the repair onward. Discordant counts are pairs in which exactly one branch was correct: b favours the repaired branch, c the unrepaired one.

measure	repaired	unrepaired	difference [89% interval]	b/c
depth 6 ($n = 37$)				
`completed_original`	0.054	0.000	$+0.054$ $[+0.000, +0.108]$	2/0
downstream gold agreement	0.126	0.000	$+0.126$ $[+0.063, +0.198]$	9/0
downstream conditional-on-state	0.342	0.297	$+0.045$ $[-0.018, +0.108]$	6/3
depth 8 ($n = 37$)				
`completed_original`	0.000	0.000	$+0.000$ $[+0.000, +0.000]$	0/0
downstream gold agreement	0.022	0.000	$+0.022$ $[+0.005, +0.038]$	4/0
downstream conditional-on-state	0.162	0.184	-0.022 $[-0.049, +0.011]$	0/2

Result. At depth 6, restoring the canonical state raises downstream gold agreement by $+0.126$ $[+0.063, +0.198]$, an interval excluding zero: the repair demonstrably returns the model to the canonical path. It does not measurably raise completion of the original task. That difference is $+0.054$ $[+0.000, +0.108]$, rests on two events in 37 pairs, and its interval includes zero, so we do not report it as an effect. Conditional-on-state competence is unchanged by the repair ($+0.045$

$[-0.018, +0.108]$), as expected: the model continues about as competently from a repaired state as from a corrupted one.

At depth 8 the completion measure is at the floor outright: zero of 37 repaired branches and zero of 37 unrepaired branches finish holding the task's true final output, so there is nothing for an interval to bound. The gold-agreement gain also attenuates sharply, from $+0.126$ at depth 6 to $+0.022$ $[+0.005, +0.038]$ at depth 8, and rests on four discordant pairs; we read it as a weak positive rather than a replication of the depth-6 figure. Conditional-on-state competence again shows no repair effect (-0.022 $[-0.049, +0.011]$). The attenuation has a straightforward reading: a single mid-trajectory repair leaves more remaining steps at depth 8 than at depth 6, and each is another opportunity to re-diverge, so what the repair buys decays with the distance still to run.

The dissociation is the result, and it holds at both depths. Returning to the canonical state and recovering the original goal are separable properties, and only the first is purchasable by handing the state back. We state the asymmetry carefully: the evidence for the first is statistically supported, the absence of the second is a null rather than a demonstrated zero, and a larger sample could move completion off the floor without disturbing the ordering.

The unrepaired branch completes the original task in 0 of 37 pairs at each depth, so the contrast is attributable to the repair rather than to a corrupted trajectory rejoining gold by coincidence. An oracle applying the routing rule to whatever value it holds completes 12 of 12 repaired branches and 0 of 12 unrepaired ones, so the measure is not penalising correct behaviour.

These results are from a single model. As with Appendix E, we report them as a single-model check on whether the distinction is measurable at all, not as suite-wide validation of its magnitude.

G GPT-OSS-20B NATIVE-MODE PROBE

Section 4 reports that `gpt-oss-20b` is excluded because the hosting provider frequently rejects its native tool calls under our text-only protocol. Since the same task succeeds once tool declarations are added to the request, we ran a small probe under native tool-calling mode to check whether the model is reachable and measurable at all, using the same task suite, depths, and scoring as the primary protocol.

Table 9: `gpt-oss-20b`, native tool-calling mode. Not pooled with the primary five-model results, which use the uniform (text-only) protocol.

depth d	tasks	$\bar{p}_d$	$\bar{g}_d$	L_d
1	5	1.000	1.000	0.000
2	5	0.900	1.000	-0.111
4	5	1.000	0.900	$+0.100$
6	4	0.958	1.000	-0.043
8	0	–	–	–

The run completed 19 of a planned 25 tasks before the day's token allowance was exhausted; depth 8 was not reached. Both arms sit at or near ceiling at every depth reached ($\bar{p}_d, \bar{g}_d \in [0.90, 1.00]$), the same ceiling behavior already reported for `gpt-oss-120b` and `qwen3.6-27b` in Section 5.2, more pronounced here at low n. The small negative L_d values at depths 2 and 6 are one or two discordant calls in cells of 5–24 calls, not evidence of recovery. This probe establishes that the model is reachable and scoreable under native tool-calling mode; it is too small, too ceiling-bound, and missing depth 8 (where propagation loss is largest and best resolved in this suite) to support any claim about propagation, and is not pooled with or substituted for the primary result.