Invocation-Level Reliability of Tool-Using Agents

Afiya Noorain[1]* **Subhranshu Mohanty**[1]* **Amritesh Banerjee**[2] **Abhijit Dasgupta**[3]†
[1]SAI International School, Bhubaneswar, Odisha, India
[2]University of Massachusetts Amherst, Amherst, Massachusetts, USA
[3]SP Jain School of Global Management, Mumbai, India

Abstract

Tool-using agents fail two ways: choosing the wrong tool, or forming wrong arguments, and an early failure of either kind can silently corrupt everything downstream. We measure a correct-invocation rate that separates the two, under both a clean teacher-forced context and the model's own free-running context, on five open-weight models over contamination-free multi-step tasks (depths 1–8). By depth 6, roughly 70% of a model's own clean-context capability is lost to its own earlier mistakes ($L_6 = 0.686, 0.684$). Our central finding concerns the measurement itself. **Under exact-match scoring against a fixed gold trajectory, a propagation model's severity and recovery parameters are not merely hard to estimate—they are fixed by the scoring rule.** Severity is forced to its boundary (0 of 869 poisoned steps correct); recovery is structurally unobservable (0 of 580 poisoned steps returned on-track, against an expected 0.0058 by chance). Both follow from one mechanism: post-divergence, the gold value is generated by tool constants the model never sees, so it is information the model cannot derive. A fit run anyway returns 0.92 and 0.73 for a quantity that is exactly 1.000—confident numbers for a parameter the scoring rule already determined. We give the mechanism and a remedy, conditional-on-state scoring, applied retrospectively to cached completions at zero additional cost, which un-pins severity to interior estimates excluding zero ($+0.149$, $+0.316$).

1 Introduction

A tool-using agent can reason correctly about what must happen and still fail the call, either by choosing the wrong tool or filling its arguments wrongly. Single aggregate success scores do not say which occurred, and typically reflect clean single-shot conditions rather than the multi-step, state-carrying tasks agents actually run. A model whose per-call accuracy looks fine in isolation can still fail badly in a chain once an early mistake corrupts everything depending on it.

We measure reliability at the level of the individual invocation: correct when the call both selects the right tool and supplies correct arguments, scored under a clean teacher-forced context (isolating how reliability degrades as context accumulates) and under the model's own free-running context (where an earlier mistake poisons what follows). The gap between the two is a direct signature of propagation, separable from ordinary context-length decay.

The study began as an empirical comparison and produced a measurement result instead. An interpretable severity/recovery model could not be fit to the data for a specific and general reason: the scoring rule, not the data, determines both parameters. We report the empirical findings and that result together, since the second explains why the first must be read through a fit-free metric.

Contributions.

*Equal contribution.
†Corresponding author.

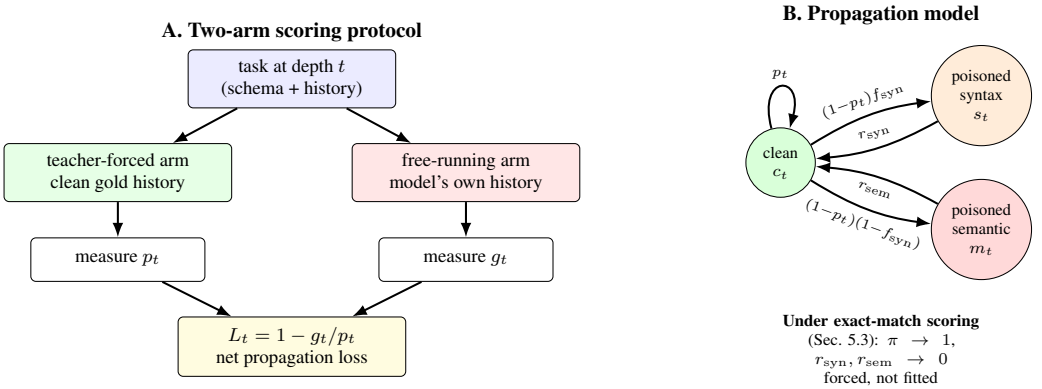

Figure 1: The two-arm scoring protocol (A) and the three-state propagation model it measures (B). Section 5.3 shows that under exact-match scoring, panel B's severity and recovery parameters are forced to the values shown, independent of the true underlying values.

1. A disaggregated correct-invocation protocol—matched clean/free-running scoring, syntactic-vs-semantic classification, and a routing-task design in which *selection* errors propagate.
2. A negative result about a scoring regime, not a model family: under exact-match scoring, severity and recovery are determined by the scoring rule, not the data. We give the mechanism and scope (any execution-match or AST-match benchmark).
3. A remedy—conditional-on-state scoring—implemented and applied retrospectively to cached completions at zero marginal cost.
4. Two per-model diagnostics invisible to aggregate accuracy: a *discrimination* statistic (one model is anti-correlated with the rule it is given) and *error position*, which bounds how much propagation a fixed-depth measurement can even observe.
5. A released pipeline with measured provider rate limits documented nowhere else.

2 Related Work

Toolformer [1] learned when and how to call an API; ReAct [2] established the reason–act–observe loop most agent frameworks still follow; Gorilla [3] first measured API-call correctness at scale, counting hallucinated calls as a distinct error.

Most large benchmarks score task success rather than per-call correctness. ToolLLM [4] and StableToolBench [5] evaluate over thousands of (simulated, cached) APIs; API-Bank [6] scores by execution match; BFCL [7] combines AST matching with executable checks; tau-bench [8] grades final environment state, where even strong models succeed on fewer than half of tasks and are inconsistent across runs. **All of these score against a fixed reference**, the property Section 5.3 shows to be limiting.

A smaller group disaggregates: RoTBench [9] separates selection, parameter identification, and content filling under increasing noise; MTU-Bench [10] reports selection and parameter accuracy without an LLM judge; FuncBenchGen [11], the closest prior work, generates contamination-free multi-step dependency-graph tasks and finds capable models carrying stale values forward as chains lengthen—though it reports task success, not a disaggregated invocation measure.

Failures sort into **selection errors** (wrong/invented/omitted tool) and **argument errors** (malformed, missing, wrong, or fabricated values). Relign [12] formalises this split and adds deferral; Hammer [13] masks function/parameter names for generalisation; representation-level detection [14] flags bad selection at inference time. **These target the rate an incorrect call is produced, not what happens to a chain once one has entered it**—the gap this work addresses.

3 Method

3.1 Definitions

A task is a sequence of calls $c_1, \ldots, c_n$ where the arguments of c_t may depend on outputs of earlier calls; dependency depth is the length of the longest such chain.

- **Local correctness** p_t: probability step t is correct given a correct upstream history of length $t-1$ (teacher-forced). Measured at every depth, so it captures context-length degradation on its own.
- **Global correctness** g_t: probability step t is correct inside the model's own free-running trajectory, where upstream inputs may already be poisoned.
- **Net propagation loss** $L_t \coloneqq 1 - g_t/p_t$.

L_t is defined purely empirically, from two measured rates, and nothing in this paper relies on any parametric relationship between L_t and a model's severity. Section 3.2 introduces a three-state model under which L_t *would* equal $\pi \cdot x_t$ for a severity parameter π and poisoned mass x_t—but Section 5.3 shows that under exact-match scoring π is forced to 1 regardless of a model's true severity, which makes that identity vacuous as an estimation tool. We keep L_t's definition and its parametric interpretation strictly separate for this reason: every empirical claim in Section 5 uses only the definition above.

3.2 A propagation model with recovery

We additionally model context as a three-state Markov process—clean, poisoned-by-syntax, poisoned-by-semantics (Figure 1B)—driven by measured p_t and measured syntactic error share $f_{\text{syn}}(t)$, with origin-specific recovery rates $r_{\text{syn}}, r_{\text{sem}}$ (syntactic failures are caught by the executor; semantic failures execute silently). A severity parameter π scales correctness while poisoned: $\rho_t = (1-\pi)p_t$. Occupancies evolve as

$$c_{t+1} = c_t\, p_t + s_t\, r_{\text{syn}} + m_t\, r_{\text{sem}}, \quad s_{t+1} = c_t\,(1-p_t)\, f_{\text{syn}}(t) + s_t\,(1-r_{\text{syn}}), \tag{1}$$

$$m_{t+1} = c_t\,(1-p_t)\,(1-f_{\text{syn}}(t)) + m_t\,(1-r_{\text{sem}}), \tag{2}$$

with $g_t = p_t\,(1-\pi\,x_t)$, poisoned mass $x_t = s_t + m_t$, so that *under this model* $L_t = \pi\,x_t$. We fit $\pi, r_{\text{syn}}, r_{\text{sem}}$ hierarchically (PyMC, NUTS), reporting $\hat{R}$, ESS, divergences, and—critically—the posterior correlation between π and each recovery rate, the diagnostic for separable identification used in Section 5.3.

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 are $(a \cdot \arg + b) \bmod M$ with per-tool constants a, b **never exposed in the schema**—the detail driving Section 5.3.

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 poisoned input is not misclassified as selection failure. **Linear tasks (control)**: order announced, argument copied verbatim. Degenerate on real models ($p_t = g_t = 1.000$, both llama models); retained as a null design but **not executed** (Section 7).

3.4 Protocol and suite

We score selection conditionally and arguments by exact match (digit-only JSON strings coerced to integers, so provider formatting is not scored as a value error); classify errors as syntactic (parse/schema failure) or semantic (executes but wrong); measure p_t/g_t as above; measure recovery from logs as the rate a poisoned context returns on-track. Both arms use the **same within-step retry budget**, so their ratio is not a retry artifact.

Five open-weight models, Groq free tier: `llama-3.1-8b-instant`, `allam-2-7b`, `qwen3.6-27b`, `gpt-oss-20b`, `gpt-oss-120b`, `llama-3.3-70b-versatile`. Greedy decoding; every response

cached by (model, calling mode, exact prompt); fixed seeds. **Design deviations, all forced by free-tier hosting**: two scale points per family rather than 3–4 (no free host offers more); no proprietary reference (our Gemini key returns 403 on `generateContent`, confirmed against raw REST); no FC-tuned-vs-base contrast (no free-hosted xLAM/Hammer variant). All model IDs were verified live before running—the Qwen2.5/Llama-3.1-instruct generation used in our original design had since been retired from this tier.

Models run **nested prefixes** of one task suite, sized per model to its own measured token allowance. Task identity depends only on $(\text{depth}, k, \text{seed})$, 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**; the analysis is a pure function of the frozen files, with nothing 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 (pilot runs consumed its daily allowance) and is excluded from all claims. The binding constraint is an undocumented daily token allowance (TPD), non-uniform across models ($5\times$ spread), readable only from 429 response bodies.

Table 2: Measured per-model rate limits, Groq free tier.

Model	TPD (measured)	RPD	TPM
`llama-3.1-8b-instant`	$>$ 500,000 (not reached)	14,400	6,000
`allam-2-7b`	**500,000**	7,000	6,000
`gpt-oss-20b`/`-120b`, `qwen3.6-27b`	200,000	1,000	8,000
`llama-3.3-70b-versatile`	**100,000**	1,000	12,000

5 Results

5.1 Propagation is large, monotone in depth, and separated from context decay

Figure 2 and Table 3 show the core measurement. p_t falls gently with depth ($0.700 \to 0.543$ on `llama-3.1-8b`, a 22% relative decline from context growth) while g_t collapses ($0.700 \to 0.174$, 76%). L_t rises monotonically with non-overlapping adjacent-depth intervals: the trend is resolved, not suggested.

$L_1 = 0.000$ **exactly on every model**: at depth 1 both arms build an identical prompt and share one cached response, so baseline purity holds by construction—the cheapest available check that the two arms are wired correctly. Step-index error counts on `llama-3.1-8b` at depth 6 show the same signature with no model at all: identical at step 0 (21/21), then the free arm rises to 40 while the teacher-forced arm stays near 20 at every later step. A flat teacher-forced profile rules out context growth as the explanation.

The model-free lag check $\Delta_1 = P(t{+}1 \text{ ok} \mid t \text{ ok}) - P(t{+}1 \text{ ok} \mid t \text{ wrong})$ is positive for every model with any errors ($+0.32$ to $+1.00$), confirming propagation with no parametric assumption; its second term is exactly zero for every model, so it reduces to $P(\text{ok} \mid \text{ok})$ here (explained in Section 5.3).

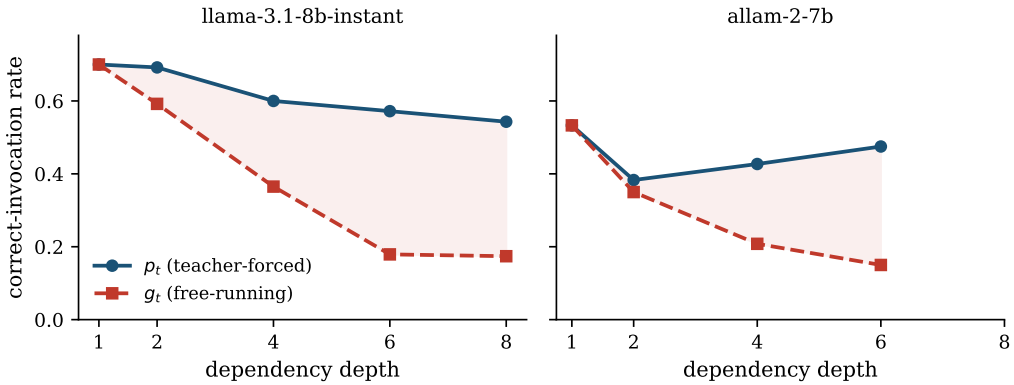

Figure 2: p_t (teacher-forced) vs. g_t (free-running) by depth. The gap (shaded) is the propagation signature this study isolates; it is absent by construction at d=1 and widens with depth on both models.

Table 3: Per-depth p_t, g_t, L_t, the two high-n models.

depth	p_t	g_t	L_t [89% CI]	n_g
`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, 0.163]	120
4	0.427	0.208	0.514 [0.429, 0.600]	260
6	0.475	0.150	**0.684** [0.624, 0.745]	240

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

Three models sit at or near ceiling (p_t = 0.865–1.000; Figure 3), so $L_t \approx 0$ there is a statement about the tasks, not robustness. On `llama-3.3-70b-versatile` at depth 4, every one of its 7 errors falls at step 2 or 3—the *final* step, where nothing downstream exists to poison, so propagation was impossible there by construction. Depth 6 gave those errors somewhere to go: $L_6 = +0.190$ $[+0.062, +0.345]$, the first propagation loss *established* in a large model here. `gpt-oss-120b` errs on a single fresh error and is not distinguishable from null at this n; `qwen3.6-27b` makes **zero errors across 298 free-arm steps**.

An error on the final step cannot propagate, so any fixed-depth measurement *under-states* loss, worst where chains are shortest; L_1=0 is the limiting case of this bias. Terminal-error share is a **property of the model**, not depth, and no aggregate exposes it: on `llama-3.1-8b` it runs consistently above the uniform expectation (0.714 vs. 0.500 at d=2; 0.197 vs. 0.167 at d=6; n = 49–320); `llama-3.3-70b-versatile`'s d=4 share of 6/7 is the vivid but weak case (n=7, CI $[0.488, 0.992]$), and it is exactly why its L_4 read as immunity. **Practical consequence**: report terminal-error share alongside L_t, and read the depth *trend*, since two models with identical per-call accuracy but different error position produce different—and non-comparable—propagation losses.

5.3 The scoring regime, not the models, fixes both parametric quantities

This is the paper's central result, and it concerns a class of measurement, not the models we happened to measure.

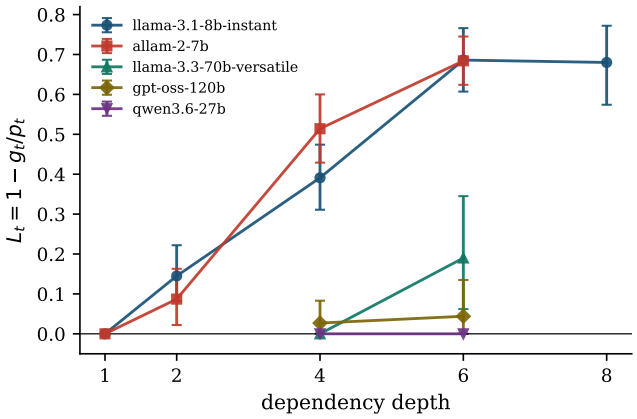

Figure 3: L_t vs. depth, all five usable models with 89% CIs. Three models sit near zero at shallow depth; `llama-3.3-70b-versatile` is the only one to leave the ceiling with an interval excluding zero, at depth 6.

Severity is forced to its boundary. $\pi := 1 - P(\text{ok} \mid \text{poisoned})/P(\text{ok} \mid \text{clean})$ is a ratio of two directly observable rates, needing no fit. Measured: **0 of 869 poisoned-context steps correct**, so $\pi = 1.000$. Support is uneven, stated as such: two models carry the claim (510, 335 poisoned steps, 95% upper bounds 0.0059, 0.0089); two are consistent but uninformative (16, 8 steps, bounds 0.171, 0.312); `qwen3.6-27b` is **undefined**, not zero—it never left a clean context, so it has no severity to report, and pooling it as 0.000 would misleadingly read as "poisoning does not hurt this model."

Recovery is structurally unobservable: $P(\text{next ok} \mid \text{this wrong}) = 0.000$, **0 of 580** poisoned steps with a successor returned on-track. Both facts share one mechanism: a poisoned step holds a non-gold value, and the gold value at step t is generated by tool constants **never exposed to the model**. After divergence, that value is information the model cannot derive; the only route back is coincidence at $\approx 1/100{,}000$ per opportunity, giving an expected **0.0058** coincidental returns over 580 chances. Observing zero is thus almost uninformative: **a model that self-corrects 20% of the time and one that never does produce identical observable data**, since neither can emit a value it has never seen. Severity is pinned at 1 and recovery unidentified for the same reason.

Substituting $\pi = 1$, $r_{\text{syn}} = r_{\text{sem}} = 0$ into Section 3.2 leaves

$$g_t = c_t\, p_t, \qquad c_t = \textstyle\prod_{j<t} p_j, \qquad L_t = x_t = 1 - c_t, \tag{3}$$

no free parameters at all: an identity in measured per-step rates, not a non-identifiable fit. Run anyway, the three-parameter posterior returns **0.92** $[0.84, 0.99]$ and **0.73** $[0.64, 0.83]$ for a quantity exactly 1.000, fitting a smooth recurrence to pooled aggregates under a prior centred at 0.5. $\hat{R} \leq 1.0005$, ESS $\geq$ 5,823, zero divergences: **clean MCMC diagnostics certify the posterior was explored, and are silent on whether it was constrained.**

Scope. This depends only on scoring against a fixed gold trajectory whose values the model cannot reconstruct—i.e. execution-match and AST-match scoring generally. **Any study fitting a severity, self-correction, or recovery parameter under such scoring reports a parameter its scoring rule has already determined.**

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 gold. Half already existed: routing-task selection was already scored both against gold and conditionally; we added the argument-side counterpart. Because the cache holds every raw completion, we applied this to data already collected—**0 API calls, 881 cache hits**—which is what makes it adoptable by any group holding cached completions.

Clean-context steps score identically under both rules (0.658 vs. 0.658), as they must, which is the implementation check. Both intervals exclude zero: poisoning *does* measurably degrade rule

Table 4: Conditional-on-state severity vs. gold-agreement severity.

model	gold-agreement π	conditional π	89% CI
`llama-3.1-8b-instant`	1.000 (boundary)	$+0.149$	$[+0.021, +0.268]$
`allam-2-7b`	1.000 (boundary)	$+0.316$	$[+0.066, +0.503]$

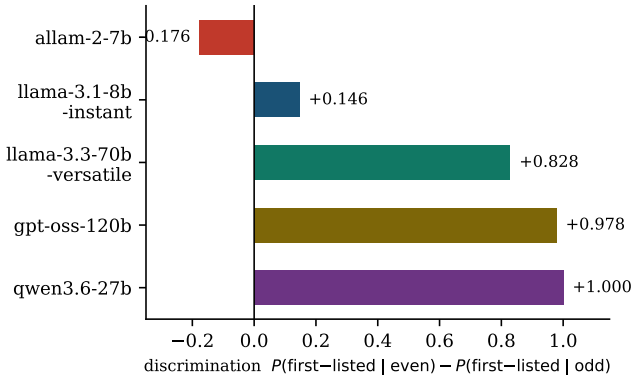

Figure 4: Discrimination—$P(\text{first-listed} \mid \text{even}) - P(\text{first-listed} \mid \text{odd})$—per model. Order matches the independently established reliability order of Section 5.1.

application, so the strongest reading ("purely information loss") is not supported. The supported reading is quantitative—gold-agreement attributes all loss to severity; conditional scoring attributes **0.15–0.32** to genuine degradation, the remainder to trajectory unreachability. We flag this as a hypothesis the data supports on two models with wide intervals, not a settled result: a correct argument here means transcribing an integer shown one turn earlier, a weak competence test, so this may not survive harder argument construction (a transformed-argument condition for this was built but **not run**; Section 7).

Cost. No single right answer per call; more implementation; looser cross-system comparability; a high correct-invocation rate stops implying end-task success. Exact-match buys comparability at the price of making severity/recovery unmeasurable; conditional scoring trades the opposite way. We report both, with gold-agreement as the primary headline.

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 *same data*, conditioned on parity, resolves discrimination across the full range (Figure 4), 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` *derives* the rule ($+1.000$, zero errors in 298 steps, explicit reasoning trace); `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 misattributed under poisoning, a contaminated syntactic-share estimator, among others). 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. Correct code, correct configuration, invalid measurement—and the second artifact is why the recovery channel was never estimable from observable data in any version of this pipeline.

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 is a controlled, synthetic-task measurement, not a validation on production tool-calling benchmarks. Integer-argument tasks give exact ground truth and let us isolate propagation cleanly, at the cost of ecological validity: whether these findings—particularly the conditional-scoring decomposition of Section 5.4, where a correct argument means transcribing a value shown one turn earlier—hold under natural-language or multi-field arguments is open. Extending this protocol to BFCL- or tau-bench-style tasks, where gold values are not perfectly opaque tool outputs, is the direct next step and would additionally test whether the identifiability result of Section 5.3 is a property of exact-match scoring in general or specific to opaque-constant synthetic tasks.

Four designed conditions were not executed within this study's compute budget, each a scope boundary rather than a missing result: a linear-task null control (Section 5.1's evidence for task-structure specificity therefore rests on internal signatures—flat teacher-forced profile, exact $L_1{=}0$, step-index divergence—rather than a direct null arm); a transformed-argument condition (leaves the error-composition hypothesis and the conditional-scoring generality question of Section 5.4 open); a presentation-order control (the parity confound of Section 5.5 is addressed by the discrimination statistic alone, not removed by design); and a calling-mode ablation (native mode was verified feasible on 5 of 6 models but not run at scale, so we make no claim about how much measured unreliability is a calling-mode artifact). We name these individually because a paper reporting effect sizes should be explicit about which arms produced them.

Other scope notes. Soft argument matching coincides with strict matching here (single-integer arguments); the opaque-feedback condition is implemented but unrun, so feedback-format hypotheses are not addressed; `distractor_level` does not vary on routing tasks, so selection rates are not comparable across the routing and linear arms; interval widths differ substantially across models by design (nested unequal n), so no scale or family contrast should be read from point estimates alone.

8 Conclusion

We measured tool-use reliability at the invocation level, separating tool selection from argument correctness and context-length decay from error propagation, and found propagation dominant on small models: by depth 6, roughly 70% of a model's own clean-context capability is lost to its own earlier mistakes.

The unexpected result concerns measurement. Under exact-match scoring against a fixed gold trajectory, a propagation model's severity and recovery parameters are not weakly identified—they are **determined in advance by the scoring rule**, because after divergence the gold value is information the model has never received and cannot derive. The recurrence collapses to an identity in measured per-step rates; a fit run anyway returns confident, wrong numbers while every convergence diagnostic reports health. The remedy is a scoring change, not a modelling one: credit a call that correctly continues from the value the model actually holds, computable retrospectively from cached completions at zero marginal cost, which moves severity from a boundary artifact to interior estimates excluding zero.

For agent benchmarks scoring against reference trajectories generally: **if the scoring rule makes the reference unreachable after a divergence, any severity or recovery parameter fit under it is a property of the scorer, not the model.**

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] S. Maekawa, H. Iso, S. Gurajada, and N. Bansal. Towards reliable benchmarking: A contamination-free, controllable evaluation framework for multi-step LLM function calling. *arXiv:2509.26553*, 2025.

[12] 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.

[13] 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.

[14] K. Healy, A. Srinivasan, N. Madathil, and Y. Wu. Internal representations as indicators of hallucinations in agent tool selection. *arXiv:2601.05214*. Presented at the AAAI 2026 TrustAgent Workshop.

[15] W. Liu, X. Huang, X. Zeng, X. Hao, S. Yu, D. Li, S. Wang, W. Gan, Z. Liu, Y. Yu, Z. Wang, Y. Wang, W. Ning, Y. Hou, B. Wang, C. Wu, X. Wang, Y. Liu, Y. Wang, D. Tang, D. Tu, L. Shang, X. Jiang, R. Tang, D. Lian, Q. Liu, and E. Chen. ToolACE: Winning the points of LLM function calling. In *ICLR*, 2025.