One Reflex, Two Answers: Why Model Judges Cannot Discriminate Localized from Distributed Causes in Agent Failure Attribution

Brian Scavotto *

Posted Date: 23 July 2026

doi: 10.20944/preprints202607.1745.v1

Keywords: LLM-as-a-judge; failure attribution; LLM agents; evaluation reliability; controlled fault injection; construct validity; benchmarking

 doi:10.20944/preprints202607.1745.v1

Article

One Reflex, Two Answers: Why Model Judges Cannot Discriminate Localized from Distributed Causes in Agent Failure Attribution

Brian Scavotto

Department of Computer Science, North Carolina Agricultural and Technical State University, Greensboro, NC 27411, USA; bascavotto@ncat.edu

Abstract

When an autonomous language-model agent fails after a long run of tool calls, a growing body of work hands the diagnosis to a large language model (LLM) that reads the trace and names the responsible step. We ask whether this LLM-as-judge attribution is reliable, and whether it can separate two different causes, a localized corruption from a single bad input and a distributed degradation across many steps. Using controlled fault injection across an arithmetic and a non-arithmetic agent task, four levels of trace completeness, and five judges, we find discrimination is weak at best. Each judge applies one propensity, blaming the step that produced the result, so its contamination and drift accuracies become two readouts of one number whose discrimination margin stays near zero. More of the trace strengthens this default rather than enabling discrimination. The behavior holds across model families and capability tiers, including two frontier reasoning models at opposite extremes of one behavioral axis, one fabricating an inconsistency and the other abstaining. A realistic six-step tool task with additional current-generation models reproduces and sharpens the finding. We give a single-propensity account of why the two per-class accuracies move in lockstep, and read the result as a reliability caution for LLM-as-judge attribution.

Keywords: LLM-as-a-judge; failure attribution; LLM agents; evaluation reliability; controlled fault injection; construct validity; benchmarking

1. Introduction

Autonomous agents built on large language models (LLMs) now execute long trajectories of tool calls and intermediate decisions, and when they fail the failure is often hard to diagnose. The execution trace an agent leaves behind is the main record available for reconstructing what went wrong. Attributing a failure to a cause matters for deciding whether to trust, repair, or retire an agent, yet a trace rarely reveals on its face which step was decisive. A method that returns a confident but wrong cause is worse than no method, since it points the fix in the wrong direction.

One response to this problem is gaining ground, using a capable model as a judge that reads the trace and names the step that caused the failure. The same models that drive agentic systems should be able to read their traces without a human writing large sets of hand-built rules, and this appeal is what made LLM-as-judge a standard evaluation tool in the first place [1]. These judges carry documented biases of their own, such as sensitivity to the order in which candidates appear [2]. Recent work nonetheless builds attributors and benchmarks on exactly this premise, and reports that current judges localize the causative step poorly, often near chance [3–6]. That work bets that as models grow stronger and traces grow more complete, the gap will close.

This study tightens the question. Two failures must be treated differently yet can look almost the same. The first is a localized corruption, in which a single bad value enters at one step and propagates, which calls for distrusting that source. The second is a distributed degradation, in which many small errors accumulate with no single decisive step, which calls for distrusting the whole run.

 doi:10.20944/preprints202607.1745.v1

Telling them apart is not the same as localizing a step, because the two demand different responses. We measure whether a model judge can make that distinction, and whether giving it more of the trace helps.

Our method is controlled fault injection, which supplies an exact answer to grade against rather than a human-annotated guess. We run an agent on two deliberately different tasks, an arithmetic order-total task and a non-arithmetic majority-vote task, and break each run in a known way, by poisoning one input, by misdirecting the agent at the start, or by spreading a small perturbation across many steps. We present the captured trace at four levels of completeness and ask five judges, spanning four model families and including two frontier models from different labs, to name the decisive step. Because we injected the fault, we know whether the judge is right.

The judges show little reliable discrimination between the two cause types. Each applies one fixed move, blame the step that produced the result, which is correct by construction when the cause is a localized corruption feeding that step and wrong when the cause is distributed. A model's accuracy on contamination and its accuracy on drift turn out to be two mechanical readouts of that single propensity rather than two independent results, and giving the judge more of the trace makes the propensity more confident, not more discriminating. A prompt that forbids the judge from recomputing the aggregate removes its false attribution on distributed failures, but the same change blinds it to the localized one, so the two cannot be fixed together. The pattern survives at the frontier and across labs. Two frontier models land at opposite extremes of a single behavioral axis, one fabricating an inconsistency to justify the step it blamed, and even catching its own miscalculation mid-rationale before blaming the step anyway, the other abstaining whenever the trace looks internally consistent, which at full detail it does, since the agent processed the corrupted value correctly. Neither separates a localized corruption from a distributed one at a level that would support reliable attribution.

The contribution is a reframing supported by measurement. The open problem is not whether more logging improves attribution, a question the recent literature has crowded, but whether a model judge can discriminate the structure of a cause at all. We show that across families, capability tiers, and both frontier labs it does so weakly at best, that the failure reduces to one model-specific axis from confident fabrication to conservative abstention, and that trace completeness moves a judge along that axis without granting discrimination. For anyone building or trusting model-as-judge attribution of agent failures, the practical warning is concrete. A judge's confidence and its accuracy on one fault class say nothing about its ability to tell that class from another. Understood as an evaluation question, this is a reliability assessment of an attribution method before it is trusted to support a conclusion, and the method does not yet clear that bar. We draw these results from constructed environments by design, since controlled ground truth requires faults we inject rather than label after the fact, and the frontier result answers the obvious objection, that this is a limitation of weaker models, by showing that the strongest models evaluated here fail on the same distinction. A realistic multi-tool environment added in revision reproduces the failure across six further current-generation models, where it sharpens into a collapse of contamination localization rather than easing.

2. Related Work

Agents, tools, and benchmarks. Autonomous agents built on the reason-and-act pattern [7] and equipped with large tool repertoires [8] now execute long, consequential trajectories, and benchmarks such as WebArena [9], GAIA [10], SWE-bench [11], and AgentBench [12] measure their capability while surfacing the multi-step failures this work sets out to attribute.

Failure attribution for LLM agents. A fast-moving body of 2025 and 2026 work formalizes failure attribution for LLM and multi-agent systems and supplies shared benchmarks. Zhang et al. [3] introduced the Who&When dataset, comprising failure logs from 127 LLM multi-agent systems with step-level annotations, and reported that the best automated method localized the decisive step at low accuracy. AgenTracer [13] built annotated failure trajectories through counterfactual replay and programmed fault injection, and Cemri et al. [4] produced the MAST taxonomy over more than

 doi:10.20944/preprints202607.1745.v1

1,600 annotated traces. Further benchmarks and localizers followed quickly, among them the human-annotated TRAIL traces, on which the strongest model scored only 11% [6], spectrum-based attribution over system logs [14], and causal-graph attribution that converts a flat trajectory into a structured graph [15]. Closest to the present question, Chen et al. [5] benchmarked attribution under full versus partial execution observability, establishing that completeness matters while stopping at that dichotomy. This work overwhelmingly proposes attributors and the data to evaluate them. The present study instead characterizes a failure mode common to these attributors, the structural tendency of a model judge to blame the step that produced the result, and the low pinpointing accuracy these benchmarks report is consistent with that bias.

LLM-as-a-judge reliability. The judges these attributors depend on are themselves the subject of a fast-growing literature, surveyed by Gu et al. [16]. That work documents how judges deviate from human preference, through order sensitivity [2], a self-preference for text they find familiar [17], and a general gap between a judge's apparent reliability and its validity [18]. These studies ask whether a judge agrees with human preference and whether it is internally consistent. We ask a different question, whether a judge can tell one causal structure from another, and we find a structural blame propensity that agreement and consistency audits do not surface, because it is correct by construction on one cause type and wrong on the other.

Agent observability and provenance. A parallel line builds the recording infrastructure attribution consumes. AgentOps [19] offers a lifecycle taxonomy of what to trace, AgentSight [20] performs boundary tracing from outside the agent at stable system interfaces using eBPF, and AgentTrace [21] instruments agents with schema-based structured logging. Provenance research extends to agents through PROV-AGENT [22], which captures agent prompts, responses, and decisions in end-to-end workflow provenance, and broader treatments establish traceability as a trustworthy-AI requirement [23,24]. These systems address what can be recorded and where the recorder sits. Our question is orthogonal. Given a recorder, how much and what must be present for an attributor to discriminate causes, and we show that beyond a point more capture amplifies a judge's default rather than improving discrimination.

Fault localization. The classic software-engineering literature is the methodological ancestor. Spectrum-based fault localization and its accuracy limits on realistic programs [25,26], trace-based root-cause localization in distributed systems [27], and studies of what execution logs retain and how they evolve [28] frame the trade between trace completeness and diagnosability, which we carry into black-box agents. What none of these supplies, and what we measure, is whether a model judge can discriminate a localized from a distributed cause, and why it cannot.

3. Materials and Methods

3.1. Overview

We measure whether a model judge can attribute the cause of an agent failure, and how that ability depends on how much of the execution trace it sees. The paradigm is controlled fault injection. We run an agent on a task, inject a fault whose true cause we control, capture the full execution trace, present that trace at varying completeness to an attribution procedure, and grade its localization against the known ground truth. Because we control the injected fault, we hold an exact answer to grade against, which human-annotated failure benchmarks generally lack. The measurement harness and analysis scripts were written with the assistance of a large language model (Claude, Anthropic), and all code was reviewed and tested by the author.

3.2. Task Environments

We use three task families behind a common interface, so the measurement machinery is environment-agnostic. The order-total environment asks an agent to compute the total of an order by looking up a record and summing its line items, generated with five to eight line items and values up to several hundred, so totals run to the thousands. The consensus environment is non-numeric, where

 doi:10.20944/preprints202607.1745.v1

the agent reports the majority recommendation among a panel of reviewers. These two are deliberately different in kind, one arithmetic and one categorical, so a shared finding is not an artifact of summation, and each is deterministic given a seed and yields fifty distinct instances per condition.

The third environment, vendor-spend, is realistic and procedurally closer to a deployed tool-agent workflow. The agent identifies the highest-spending vendor in a ledger over six heterogeneous tool steps, querying the ledger for a period, fetching the raw transactions, aggregating spend per vendor, ranking, writing the result, and reporting it. The longer and more varied tool chain tests whether a finding from the two compact tasks survives a trajectory that no longer reduces to a single arithmetic or categorical operation. Contamination inflates one transaction at the fetch step, so the localized cause enters at a read far from the reported result. Drift spreads a small increment across many of the runner-up's transactions, so it overtakes the leader with no single decisive transaction. Early error queries the wrong period. This environment is run at two hundred instances per cell, and its result is in Section 4.8.

3.3. Fault Injection

Each run is broken in one of three ways, chosen so the true cause differs in structure. Contaminated input poisons a single value the agent reads at a known step, an outlier price or a flipped decisive reviewer, so the cause is localized and the ground-truth decisive step is the read. Early error perturbs the early instruction so the agent pursues the wrong target, and the decisive step is the first. Drift applies a small perturbation across many items so that no single step is decisive, so the cause is distributed and has no single ground-truth step. Every injected run is paired with a matched clean run to confirm the injection, not incidental variance, caused the failure. Because the decisive step for contaminated input is genuinely ambiguous, where the bad value is read at one step and consumed at the next, each injection carries an explicit set of acceptable steps rather than a single index, together with a true class label scored separately. The magnitudes are fixed and reported for reproducibility. Contamination replaces one value with a clear outlier, an order line item set to fifty times the order's largest price, a single decisive reviewer flipped in a one-vote majority, or one vendor transaction inflated to about 1.3 times the leading vendor's total. Drift applies a small distributed perturbation, a three percent shift across every order line item, two reviewer flips in a three-vote margin, or ten percent of the leader's total spread across the runner-up's transactions, so no single change is decisive. For contaminated input the accepted set is the reading step and the step that consumes the value.

3.4. Trace Completeness

The agent loop records a maximal trace, capturing for each step the prompt, the model response, the tool call and its full result, the agent's context and memory state, the external state the tools exposed, and the model provenance. From this maximal trace we derive four nested completeness levels by ablation. L0 is outputs only. L1 adds tool calls and a debug-grade result summary. L2 adds the agent's internal context and memory. L3 adds the full external state and provenance. The design places the raw per-item data the agent acted on, line-item prices and individual reviewer votes, only in the external state, so it appears solely at L3, mirroring the gap between what typical agent logs retain and what causal reconstruction requires.

3.5. Attribution and Scoring

The primary attributor is a model judge. Given a task and a trace at one completeness level, it names the single decisive step and a root-cause class, using only the information present at that level, and declines when the information is insufficient. Two judge prompts are compared. The standard prompt asks the judge to identify the decisive step. The arithmetic-free prompt forbids recomputation and tells the judge to blame a step only when its recorded input is itself anomalous, otherwise to label

 doi:10.20944/preprints202607.1745.v1

the cause distributed, which controls for whether the judge's recomputation, rather than the trace, drives its attributions. A deterministic numeric-anomaly heuristic serves as a non-model baseline.

We score three quantities separately, where strict step localization requires the predicted step to equal the recorded decisive step. Tolerant step localization credits any step in the injection's acceptable set, or within a small distance of it. For a distributed cause, it also credits either a decline or a drift label. This results in reporting strict and tolerant together, exposing how much apparent error is a brittle rubric versus a genuine miss. Root-cause classification is scored against the true class, separately from localization, because a judge can localize a step while mislabeling its cause. Beyond accuracy, we read each judge's per-step prediction distribution directly, since the share of predictions that blame the result-producing step, measured identically for localized and distributed causes, is what exposes whether a judge discriminates or applies a single fixed propensity.

3.6. Judges and Statistics

Judges span families and capability tiers, comprising GPT-4.1 (OpenAI), Qwen3-235B-A22B-Instruct (Alibaba), DeepSeek V3 0324 (DeepSeek), and two frontier reasoning models, GPT 5.5 (OpenAI) and Claude (Anthropic). The realistic-environment experiment of Section 4.8 adds six current-generation models, Claude, Grok 4.20 (xAI), Gemini 3.5 (Google), Llama 4 (Meta), Mistral Medium 3.5 (Mistral), and DeepSeek V4 (DeepSeek), each run at two hundred instances per cell. All are accessed through a single OpenAI-compatible client with structured-output JSON mode to control the format-failure rate seen on lighter open-weight models, and judge temperature is set to the lowest available value. Each reported cell aggregates fifty distinct instances, one attribution per instance, so instances are independent observations and we report Wilson 95% confidence intervals on each proportion [29]. Format-failure rates are reported per model, and because such failures fall on all cause types equally, the central comparison of how a model treats contaminated versus distributed causes is unaffected by them. Beyond per-cell intervals we test the central comparison inferentially. For each judge we form the two-by-two table of predicted move, blaming the result step or not, against true cause class, contaminated or distributed, and test their association with Fisher's exact test, pool across completeness with a Cochran-Mantel-Haenszel test stratified by level, and report Cohen's h as an effect size with a bootstrap 95% interval on the difference in blame rate. The completeness effect is tested within class as a paired comparison of the blame decision at outputs-only against full detail, matched by instance, with an exact McNemar test. Fifty instances give a Wilson half-width near fourteen points on a proportion at one half, and two hundred give near seven, which sets the precision of each reported cell. We justify the sample by that interval width rather than a power calculation, since the study estimates behavior rather than testing a single planned hypothesis, and because we report many cells we read individual significance descriptively and rest the central claim on the discrimination margin across cells rather than on any one test. Table 1 lists the judges, providers, call counts, and format-failure rates.

Table 1. To ensure reproducibility and consistent access, all judge models were evaluated at the lowest available temperature. The total number of calls, N, is the product of the test instances, three fault classes, and four completeness levels. Unparseable responses were classified as format failures; because these occurred uniformly across all cause types, they were excluded from the reported data. Vendor-spend judges specify their exact OpenRouter model slug, whereas compact-task judges use standard model identifiers paired with the precise snapshot and access route detailed in the released run configuration. Gemini required an expanded output budget of 8,192 tokens to prevent its reasoning from exhausting the default cap, while all other judges operated under standard defaults. Because commercial model aliases shift over time, the released configuration documents the exact API identifier, access route, date, and request parameters for every run.

Label	Model identifier	Provider	Experiment	Date 2026	N calls	Fmt. fail.
GPT-4.1	GPT-4.1	OpenAI	compact	Jun 24	1200	0%

 doi:10.20944/preprints202607.1745.v1

Claude Sonnet 4.6	Claude Sonnet 4.6	Anthropic	compact	Jun 26	1200	0%
GPT 5.5	GPT 5.5	OpenAI	compact	Jun 26	1200	0%
Qwen3-235B-A22B	Qwen3-235B-A22B-Instruct	Alibaba	compact	Jun 25	1200	0%
DeepSeek V3 0324	DeepSeek-V3-0324	DeepSeek	compact	Jun 26	1200	20%
Claude Sonnet 4.6	anthropic/claude-sonnet-4.6	Anthropic	vendor-spend	Jun 27	2400	1%
Grok 4.20	x-ai/grok-4.20	xAI	vendor-spend	Jun 28	2400	0%
Gemini 3.5 Flash	google/gemini-3.5-flash	Google	vendor-spend	Jul 1	2400	0%
Llama 4 Maverick	meta-llama/llama-4-maverick	Meta	vendor-spend	Jun 27	2400	0%
Mistral Medium 3.5	mistralai/mistral-medium-3-5	Mistral	vendor-spend	Jun 28	2400	0%
DeepSeek V4 Pro	deepseek/deepseek-v4-pro	DeepSeek	vendor-spend	Jun 28	2400	17%

3.7. A single-Propensity Account of the Two Accuracies

The scoring makes the two headline numbers analytically dependent. On a contaminated run the accepted answer includes the result-producing step, so blaming that step scores correct. On a distributed run no single step is decisive, so blaming any step scores wrong and only a decline or a drift label scores correct. Writing p_1 for the rate at which a judge blames the result step on contaminated runs and $_{33}$ for that rate on distributed runs, tolerant contamination accuracy equals$_{11}$ and tolerant drift accuracy equals $1\ {}_{33}p_3$, so the two sum to 1 plus a discrimination margin $\delta_1 = p_3\ {}_3\ p_3$. A judge that truly separates the causes drives δ toward one, blaming the result step on the localized cause and withholding it on the distributed one. A judge that applies one propensity to both has δ near zero and a sum near one. The margin δ is thus a single measurable test of discrimination, and we report it alongside the raw accuracies as the study's primary outcome, since it collapses the two per-class accuracies into the one quantity the reliability question turns on.

4. Results

4.1. What the Headline Accuracy Hides

Read as a simple accuracy table, the standard-prompt judge looks like it improves with trace detail on contaminated input and degrades on drift. With GPT-4.1, contamination localization climbs from 0 at outputs-only to 100 at full detail (0, 54, 90, 100 in the arithmetic environment; 0, 12, 68, 100 in the consensus environment), while drift collapses over the same range (100, 66, 16, 0 arithmetic; 100, 88, 38, 0 consensus). The early-error case stays pinned at 100 throughout, because the wrong target shows up in the agent's own output and needs no deep trace to spot. Those two curves are not two results. They are one behavior seen twice.

4.2. Discrimination is Weak at Best

The discrimination margin δ makes the comparison precise. At full detail it is zero for GPT-4.1, Claude, and GPT 5.5 in both environments, 0.16 and 0.18 for Qwen and DeepSeek on the arithmetic task, and at most 0.18 anywhere, against the value near one a reliable attributor would show. Pooled across completeness with a Cochran-Mantel-Haenszel test, the association between true cause class and the blamed step is significant in a single cell, GPT 5.5 on the arithmetic task ($\chi^2 = 7.75$, $p = 0.005$),

 doi:10.20944/preprints202607.1745.v1

marginal for GPT-4.1 there (p = 0.046), and non-significant everywhere else, including every cell in the consensus environment and both Claude cells. That one clear effect does not replicate across environments, the bootstrap intervals on δ include zero for DeepSeek and for Qwen on consensus, and effect sizes at full detail are small, with Cohen's h at most 0.52. The honest reading is not that discrimination is exactly absent but that it is weak, inconsistent, and far below what an attributor could be relied on for, and that the dominant behavior is one propensity to blame the result-producing step applied almost identically to both causes (Figure 2). Because tolerant contamination accuracy equals that blame rate and tolerant drift accuracy equals one minus it, the two accuracies move together as completeness deepens, contamination rising and drift falling while their sum holds near one (Figure 1). Table 2 reports tolerant localization and the margin at full detail for all five judges.

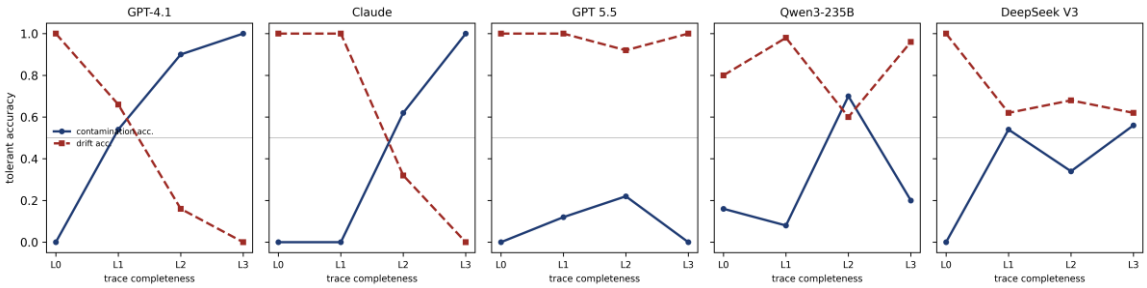


Figure 1. As trace completeness rises from outputs-only (L0) to full detail (L3), contamination accuracy rises and drift accuracy falls for the blame-prone judges, while their sum stays near one (synthetic task).

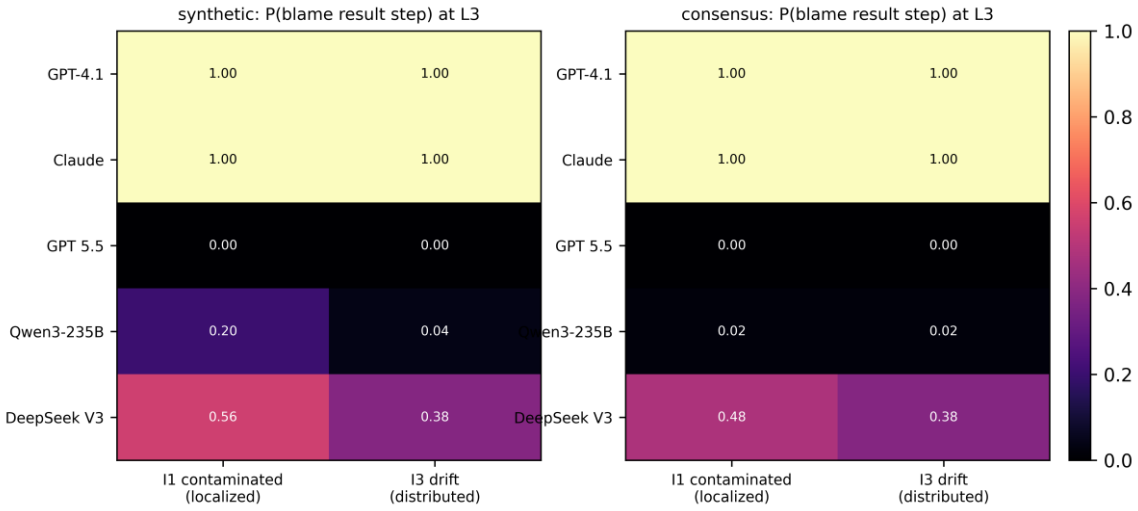


Figure 2. The rate at which a judge blames the result-producing step is nearly identical for localized (I1) and distributed (I3) causes within each model, the signature of one propensity rather than discrimination.

Table 2. Tolerant step-localization accuracy (percent) at full trace detail (L3) under the standard prompt, for the two compact environments and all five judges. Contamination, drift, and early error are the three injected fault classes. The discrimination margin is the difference between the result-step blame rates on contamination and drift, near zero when a judge applies one propensity to both. Full per-level results are in the released runs.

Model	Environment	Contamination	Drift	Early error	Disc. margin
GPT-4.1 (OpenAI)	order-total	100	0	100	0.00
GPT-4.1 (OpenAI)	consensus	100	0	100	0.00

 doi:10.20944/preprints202607.1745.v1

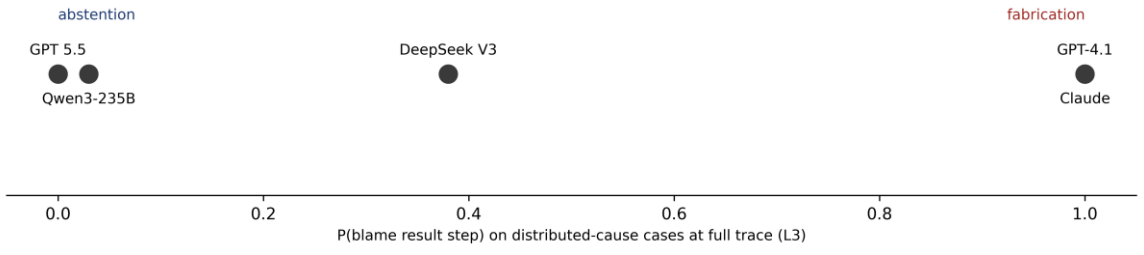
Claude (Anthropic)	order-total	100	0	100	0.00
Claude (Anthropic)	consensus	100	0	100	0.00
GPT 5.5 (OpenAI)	order-total	0	86	100	0.00
GPT 5.5 (OpenAI)	consensus	0	46	100	0.00
Qwen3-235B (Alibaba)	order-total	20	72	100	0.16
Qwen3-235B (Alibaba)	consensus	2	98	100	0.00
DeepSeek V3 (DeepSeek)	order-total	56	60	82	0.18
DeepSeek V3 (DeepSeek)	consensus	48	62	76	0.10

4.3. The Prompt Control and a Tradeoff with no Free Side

To test whether the judge's habit of recomputing the aggregate drove the behavior, we ran an arithmetic-free prompt that forbids recomputation and tells the judge to blame a step only when its recorded input is anomalous. The drift artifact disappears. In the arithmetic environment, drift attribution holds flat near the top (100, 100, 100, 98) instead of collapsing to 0. The same prompt that fixes drift cripples contamination, dropping its full-detail accuracy from 100 to 36, and the two cells do not overlap (36, interval 24 to 50, against 100, interval 93 to 100; Figure 4). The mechanism is shared, which is why the prompt cannot win on both. The judge catches contamination only by noticing a numeric inconsistency, and it fabricates a drift cause by the same act of noticing one. The judge has no other handle on a localized corruption, since the agent processes the poisoned value without complaint and the surface of a contaminated run looks like a clean one, so the recomputed mismatch is the single cue the fault leaves, and forbidding recomputation removes it.

4.4. Cross-Model Evidence, One Axis With no Exit

Five judge checkpoints across four families place the behavior on a single dimension, a propensity to blame the result-producing step, and no point on that dimension produces a model that discriminates reliably (Figure 3). GPT-4.1 and Claude sit at the high end, blaming that step in nearly every case. GPT 5.5 and Qwen3-235B sit at the low end, declining or scattering when the trace looks internally consistent. DeepSeek V3 0324 falls between them, blaming the result step on about half of contaminated runs and a third of distributed ones, though a format-failure rate near 20% blurs its estimate (Section 4.6). Within every model the contamination and drift prediction distributions stay close to identical, the signature of a judge that is not separating the two. The two ends are not a capability gap, because they hold two frontier models from two labs, one at each extreme.



 doi:10.20944/preprints202607.1745.v1

Figure 3. The five judges occupy a single behavioral axis from abstention to fabrication, with the two frontier models at opposite ends; none discriminates the two causes.

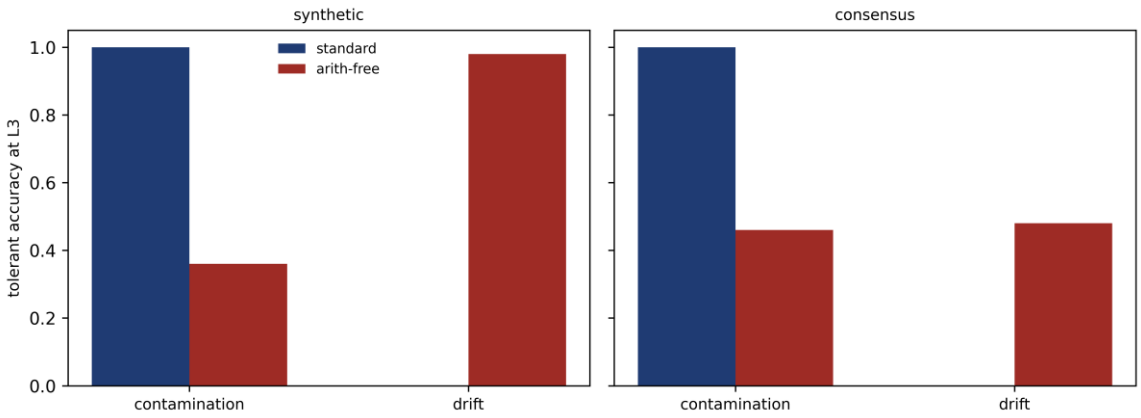


Figure 4. Forbidding recomputation with the arithmetic-free prompt removes false blame on distributed faults but also erases the signal the judge used to catch localized ones, so the tradeoff has no free side.

The completeness effect is sharp wherever the propensity is active. Comparing the blame decision at outputs-only against full detail, matched by instance, the blame rate climbs from 0 to 1 for GPT-4.1 and Claude on both cause types and in both environments (exact McNemar, $p < 10^{-15155}$), and from 0 to between 0.38 and 0.56 for DeepSeek ($p < 10^{-5}$), while GPT 5.5 holds at abstention throughout. The trace does not teach the judge to discriminate. It moves a blame-prone judge from withholding to blaming, on contaminated and distributed runs alike (Figure 1).

4.5. The Failure Holds Across the Frontier, at Both Extremes

Two frontier models, GPT 5.5 from OpenAI and Claude from Anthropic, are the strongest test of whether this is fundamental or a quirk of weaker models. Each returned clean output on all 1,200 of its calls and localized the early-error case at 100 everywhere, so neither failure is incompetence. They fail in opposite directions, and neither discriminates.

GPT 5.5 abstains. It does not carry the reflex to blame the computation step, naming that step in essentially none of the full-detail cases, and its contamination localization falls to 0 at full detail in both environments. Deeper traces make it worse on contamination, not better, which inverts the usual intuition. The reason is the honest-abstention logic Qwen first showed. At full detail the trace looks internally consistent, since the agent did process the poisoned value correctly, so the more carefully the model reasons, the more confidently it concludes that nothing went wrong.

Claude does the opposite. It blames the result step in all fifty contaminated and all fifty drift cases, fabricating the inconsistency that justifies the choice. One of its rationales is the clearest evidence in the study. On a drift case Claude wrote that the tally reported accept "when the 7 recommendations were actually 4 accept and 3 reject, wait, 4 > 3 so accept," caught its own error mid-sentence, confirmed the tally was correct, and still blamed that step. The structural prior overrode the model's own correct observation.

One frontier lab abstains into missing the localized cause, the other fabricates a cause that is not there, and the two sit at opposite ends of the same axis. Reasoning changes how a judge fails. It does not give any of the five models the ability to tell a localized corruption from a distributed one.

4.6. Localization, Classification, and a Data-Quality Note

Step localization and root-cause classification do not move together. GPT-4.1 localizes the early-error step at 100 in every condition while its class label for that same case lands in the low forties at best, so a judge can point at the right step and still misname why it failed. The two are reported as separate quantities.

 doi:10.20944/preprints202607.1745.v1

The two DeepSeek checkpoints returned empty or malformed content on a meaningful share of calls, about 20% for V3 0324, and enabling structured-output JSON mode did not reduce it. Because these failures fall on contamination and drift equally, they do not affect the central within-model comparison, but they confound DeepSeek's absolute position on the axis. We report DeepSeek with that caveat. The four clean checkpoints, GPT-4.1, Claude, Qwen, and GPT 5.5, carry the cross-model claim and the frontier result on their own. A worst-case check bounds the concern anyway. Even if every malformed DeepSeek call were assigned adversarially, all toward blaming the result step on contamination and all toward declining on drift, the reconstructed discrimination margin would shift by at most the failure rate, roughly two tenths, which leaves it far below the near-one margin a reliable attributor would show. The finding does not depend on how the malformed calls are treated.

4.7. The Shape of the Wrong Attribution

Recoding every rationale, rather than scoring it only right or wrong, shows that the failure on a distributed cause takes one of two forms, fabrication or abstention, and which one a judge chooses is what places it on the axis. When GPT-4.1, Claude, and DeepSeek blame a distributed failure on a single step, the rationale asserts a specific, invented inconsistency to justify the choice in 99, 77, and 97 percent of those cases. The judge does not merely guess a step. It manufactures a concrete and false account of what went wrong there, the kind of confident, specific cause story a downstream consumer of the attribution is most likely to accept. GPT 5.5 and Qwen instead abstain, avoiding fabrication at the cost of missing the localized cause entirely. The fabrication mode is the more hazardous of the two, because its output is indistinguishable in form from a correct attribution.

4.8. The Result Holds in a Realistic Multi-Tool Environment

On the vendor-spend environment of Section 3.7, run at two hundred instances per cell across six current-generation models from six developers, the central result holds and sharpens (Table 3, Figure 5).

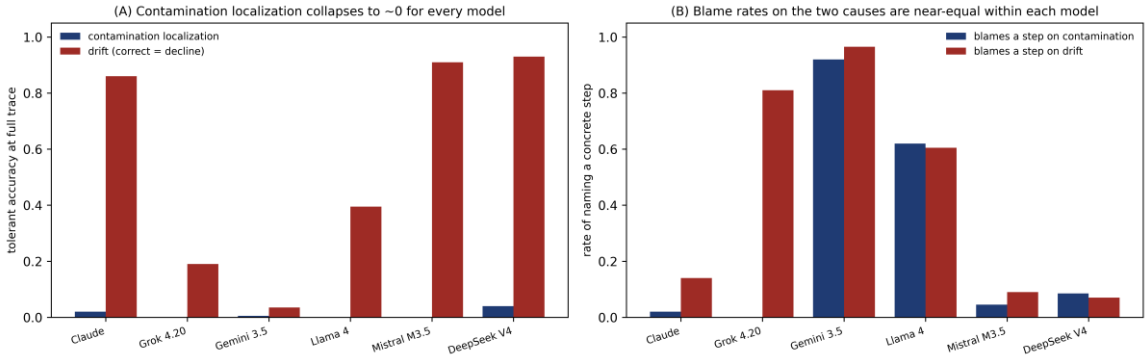


Figure 5. The realistic vendor-spend task across six current-generation models (n = 200). Contamination localization collapses to near zero for every model, and where models name a concrete step they do so more often on the distributed cause than on the localized one.

No model localizes the contaminated transaction at full trace, with tolerant contamination accuracy at most 0.04 across all six, spanning Claude, Grok, Gemini, Llama, Mistral, and DeepSeek V4. The blame-the-result-step propensity persists, but the longer trace relocates the result-producing step, the write and rank steps, away from the decisive read, so the models that blame a step, Grok, Gemini, and Llama, blame that distant step and miss the contamination, while Claude, Mistral, and DeepSeek V4 abstain instead. Neither habit is discrimination. Where the association between the true cause class and the blamed step reaches significance it is inverted rather than correct, with a concrete step named more often on the distributed cause than on the localized one (Cochran-Mantel-Haenszel odds ratios below one for Claude, Grok, Gemini, and Llama, strongly so for Grok at 0.02), and an

 doi:10.20944/preprints202607.1745.v1

inverted attributor is worse than an indifferent one. Completeness still amplifies the blame, the paired blame rate rising from outputs-only to full detail for all six models (exact McNemar $p < 0.001$). The realistic environment does not rescue attribution. It splits the models into blamers and abstainers on the same axis, and neither tells the two causes apart.

Table 3. Vendor-spend environment, tolerant accuracy (percent) at full trace detail (L3) and the discrimination margin, for six current-generation models at two hundred instances per cell. Contamination localization collapses for every model, while the margin stays near zero or turns negative, the negative values marking judges that blame a concrete step more readily on the distributed cause than on the localized one.

Model	Contamination	Drift	Early error	Disc. margin
Claude (Anthropic)	2	86	100	−0.12
Grok 4.20 (xAI)	0	19	100	−0.81
Gemini 3.5 (Google)	0	4	100	−0.04
Llama 4 (Meta)	0	40	100	+0.02
Mistral Medium 3.5 (Mistral)	0	91	100	−0.04
DeepSeek V4 (DeepSeek)	4	93	100	+0.01

4.9. The Propensity is Robust to Temperature and Sensitive to Trace Order

Two ablations on a representative blamer, Llama on the vendor-spend task at fifty instances per cell, probe whether the propensity is an artifact of the decoding or the prompt layout. Raising the judge temperature from zero to 0.7 changes nothing material, the blame rate on contamination moving from 0.58 to 0.56 and on drift from 0.72 to 0.68, so the behavior is not a product of greedy decoding. Shuffling the order in which the steps are presented to the judge, while preserving each step's own index, is more revealing. Under natural order the judge parks its blame on the final step, the report, naming it in fifty-six of sixty-five blamed cases. Under shuffle that concentration dissolves and the blamed index scatters across every position, which a judge reasoning about a step's causal role would not do, since the index travels with the step. The attribution tracks where a step sits in the prompt rather than what it did, and the scatter even lifts contamination accuracy spuriously to 74 percent as randomly placed blame lands by chance inside the accepted set. The one quantity that does not move is the discrimination margin, which stays near zero under natural order, shuffle, and raised temperature alike (Table 4). Neither decoding setting nor trace ordering makes the judge distinguish a localized cause from a distributed one.

Table 4. Ablations on Llama at fifty instances per cell on the vendor-spend task, at full trace detail. Blame rates are the fraction of cases in which the judge named a concrete step; contamination and drift are tolerant accuracy. Temperature leaves the blame rate essentially unchanged, shuffling the trace order scatters the blamed step and lifts contamination accuracy spuriously, and the discrimination margin stays near zero throughout.

Condition	Blame I1	Blame I3	Contamination	Drift	Margin
Natural order, T=0	0.58	0.72	0	28	−0.14
Natural order, T=0.7	0.56	0.68	0	32	−0.12
Shuffled order, T=0	0.74	0.74	74	26	0.00

5. Discussion

 doi:10.20944/preprints202607.1745.v1

The practical message is narrow and firm. A model judge's confidence on one fault class, and its accuracy on that class, say nothing about its ability to tell that class from another. Across five checkpoints from four developers, every judge was dominated by a single propensity to blame the step that produced the result, and that one move drove both its apparent skill on contaminated input and its apparent failure on drift. Reading those two numbers as separate competencies, which a benchmark that reports per-class accuracy invites, mistakes one behavior for two. Treating a single accuracy number as evidence of a capability it does not isolate is a construct-validity failure of the kind benchmark critiques have long warned about [30], and closing such gaps is the aim of a maturing evaluation science for generative systems [31]. The discrimination margin we can measure is small, and in one model on one task it is not quite zero, but it never approaches the level a reliable attribution would need, so the practical conclusion stands.

This reframes what the recent surge of agent-attribution work is measuring. Benchmarks such as Who&When [3], MAST [4], and the full-trace benchmark of Chen et al. [5] report that current judges localize the decisive step poorly. Our results give a mechanism for that poor performance rather than another number. The judge is not failing to find the cause so much as declining to look for the right kind of cause, because it has one structural answer it reaches for regardless of what the trace contains. Methods that add more trace, in the spirit of richer observability [19,20] or fuller provenance [22,23], make that answer more confident without making it more discriminating. The classic fault-localization literature anticipated the shape of this trade, that more execution information does not monotonically improve diagnosis [25,26], and our result is the agentic, black-box version of it.

A head-to-head comparison against the published attributors is not the right test, and we say why rather than leave it implicit. Who&When [3], AgenTracer [13], MAST [4], and TRAIL [6] each pair a proposed attributor with its own annotated dataset and report localization accuracy on naturally occurring failures, where the true cause is labeled after the fact and the cause structure is not controlled. Our claim is not that one attributor beats another but that the model judge underneath all of them carries a structural propensity that a controlled localized-versus-distributed contrast exposes and an after-the-fact benchmark cannot isolate. Running those systems on our injected tasks would measure the same judges we already test, and running our judges on their datasets would reintroduce the missing ground truth the injection harness was built to supply. The low localization numbers those benchmarks report are consistent with the propensity we measure, which is the comparison that matters.

The arithmetic-free control isolates the costly half of the mechanism. When the judge recomputes the aggregate it does so unreliably and manufactures an inconsistency that justifies blaming the producing step. Forbidding recomputation removes the false attribution on distributed faults, yet the same prohibition removes the only signal the judge had for catching a localized one, so the two cannot be repaired together by prompt design. The Claude transcript makes the bias visible in a single line. The model recomputed a categorical majority, noticed mid-sentence that the tally was correct, and still blamed the tally step. A correct observation lost to a structural prior is not a calibration problem that a temperature or verbosity fix addresses. It is a property of how these models attribute. An ablation sharpens this. The blame is unmoved by temperature but relocates when the trace order is shuffled, concentrating on the final position under natural order, so it tracks where a step sits in the prompt rather than the step's causal role.

The cross-lab frontier split carries the durability claim. A reader could dismiss a single-model result as idiosyncrasy and a single-lab result as a house style. Two frontier reasoning models, from OpenAI and Anthropic, sit at opposite extremes of the same axis, one fabricating a cause and the other abstaining past the real one, and neither discriminates. The failure does not track capability, since the weakest and the strongest models we tested both fail, and it does not track the failure direction, since reasoning ability moves a model only toward a more articulate version of its default. The implication for attribution is direct. A model judge can supply a confident attribution and a plausible rationale while showing at most a weak and unreliable ability to distinguish the two cause

 doi:10.20944/preprints202607.1745.v1

structures a diagnosis most needs to separate, and the rationale can contradict the model's own correct intermediate observation. The realistic environment widens that durability further, since six current-generation models from six developers reproduce the failure on a six-step heterogeneous task, where it deepens into a collapse of contamination localization rather than easing with a more lifelike trace. The issue is not merely low accuracy but the absence of a validated error rate for the discrimination the method is asked to make. A method that cannot reliably distinguish a localized corruption from a distributed process failure should not be treated as an independent basis for attribution without corroborating analysis.

One assumption underlies all of this, that the trace the judge reads is faithful to what the agent did. That need not hold. A trace can be altered after the fact, or shaped during execution by prompt injection into a tool result [32–34], and a judge that anchors on the final position and manufactures a specific cause is easy to steer, since whoever controls what sits in the blamed position controls the attribution. Trace faithfulness therefore bounds what any judge built on top can be trusted to say, and a model judge adds a second point of failure rather than removing the first.

6. Limitations

The environments are constructed rather than drawn from production logs, which is the price of controlled ground truth, since naturally occurring failures do not arrive labeled with their true cause. We answered the most common form of this objection, that the tasks are too simple, by adding the realistic vendor-spend environment of Section 4.8, a six-step heterogeneous tool chain on which the result not only held but sharpened across six further current-generation models. What remains open is the step from a constructed realistic task to a production deployment or a public agent benchmark such as WebArena or ToolQA, where the cause cannot be injected and ground truth must be annotated after the fact. We expect the propensities to persist, since they held from the compact tasks through the realistic one and across the capability range, but we do not claim to have measured a production system. We use "realistic" here to mean procedurally closer to a deployed tool-agent workflow, not naturally occurring production evidence.

The two compact-task experiments use fifty instances per cell, with Wilson and bootstrap intervals on every proportion and the central within-model comparison tested directly. The realistic environment raises this to two hundred per cell and adds six current-generation model families, so the broad claim no longer rests on the smaller samples alone. The two frontier models in the compact tasks were run once per environment, so their exact percentages there are point estimates, though their qualitative placement is unambiguous. The DeepSeek V3 checkpoint in the compact tasks returned malformed output on roughly one call in five even with structured output requested, which blurs its absolute position, but because those failures fall on both cause types equally they do not disturb the central within-model comparison. The temperature and trace-order ablations rest on a single model at fifty instances, so they illustrate the propensity's robustness and its positional character rather than establishing either across the full panel.

The fault taxonomy is three classes, not the full space of agent failures. Contaminated input, early misdirection, and distributed drift were chosen because they separate a localized from a distributed cause cleanly, which is the distinction under test, not because they exhaust how agents fail. A judge that cannot make this distinction may still be useful for failure modes outside our taxonomy, and our claim is scoped to the distinction we measured.

7. Conclusion

Using a model to attribute the cause of an agent failure is appealing and increasingly common, and it rests on an assumption the evidence does not support. Asked to separate a single bad input from a slow distributed degradation, none of the evaluated judges did so reliably, from open-weight models through two frontier reasoning models from different labs, and through six further current-generation models on a realistic task. Each was dominated by one propensity to blame the step that

 doi:10.20944/preprints202607.1745.v1

produced the result, correct by accident for the localized cause and wrong for the distributed one, and more of the trace strengthened that propensity rather than granting discrimination. The behavior spans the capability range and both frontier labs, which makes it a property of model-as-judge attribution rather than a weakness of any one system. Any use of these judges to attribute agent failures should carry that caveat plainly. An attribution and a fluent rationale are not evidence that the model distinguished the cause it named from the one it did not.

Author Contributions: B.S. was solely responsible for conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing of the original draft, review and editing, and visualization. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The measurement harness, generated task instances, injected faults, per-call prediction logs, analysis and figure scripts, run configuration, and judge prompts are openly available in Zenodo at https://doi.org/10.5281/zenodo.21148349. Every reported figure and statistic can be reproduced from the released runs. Appendix A gives the exact judge prompts and Appendix B a sample trace.

Acknowledgments: During the preparation of this manuscript, the author used Claude (Anthropic) to assist in creating and debugging the measurement and analysis code. The author reviewed and edited all code and takes full responsibility for the content of this publication.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. Judge prompts

The standard judge prompt, given as the system message.

```
You are a failure analyst for autonomous AI agents. You are given a task and an
execution trace. Identify the single decisive step that caused the failure,
using ONLY the information present in the trace. If the failure has no single
decisive step because the cause is spread across many steps, set decisive_step
to null and root_cause_class to "drift". If the trace does not contain enough
information to tell, set decisive_step to null and root_cause_class to
"unknown". Do not guess a step. Respond with JSON only, no prose, in exactly
this shape: {"decisive_step": <integer step index or null>, "root_cause_class":
"contaminated_input|early_error|drift|unknown", "rationale": "<one sentence>"}
```

The arithmetic-free control prompt is identical except that it forbids recomputation.

```
You are a failure analyst for autonomous AI agents. You are given a task and an
execution trace. Do NOT perform arithmetic and do NOT recompute any totals or
sums. Attribute the failure to a single step only if that step's recorded input
is itself clearly anomalous relative to comparable values in the trace. If the
degradation is spread across many steps with no single anomalous input, set
decisive_step to null and root_cause_class to "drift". If you cannot tell from
the trace, set decisive_step to null and root_cause_class to "unknown". Respond
with JSON only, no prose, in exactly this shape: {"decisive_step": <integer step
index or null>, "root_cause_class": "contaminated_input|early_error|drift|
unknown", "rationale": "<one sentence>"}
```

 doi:10.20944/preprints202607.1745.v1

Each call appends the task and the trace at the given completeness level in a fixed user message.

```
TASK: <task text>
OUTCOME: the run failed (the reported result was wrong).
TRACE (one object per step, only the fields captured at this level are present):
<steps serialized as JSON>

Return the JSON now.
```

Appendix B. Sample execution trace

A vendor-spend run at full detail records six steps. The first two are shown, with the raw per-transaction data present only at L3.

```
step 0 query_ledger(ledger=RPT-1000, period=Q3)
tool_result: {period: Q3, num_transactions: 23}
external_state: {transaction_ids: [TX-SOY-0, TX-UMB-0, ...]}
step 1 fetch_transactions(ids=[...])
tool_result: {num_records: 23}
external_state: {transactions: [{txn: TX-SOY-0, vendor: Soylent,
amount: 812.44}, ...]} # present only at L3
step 2 total_by_vendor() -> {totals: {Soylent: 9210.5, Umbrella: 8944.1, ...}}
step 3 rank_vendors() -> {top_vendor, top_total, ranking}
step 4 write_file(summary.txt, "Top vendor: <name>")
step 5 final report
```

Contamination injects at the fetch step (step 1), with accepted steps 1 to 3; drift perturbs many transactions with no single decisive step; early error queries the wrong period.

References

1. Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. Advances in Neural Information Processing Systems, 36, 2023.
2. Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), 2024.
3. S. Zhang, M. Yin, J. Zhang, J. Liu, Z. Han, J. Zhang, B. Li, et al. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In Proceedings of the 42nd International Conference on Machine Learning (ICML), 2025a. arXiv:2505.00212.
4. Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, et al. Why do multi-agent LLM systems fail? In Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2025. arXiv:2503.13657.
5. M. Chen, J. Wang, F. Mu, Y. Wang, Z. Liu, H. Feng, et al. Seeing the whole elephant: a benchmark for failure attribution in LLM-based multi-agent systems. arXiv preprint arXiv:2604.22708, 2026.
6. Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. TRAIL: trace reasoning and agentic issue localization. arXiv preprint arXiv:2505.08638, 2025.
7. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR), 2023.
8. Yujia Qin, Shengding Liang, Yining Ye, et al. ToolLLM: facilitating large language models to master 16000+ real-world APIs. In International Conference on Learning Representations (ICLR), 2024.

 doi:10.20944/preprints202607.1745.v1

9. Shuyan Zhou, Frank F. Xu, Hao Zhu, et al. WebArena: a realistic web environment for building autonomous agents. In International Conference on Learning Representations (ICLR), 2024.
10. Gr'egoire Mialon, Cl'ementine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In International Conference on Learning Representations (ICLR), 2024.
11. 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 (ICLR), 2024.
12. Xiao Liu, Hao Yu, Hanchen Zhang, et al. AgentBench: evaluating LLMs as agents. In International Conference on Learning Representations (ICLR), 2024.
13. G. Zhang, J. Wang, J. Chen, W. Zhou, K. Wang, et al. AgenTracer: who is inducing failure in the LLM agentic systems? arXiv preprint arXiv:2509.03312, 2025b.
14. Y. Ge, L. Xie, Z. Li, Y. Pei, and T. Zhang. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. arXiv preprint arXiv:2509.13782, 2025.
15. Y. Wang, W. Wu, J. Wang, and Q. Wang. From flat logs to causal graphs: hierarchical failure attribution for LLM-based multi-agent systems. arXiv preprint arXiv:2602.23701, 2026.
16. Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, et al. A survey on LLM-as-a-judge. The Innovation, 70 (6):0 101253, 2026. https://doi.org/10.1016/j.xinn.2025.101253.
17. Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in LLM-as-a-judge. arXiv preprint arXiv:2410.21819, 2024.
18. J. D. Norman, M. U. Rivera, and D. A. Hughes. Reliability without validity: a systematic, large-scale evaluation of LLM-as-a-judge models across agreement, consistency, and bias. arXiv preprint arXiv:2606.19544, 2026.
19. Liming Dong, Qinghua Lu, and Liming Zhu. AgentOps: enabling observability of LLM agents. arXiv preprint arXiv:2411.05285, 2024.
20. Yusheng Zheng, Yanpeng Hu, Tong Yu, and Andrew Quinn. AgentSight: system-level observability for AI agents using eBPF. In Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems, 2025.
21. A. AlSayyad, K. Y. Huang, and R. Pal. AgentTrace: a structured logging framework for agent system observability. arXiv preprint arXiv:2602.10133, 2026.
22. Renan Souza, Amal Gueroudji, Stephen DeWitt, Timothy Poteet, Brian Etz, Daniel Rosendo, et al. PROV-AGENT: unified provenance for tracking AI agent interactions in agentic workflows. In 2025 IEEE International Conference on e-Science (eScience), 2025. https://doi.org/10.1109/escience65000.2025.00093.
23. Maraal Mora-Cantallops, Salvador S'anchez-Alonso, Elena Garc'ia-Barriocanal, and Miguel-Angel Sicilia. Traceability for trustworthy AI: a review of models and tools. Big Data and Cognitive Computing, 50 (2):0 20, 2021. https://doi.org/10.3390/bdcc5020020.
24. Karl Werder, Balasubramaniam Ramesh, and Rongen Zhang. Establishing data provenance for responsible artificial intelligence systems. ACM Transactions on Management Information Systems, 130 (2), 2022. https://doi.org/10.1145/3503488.
25. Vidroha Debroy and W. Eric Wong. A consensus-based strategy to improve the quality of fault localization. Software: Practice and Experience, 430 (8):0 989--1011, 2011. https://doi.org/10.1002/spe.1146.
26. Simon Heiden, Lars Grunske, Timo Kehrer, Fabian Keller, Andr'e van Hoorn, Antonio Filieri, and David Lo. An evaluation of pure spectrum-based fault localization techniques for large-scale software systems. Software: Practice and Experience, 490 (8):0 1197--1224, 2019. https://doi.org/10.1002/spe.2703.
27. Guangba Yu, Zicheng Huang, and Pengfei Chen. TraceRank: abnormal service localization with dis-aggregated end-to-end tracing data in cloud native systems. Journal of Software: Evolution and Process, 2021. https://doi.org/10.1002/smr.2413.
28. Weiyi Shang, Zhen Ming Jiang, Bram Adams, Ahmed E. Hassan, Michael W. Godfrey, Mohamed Nasser, and Parminder Flora. An exploratory study of the evolution of communicated information about the execution of large software systems. Journal of Software: Evolution and Process, 260 (1):0 3--26, 2013. https://doi.org/10.1002/smr.1579.

 doi:10.20944/preprints202607.1745.v1

29. Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association, 220 (158):0 209--212, 1927. https://doi.org/10.1080/01621459.1927.10502953.
30. Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. AI and the everything in the whole wide world benchmark. In Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2021.
31. Laura Weidinger, Inioluwa Deborah Raji, Hanna Wallach, Margaret Mitchell, Angelina Wang, Olawale Salaudeen, et al. Toward an evaluation science for generative AI systems. arXiv preprint arXiv:2503.05336, 2025.
32. Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: compromising real-world LLM-integrated applications with indirect prompt injection. In Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023. https://doi.org/10.1145/3605764.3623985.
33. Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: benchmarking indirect prompt injections in tool-integrated large language model agents. In Findings of the Association for Computational Linguistics: ACL 2024, 2024. https://doi.org/10.18653/v1/2024.findings-acl.624.
34. Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tram`er. AgentDojo: a dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2024.