Reliable Agents for Systems:
Problem Statements, Proofs of Concept, and Preliminary Results on Verification-Bounded Autonomy

Hannah B. Pasandi
University of California, Berkeley h.pasandi@berkeley.edu

July 2026 Working draft, the reliability and verification thread of *The Agent Stack*

Abstract

AI agents are graded on accuracy, but deployment is decided by reliability, and reliability is decided by verification. This proposal states the problem from four vantage points, each with evidence, a proof of concept, and preliminary results. The unifying claim is an autonomy bound derived from a simple verification law: the degree of unsupervised operation an agent can safely be granted is limited by the false-accept rate f of the best verifier available in its domain, not by model quality. Today that verifier is a language model or a human. Published measurements show language-model judges plateau near chance on exactly the failures that matter, the most trusted test suites reject correct solutions at double-digit rates, and in operations settings the fraction of agent actions gated by a sound check is, in the words of the researcher who built the leading SRE benchmark, zero. We propose to measure the verifier constants, to measure the verifiable slice, and to demonstrate formal-specification-gated actuation on a real failover interface, with a first submission target of the NeurIPS 2026 Verify-Agents workshop (deadline August 29, 2026). Every number in this document is either derived from a stated closed form or carries a verified citation.

1 Overview: the claim, and where we start

Agentic AI is crossing into production while its measurement remains pre-scientific. The first field study of deployed agents, MAP, surveyed 306 practitioners and analyzed 86 deployed or pilot systems: 61.3 percent of deployed agents were never compared against any baseline, 74.2 percent are evaluated by manual human inspection, and practitioners rank reliability and resource cost as their largest challenge [16]. Adoption is projected to grow roughly eightfold in a single year [8]. The gap between what we ship and what we can certify is widening now.

This document is the reliability and verification thread of our broader Agent Stack program. It makes one theoretical move and four empirical ones. The theoretical move is small and deliberately so. Let a generator succeed per attempt with probability p, and let a verifier accept correct work at rate t and incorrect work at rate f. Retrying until acceptance gives

$$\Pr[\text{correct} \mid \text{accepted}] = \frac{pt}{pt + (1-p)f}, \qquad \mathbb{E}[\text{attempts}] = \frac{1}{pt + (1-p)f}. \tag{1}$$

Requiring a target correctness γ among accepted actions and solving for the admissible false-accept rate yields the **autonomy bound**:

$$f \;\leq\; \frac{pt\,(1-\gamma)}{\gamma(1-p)}. \tag{2}$$

Read operationally: raising autonomy is a verification problem, not a capability problem. We are explicit about lineage: Equation 2 is a modern, agent-specific instance of imperfect-coverage reliability modeling

from the dependability literature, where f plays the role of one minus detection coverage [3, 2]. The contribution is not the algebra. It is naming the constant that governs agent autonomy, measuring it across real verifiers, and building the systems that drive it toward zero where it matters.

Where we start. Three steps, in order, chosen so that each produces a figure and each de-risks the next.

1. **Weeks 1 to 2, PoC-1: measure the verifier constants.** Estimate (t, f) for LLM judges and for unit-test oracles on released agent logs and human-audited labels [13, 15]. Output: the first comparable verifier-ladder table and a (t, f) scatter with iso-autonomy curves from Equation 2.
2. **Weeks 2 to 4, PoC-2: measure the verifiable slice.** Census the action space of SREGym [6] and classify each action class by whether a sound cheap check exists or could be written as a formal operations specification. Output: the first quantitative estimate of the slice, with a growth path.
3. **Weeks 3 to 7, PoC-3: demonstrate specification-gated actuation.** Gate a real PostgreSQL failover action on its formal preconditions and measure f with and without the gate under injected faults. Output: the headline before-and-after bar.

Submission target: NeurIPS 2026 workshop *Who Verifies the Agents?*, deadline August 29, 2026, four to nine pages, double blind, non-archival, dual submission welcome. If PoC-3 slips, PoC-1 plus PoC-2 make a complete four-page demo-track paper. The work then feeds the flagship paper's verification experiment and the AgentMetrics workshop.

2 Four problem statements

Each vantage point below states a problem, gives the verified evidence that it is real, proposes a proof of concept, and reports what we already know analytically or from the literature.

2.1 PS1. The verifier is unmeasured (measurement vantage)

Problem statement. *The constants that govern agent reliability, the true-accept and false-accept rates of the verifiers actually in use, have never been measured comparably across verifier classes, even though Equation 1 makes them the bottleneck parameters.*

Why it is real. What evidence exists is alarming and scattered. On multi-turn agent trajectories, no configuration across five LLM judges and five prompt strategies exceeds 0.65 AUROC at detecting false successes, and the same judges fall to 0.54 AUROC on API-call traces, near chance [1]. In a production multi-turn transaction setting, an LLM judge surfaced 2 of 9 human-confirmed defect patterns, and its operational gate flagged zero of one hundred rounds in a batch where humans confirmed 23 distinct defects [30]. On hard response pairs, the best judge reaches 64 percent accuracy [25]. A manual audit of math graders found false-positive rates up to 15 percent for one open judge, while another judge traded a 0 percent false-positive rate for a 10 percent false-negative rate [20], showing (t, f) varies by design in exactly the way Equation 1 prices. Judges are also adversarially gameable: single-token attacks induce high false-accept rates across model families [31]. And the ladder's supposedly sound middle rung is not sound: OpenAI's own construction of SWE-bench Verified flagged 61.1 percent of raw samples for unit tests that may unfairly reject valid solutions [14], and its February 2026 audit of the hard subset (27.6 percent of the 500 tasks) found at least 59.4 percent have flawed tests that reject functionally correct solutions [15]. Both directions of error are live, and nobody has put the tiers on one axis.

Proof of concept (PoC-1). Use the released HAL logs, 21,730 agent rollouts with ground-truth outcomes [13], plus the human-audited SWE-bench Verified labels [15]. For each rollout, run candidate judges and record acceptance; estimate (t, f) with Wilson intervals per judge, per domain; estimate the unit-test oracle's

own (t, f) against the human audit. One to two weeks of work; the data are public.

Preliminary results. Analytically, the stakes are already fixed. Take a weak generator, $p = 0.3$, $t = 0.9$. A sound verifier ($f = 0$) yields 100 percent correctness among accepted outputs at 3.7 attempts. $f = 0.1$ yields 79 percent at 2.9 attempts. $f = 0.5$ yields 44 percent at 1.6 attempts, worse than a coin flip. The perverse incentive is visible in those numbers: the worse verifier is the cheaper one, so systems tuned on cost drift toward unsoundness. And the bound is brutal at high assurance: at $\gamma = 0.99$ the admissible false-accept rate is $f \leq 0.004$, which no measured judge approaches; published judge behavior sits one to two orders of magnitude above it [1, 30, 25]. The expected PoC-1 figure, a (t, f) scatter with iso-autonomy curves, will make that gap visible on one axis.

Related work. JudgeBench and the judge-robustness line measure judges against each other [25, 31]; the reliability-science program defines metric families [18]; neither connects measured (t, f) to an autonomy criterion. That connection is the contribution.

2.2 PS2. The interface is unwritten (systems vantage)

Problem statement. *Agents act through management interfaces whose correctness preconditions are undocumented, so no verifier, human or machine, can check what was never written down; the false-accept rate has a floor set by missing specifications, not by judge quality.*

Why it is real. Two results from the systems community establish this. First, failures concentrate at boundaries between independently correct systems. A study of cross-system interaction failures found that 20 percent of 55 public cloud incidents at major providers were interaction-induced, assembled a 120-case open-source dataset, and showed 51 percent occur in the data plane and 32 percent in the management plane, with 82 percent of data-plane cases rooted in metadata discrepancies; notably, 40 percent of fixes simply add condition checking or error handling, which is precondition enforcement by another name [26]. Second, the preconditions themselves are missing. A new position paper from the same group, shared with us in preparation [29], documents a PostgreSQL failover interface with three preconditions, backup readiness, replication-lag limits in asynchronous mode, and sync-standby status in synchronous mode, none documented in the manual, each discovered only after a real failure caused by a Kubernetes operator; the group's public NSDI 2026 study independently establishes that failover operations require operators to respect fine-grained preconditions on application-internal state [10]. Their answer is the formal operations specification (FOS): a state-machine model of the managed system plus invariants, written in the P language, encoding pre- and postconditions per management interface, built from source code rather than documentation. Writing FOSes for PostgreSQL, CockroachDB, Solr, Kafka, and RabbitMQ proved affordable, partially automatable with LLM assistance, and immediately useful: checking four Kubernetes operators against them found 39 new bugs [29].

Proof of concept (PoC-3). An agent proposes a failover on a live PostgreSQL cluster managed by an open-source operator (CloudNativePG or the Zalando postgres-operator). A checker validates the three formal preconditions before execution. Inject faults that violate each precondition, using the fault-injection tooling the group has already open-sourced [23, 9], and measure the false-accept rate of the gate versus an LLM-judge gate versus no gate. Three to five weeks. Expected figure: $f_{\text{gated}} \approx 0$ versus $f_{\text{ungated}} > 0$ under injected precondition violations, the headline empirical result.

Preliminary results. The three preconditions are already enumerated and mechanically checkable from Patroni state, so the gate is implementable without new theory. The 40-percent-of-fixes-add-condition-checking statistic [26] is the field independently converging on the same remedy after failures; we propose applying it before.

Related work. Sieve, Acto, and Anvil test and verify the controllers themselves [23, 9, 24], with compo-

sitional control-plane verification now emerging [5]. The FOS reframing, and ours, is complementary: the contract sits between the managed system and whatever manages it, including an agent. To our knowledge no one has gated a learning-based agent's actions on an FOS and measured the resulting f reduction.

2.3 PS3. The human is the verifier (autonomy vantage)

Problem statement. *Today's deployed agents are gated by human review, which caps autonomy at human bandwidth; removing the human without a measured, sufficient substitute verifier is unsafe by Equation 2, yet no one has measured which agent actions admit a substitute at all.*

Why it is real. MAP finds 74.2 percent of production systems evaluated by manual inspection and 68 percent of agents halting within ten steps for review [16]. In operations specifically, the builder of the leading live SRE benchmark reports that gating today is entirely model self-assessment, with no dry-run and no invariant check in the loop, that 99 percent of SRE agents perform diagnosis only, and that the fraction of real agent actions currently admitting a sound cheap check, the verifiable slice, is zero (T. Xu, personal communication, July 2026). The benchmark itself, SREGym, is a live environment over real cloud-native stacks with 90 problems, fault injection across layers, and agent actuation through configuration and code changes, scored by an LLM judge plus outcome checks [6], which means even our best evaluation harness sits on the highest-f rung of the ladder.

Proof of concept (PoC-2). Enumerate SREGym's action space and, for each action class, label it into three bins: a sound cheap check exists today; a sound check could be written as an FOS; no known sound check. Weight bins by empirical action frequency across the 90 problems. Two to three weeks, ideally co-labeled with the benchmark's authors. Output: the first measured estimate of the verifiable slice

$$S = \sum_{c \in \text{action classes}} w_c \, \sigma_c, \qquad \sigma_c \in \{0,1\},\ w_c = \text{frequency of class } c, \tag{3}$$

turning an expert's "zero today" into a number with a growth path: the slice grows exactly as FOS coverage grows.

Preliminary results. Two analytical pieces are ready. First, a cost-weighted refinement answers the scope question raised in our discussions: the bound binds where errors are costly and relaxes where they are not. Setting the required correctness from error costs, $\gamma(C) = C_{\text{fa}}/(C_{\text{fa}} + C_{\text{rej}})$ with C_{fa} the cost of a false accept and C_{rej} the cost of a wrongful rejection, recovers a Neyman-Pearson-style criterion; as C_{fa} grows, $\gamma \to 1$ and admissible $f \to 0$, which is why operations demand sound gates while low-stakes drafting does not. Sequential testing gives the adaptive-budget version [28]. Second, verifier composition: stacking independent verifiers in series multiplies both rates, $f_\wedge = \prod_i f_i$ and $t_\wedge = \prod_i t_i$, the cascade structure [27], so a cheap high-t judge in front of an expensive near-sound check buys soundness at tolerable cost; the independence caveat is load-bearing, since judges sharing a base model correlate, and PoC-1 will report joint rates empirically rather than assume the product.

Related work. Scalable-oversight work studies supervising stronger models with weaker ones [4] but does not state a false-accept autonomy criterion; the reliability-metrics program [18] defines dimensions without the bound; ITBench and SREGym supply environments [12, 6] without slice measurement.

2.4 PS4. The effect is unchecked (actuation vantage)

Problem statement. *Verification today reads what the agent says; safety depends on what the agent does. No deployed system checks, before execution, that a proposed action preserves the managed system's invariants, and most managed systems cannot even express those invariants.*

Why it is real. The question that must be answered is the one put to us directly: how would you know

the effect (T. Xu, personal communication, July 2026). The classical machinery exists: a gate checks the Hoare triple $\{P\}\,a\,\{Q\}$, preconditions before firing and postconditions against observed telemetry [11, 7]. The FOS supplies P and Q for management interfaces [29]. Two consequences follow. Soundness of P is the whole game: an incomplete precondition silently reintroduces f, which is why specification is hard and why it should follow source code, not documentation. And Q must be observable: a postcondition the agent cannot read from telemetry cannot be enforced, which ties verified actuation to the observability agenda of our broader program. Existing guardrails do something different: GoEX validates post-facto and relies on undo [17], Progent confines privileges rather than effects [21], and ToolEmu evaluates risk in an emulated sandbox, itself with a measured 31 percent false-positive rate on identified failures and severe-risk behavior in 23.9 percent of cases for the safest agent tested [19]. At the sound end, closed-loop formal checking is possible where formal artifacts exist: Clover accepts up to 87 percent of correct Dafny programs with zero false accepts on adversarial incorrect ones [22], and verified controllers carry liveness guarantees by construction [24]. The gap is the middle: pre-facto, invariant-grounded gating of agent actions on real systems.

Proof of concept. PoC-3 above is the demonstration; the research program behind it is the systems contribution: certificate-carrying actuation, in which an agent emits an action together with a machine-checkable claim that the action satisfies the FOS preconditions, checked by a monitor the agent cannot bypass. This needs system support by design, which is precisely the expert assessment of the problem's difficulty and value (T. Xu, personal communication, July 2026).

Preliminary results. The FOS experience shows the specification side is tractable: five systems specified, operator models largely automatable with LLM assistance, 39 real bugs found by checking operators against the specifications [29]. Our contribution formalizes what the gate buys: within FOS coverage, $f \to 0$ for the covered class, so by Equation 2 autonomy inside the verifiable slice is safe at any γ, and outside it the human stays. That is the honest shape of self-controllable agents: total autonomy nowhere, measured autonomy exactly where the specification reaches.

Related work. Positioning in one sentence each: against GoEX, we gate before effects rather than undo after [17]; against Progent, we check semantic invariants rather than privileges [21]; against Anvil, we certify the manager's actions at run time rather than verify the controller's code offline [24]; against judge-based scoring [6], we replace the highest-f rung with a sound one where the slice permits.

3 The verifier ladder, with measured constants

Table 1 collects the published constants that PS1 will make comparable. They are order-of-magnitude evidence from heterogeneous settings, which is exactly the problem PoC-1 fixes.

4 Roadmap, venues, and team

Roadmap. Week 0: circulate this document to the two collaborators named below; read-back on the FOS paper; set up the OpenReview submission. Weeks 1 to 2: PoC-1, verifier constants (figure: (t, f) scatter with iso-autonomy curves). Weeks 2 to 4: PoC-2, verifiable-slice census on SREGym (figure: stacked action-frequency bars in three bins). Weeks 3 to 7: PoC-3, FOS-gated failover under fault injection (figure: f with and without the gate). August 29, 2026: submit to the NeurIPS 2026 Verify-Agents workshop, full paper if PoC-3 lands, four-page demo with PoC-1 and PoC-2 otherwise; the workshop is non-archival with dual submission welcomed, so the results feed the flagship measurement paper and the October 9 HotMobile submission without conflict. The three experiments also instantiate experiment E3 of the flagship program.

Verifier tier	Published constant	Source
LLM judge, agent trajectories	$\leq$0.65 AUROC detecting false success; 0.54 on API traces	Advani et al. [1]
LLM judge, production gate	2 of 9 defect patterns surfaced; 0 of 100 rounds flagged against 23 confirmed defects	Zhang et al. [30]
LLM judge, hard pairs	best model 64 percent accuracy	JudgeBench [25]
LLM judge, math grading	false-positive rate up to 15 percent (audited)	[20]
Judge under attack	single-token prompts induce high false accepts	[31]
Unit tests as oracle	61.1 percent of raw tasks flagged unfair; audit: $\geq$59.4 percent of hard subset reject correct solutions	OpenAI [14, 15]
Emulated sandbox	68.8 percent of flagged failures valid; 23.9 percent severe-risk rate for safest agent	ToolEmu [19]
Formal consistency check	up to 87 percent acceptance of correct programs, zero false accepts on adversarial incorrect	Clover [22]
Verified controller	property holds by construction	Anvil [24]

Table 1: The verifier ladder as the literature currently measures it. False-accept behavior improves by roughly two orders of magnitude from judge tiers to formal tiers, which under Equation 2 is the difference between supervised and unsupervised operation at high assurance.

Why this order. PoC-1 needs only public data and produces the paper's framing figure, so it de-risks everything in two weeks. PoC-2 requires labeling but no infrastructure. PoC-3 is the only one with systems risk, so it starts early but is not on the critical path for a submittable paper.

Team (tentative; listed as targets, not commitments). Measurement and analysis: this proposal's author. Systems and environments: the SREGym and FOS group (UIUC, Toronto, Wisconsin), whose benchmark, specifications, and fault-injection tooling PoC-2 and PoC-3 build on [6, 29, 9, 23]. Verified actuation and interface extraction: Rishabh Iyer (UC Berkeley), whose performance-contracts and interface line is the closest methodology for extracting sound machine-checkable contracts from systems code. Program analysis and test generation: Koushik Sen (UC Berkeley), co-author of MAP [16]. Formal tier: the Clover and Anvil authors for the proof-carrying rung [22, 24]. Runtime substrate: the GoEX group (UC Berkeley) for actuation plumbing and undo as defense in depth [17].

5 Honesty and limits

Four commitments, stated up front. First, Equation 2 is elementary algebra and a restatement of 1969-era coverage theory [3, 2]; we claim the naming, the measurement program, and the systems consequence, not the inequality. Second, we are not the first to gate agent actions; GoEX, Progent, and ToolEmu precede us [17, 21, 19]; we claim pre-facto invariant-grounded gating tied to a formal specification, with the resulting f reduction and slice size measured. Third, the FOS paper is unpublished; until it is public we cite the group's NSDI 2026 study [10] for the public evidence and mark the FOS framing as private communication. Fourth, the bound assumes verifier errors independent of the retry loop and a stationary (t, f); adaptive adversaries [31] and distribution shift violate this, which is why PoC-1 reports conditions, not just point estimates, and why every law in our broader program states its falsifier. If measured judge f turns out lower than the literature suggests, or the verifiable slice turns out near zero even under generous FOS assumptions, those are reportable findings that bound the agenda, and we will report them.

References

[1] Advani et al. From confident closing to silent failure: Judging LLM judges on agent trajectories, 2026. arXiv:2606.09863.

[2] T. F. Arnold. The concept of coverage and its effect on the reliability model of a repairable system. *IEEE Trans. Computers*, 22(3):325–339, 1973.

[3] W. G. Bouricius, W. C. Carter, and P. R. Schneider. Reliability modeling techniques for self-repairing computer systems. In *Proc. 24th National Conference of the ACM*, pages 295–309, 1969.

[4] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023. arXiv:2312.09390.

[5] Zhizhen Cai, Nikhil Date, Jiawei Tyler Gu, Cody Rivera, Tej Chajed, Oded Padon, Tianyin Xu, and Xudong Sun. Compositional verification of cluster control planes. In *ACM SOSP*, 2026. To appear.

[6] Jackson Clark, Yiming Su, Saad Mohammad Rafid Pial, Yifang Tian, Lily Gniedziejko, Hans-Arno Jacobsen, Yinfang Chen, and Tianyin Xu. SREGym: A live benchmark for AI SRE agents with high-fidelity failure scenarios, 2026. arXiv:2605.07161; system at github.com/SREGym/SREGym.

[7] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4), 1971.

[8] Gartner, Inc. Gartner predicts 40 percent of enterprise apps will feature task-specific AI agents by 2026, up from less than 5 percent in 2025, 2025. Press release, August 26, 2025.

[9] Jiawei Tyler Gu, Xudong Sun, Wentao Zhang, Yuxuan Jiang, Chen Wang, Mandana Vaziri, Owolabi Legunsen, and Tianyin Xu. Acto: Automatic end-to-end testing for operation correctness of cloud system management. In *ACM SOSP*, 2023. DOI 10.1145/3600006.3613161.

[10] Jiawei Tyler Gu, Zhen Tang, Yiming Su, Bogdan A. Stoica, Xudong Sun, William X. Zheng, Yue Zhang, Akond Rahman, Chen Wang, and Tianyin Xu. Who watches the watchers? on the reliability of softwarizing cloud application management. In *USENIX NSDI*, 2026.

[11] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10), 1969.

[12] Saurabh Jha et al. ITBench: Evaluating AI agents across diverse real-world IT automation tasks, 2025. arXiv:2502.05352; ICML 2025.

[13] Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S. Siegel, et al. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. In *ICLR*, 2026. arXiv:2510.11977; 21,730 rollouts, released logs.

[14] OpenAI. Introducing SWE-bench verified, 2024. openai.com; 61.1 percent of raw SWE-bench samples flagged for unit tests that may unfairly reject valid solutions; 68.3 percent filtered out.

[15] OpenAI. Why SWE-bench verified no longer measures frontier coding capabilities, 2026. openai.com, February 23, 2026; audited hard subset (27.6 percent of 500 tasks): at least 59.4 percent have flawed tests that reject functionally correct solutions.

[16] Melissa Z. Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, et al. Measuring agents in production, 2025. arXiv:2512.04123.

[17] Shishir G. Patil, Tianjun Zhang, Vivian Fang, Noppapon C., Roy Huang, Aaron Hao, Martin Casado, Joseph E. Gonzalez, Raluca Ada Popa, and Ion Stoica. GoEX: Perspectives and designs towards a runtime for autonomous LLM applications, 2024. arXiv:2404.06921.

[18] Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of AI agent reliability, 2026. arXiv:2602.16666.

[19] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *ICLR*, 2024. ToolEmu; arXiv:2309.15817.

[20] Setlur et al. Synthetic data generation and multi-step RL for reasoning, 2025. arXiv:2504.04736; manual audit of grader errors, N=100 per judge.

[21] Shi et al. Progent: Programmable privilege control for LLM agents, 2025. arXiv:2504.11703.

[22] Chuyue Sun, Ying Sheng, Oded Padon, and Clark Barrett. Clover: Closed-loop verifiable code generation, 2024. arXiv:2310.17807; SAIV 2024.

[23] Xudong Sun, Wenqing Luo, Jiawei Tyler Gu, Aishwarya Ganesan, Ramnatthan Alagappan, Michael Gasch, Lalith Suresh, and Tianyin Xu. Automatic reliability testing for cluster management controllers. In *USENIX OSDI*, 2022.

[24] Xudong Sun, Wenjie Ma, Jiawei Tyler Gu, Zicheng Ma, Tej Chajed, Jon Howell, Andrea Lattuada, Oded Padon, Lalith Suresh, Adriana Szekeres, and Tianyin Xu. Anvil: Verifying liveness of cluster management controllers. In *USENIX OSDI*, 2024. Jay Lepreau Best Paper Award.

[25] Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. Judgebench: A benchmark for evaluating LLM-based judges. In *ICLR*, 2025. arXiv:2410.12784.

[26] Lilia Tang, Chaitanya Bhandari, Yongle Zhang, Anna Karanika, Shuyang Ji, Indranil Gupta, and Tianyin Xu. Fail through the cracks: Cross-system interaction failures in modern cloud systems. In *EuroSys*, 2023. DOI 10.1145/3552326.3587448.

[27] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[28] Abraham Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2), 1945.

[29] Jinlang Wang, Jiawei Tyler Gu, Ruize Tang, Tej Chajed, Tianyin Xu, and Xudong Sun. Formal operations specifications for reliable system management. Position paper, in preparation. Shared with the author by private communication, July 2026. Not publicly available; do not cite without permission, 2026.

[30] Zhang, Wang, and Lei. Catching one in five: LLM-as-judge blind spots in production multi-turn transaction agents, 2026. arXiv:2606.10315.

[31] Zhao et al. One token to fool LLM-as-a-judge, 2025. arXiv:2507.08794.