

# Designing for Doubt: The Case for Informed Abstention in Autonomous Agents

Victor Ojewale, Suresh Venkatasubramanian

Center for Tech Responsibility, Brown University  
{victor\_ojewale,suresh}@brown.edu

## Abstract

As large language models gain tool access and are deployed as autonomous agents capable of editing records, executing transactions, and modifying infrastructure, we still evaluate them based on the sole metric of task completion. We argue that this evaluation focus constitutes a systematic design failure. Benchmark scoring, product metrics, and default deployment configurations all reward agents for proceeding *even when they lack the inputs, evidence, or authorization required to do so safely*. We call this phenomenon **compliance bias**.

This paper makes three contributions. First, we show how compliance bias is embedded in the evaluation regimes that currently shape agent development: prominent benchmarks either penalize agents for pausing or fail to measure whether pausing was appropriate. Second, we introduce the **Informed Abstention Framework**, which reconceptualizes abstention not as a failure mode but as a structured capability: a precondition-aware pause that blocks the next tool call, names what is missing, and routes to a concrete recovery action. Third, we specify what informed abstention requires in deployment, arguing that runtime enforcement, calibrated guard mechanisms, and auditable trace generation should become standard properties of agentic system design rather than optional additions. We perform a preliminary evaluation of our approach across 144 scenarios and seven model families. Our results show that runtime enforcement achieves 87.5–91% hazardous-action blocking and 75–92% usability on authorized scenarios, that compliance bias takes two structurally opposite forms across model families, and that the safety–usability tradeoff is tunable rather than fixed.

## 1 Introduction

The deployment of tool-using large language models as autonomous agents marks a qualitative shift in what AI systems can do and what it means for them to fail (Yao et al. 2023; Park et al. 2023; Wang et al. 2024). An agent that navigates a web browser, calls an enterprise API, or executes infrastructure commands is not producing text for human review. It is taking actions whose consequences extend into the world and may be difficult or impossible to reverse (Anthropic 2026; Google 2026; OpenClaw contributors 2026). A deleted database record, a submitted regulatory form, or

a triggered payment cannot be retracted by editing the next token.

This operational character of agent failure has not yet been adequately reflected in how agents are built or evaluated. The dominant evaluation paradigm measures task success as the fraction of assigned tasks an agent completes, the number of steps it takes to reach the goal, or the accuracy of its final answer (Liu et al. 2025; Zhou et al. 2024; Mialon et al. 2023; Jimenez et al. 2024). Under this paradigm, an agent that guesses a missing parameter and happens to complete the task scores well, while an agent that pauses to ask whether the parameter is correct may score poorly, or may not be counted at all. The implicit message is that agents that proceed are good and agents that pause are defective.

**Our Work.** We use the term **compliance bias** to describe the incentive for agents to proceed even when proceeding is not justified. We first outline how compliance bias is embedded in current agent evaluation regimes. Next, we introduce the idea of *informed abstention*, a precondition-aware agent pause that blocks the next action, identifies what is missing, and finds a concrete recovery action before continuing. We illustrate how agent infrastructures need to be instrumented to enable informed abstention, and support this with preliminary empirical evidence that our proposed framework can actually achieve the goals of safe performance with high utility.

## 2 The Agentic Failure Mode

In a conversational setting, an LLM failure is typically informational, the model gives incorrect advice, hallucinates a fact, or misunderstands a question. These failures are visible and recoverable; the user reads the output, notices the error, and either corrects the model or discards the response without any external state changing. In an agentic setting, failure is operational. The model produces actions whose effects persist beyond the conversation, so a failure that would be a correctable misstatement in a chatbot becomes an unrecoverable state change when the agent has tool access. This distinction is not merely one of severity but it is unique, and it requires different design responses.

## 2.1 What Makes Agentic Failure Distinctive

Three properties of agentic action combine to make operational failures uniquely difficult to manage.

**Irreversibility.** Many actions an agent might take are difficult or impossible to reverse. A deleted record, a submitted form, a sent message, or a committed financial transaction may have downstream consequences that persist regardless of what the agent does next (Ruan et al. 2024; Yuan et al. 2024). In multi-step workflows, an early irreversible action can constrain or determine the space of available corrections, so the cost of a wrong action is not simply the action itself but the entire downstream trajectory it sets in motion.

**Compounding uncertainty.** Agent workflows are usually multi-step by design. At each step, the agent receives a new observation and decides on a next action, and uncertainty about the correct action at step  $k$  propagates forward. If the agent makes an unverified assumption at step two, subsequent steps build on that assumption until the agent may be operating on a foundation of accumulated inference with no grounding in verified state (Sumers et al. 2024). A chatbot can hedge, but an agent that has already taken three actions cannot.

**Invisibility of the decision to proceed.** In human decision-making, the choice to proceed despite uncertainty is itself a decision that can be examined and challenged. When an agent proceeds, it typically does so silently, leaving no record of what it did not know, no log of the assumption it made to fill a gap, and no moment at which a human could have intervened (Ojewale, Suresh, and Venkatasubramanian 2026; Dong, Lu, and Zhu 2024) and as such invisible decisions are difficult to audit or correct.

## 2.2 What the Failure Literature Documents

The failure modes these properties produce are well-documented across independent empirical programs. Ruan et al. (2024) introduce ToolEmu, an LM-emulated sandbox spanning 36 toolkits and 144 test cases, and find that even the safest evaluated agent fails 23.9% of the time, with failures spanning financial loss, data deletion, and privacy exposure. Agents fabricate tool inputs when required information is absent, make unauthorized binding commitments, and proceed past ambiguous instructions rather than seeking clarification, even in settings where user intent is benign but details are omitted. Their framework explicitly excludes adversarial intent, making their findings directly relevant to the routine deployment failures that compliance bias produces.

Microsoft’s AI Red Team identified twenty-six distinct failure modes across deployed agentic systems (Bryan et al. 2025). Several are directly traceable to the proceed-when-uncertain default, including the “misinterpretation of instructions” failure in which an agent deletes an entire database table when asked to “get rid of” a specific record, and the “excessive agency” failure in which an HR agent terminates an employee when asked only for advice on managing underperformance.

Xie et al. (2025) document that tool calls which are individually safe can produce harmful outcomes when

chained, because no individual step checks whether the preceding step succeeded before committing to the next. Andriushchenko et al. (2025) benchmark susceptibility to explicitly malicious agentic requests and find that many leading models comply without any jailbreaking required. Zhang et al. (2025) provides a structured benchmark across 8 safety risk categories and finds that none of 16 leading LLM agents achieves a significant safety score, including in non-adversarial conditions.

Taken together, this literature establishes that unsafe proceeding is both common and consequential. What it does not provide is a unified account of why: none of these works names the evaluation incentive structure that makes proceeding the correct local strategy for any agent optimized under current benchmarks, nor do they specify what a correct pause should look like or what it should do.

## 3 The Origins and Entrenchment of Compliance Bias

Compliance bias does not emerge by accident in current agent design. Rather, it originates in training, is reinforced by benchmark scoring, and is entrenched by the product metrics that govern deployment decisions. In what follows, we provide evidence for each of these claims.

### 3.1 Training Regimes Encourage Compliance

Before an agent encounters a benchmark, its disposition toward compliance or caution has already been shaped by training. Sharma et al. (2025) provide direct empirical evidence that RLHF-trained models exhibit sycophancy: they tailor outputs to match the user’s apparent preferences even when this requires producing incorrect or unsupported claims. The mechanism is straightforward since human raters tend to prefer confident, task-completing responses over hedged or uncertain ones, and the reward model learns this preference.

This training-level disposition becomes a system-level problem when the model is deployed as an agent with tool access. A sycophantic chatbot produces agreeable text, but a sycophantic agent produces agreeable actions, actions that may be irreversible and whose agreeableness is no guarantee of correctness. The connection between sycophancy in training and compliance bias in deployment is direct but underexamined which is the same reward signal that teaches a model to agree with a user’s stated preference also teaches it to proceed with a task even when the conditions for safe proceeding are not met.

Alternative training approaches offer partial mitigation. Constitutional AI (Bai et al. 2022) introduces self-critique as a training signal, and direct preference optimization (Rafailov et al. 2024) simplifies the RLHF pipeline in ways that may reduce some forms of reward hacking. Neither approach, however, explicitly addresses the multi-step, action-oriented structure of agent workflows and both optimize for individual output quality rather than for the disposition to pause when the next action is not yet justified. The training-level roots of compliance bias remain largely unaddressed.

### 3.2 Task-Completion Benchmarks Do Not Value Abstention

Under task-completion scoring, an agent facing an ambiguous task with a missing required parameter must choose between pausing to ask (costing turns and risking non-completion) and guessing (potentially scoring credit). The expected value of guessing is weakly better than asking under most current scoring regimes.

AgentBench (Liu et al. 2025) evaluates agents across eight environments and uses outcome-based scoring throughout. In the database environment specifically, success is measured by comparing the hash of the final table state against a gold standard; an agent that guesses a missing record ID and happens to produce the correct final state receives the same score as one that verified the correct ID before acting, and the scoring mechanism cannot distinguish between them. WebArena (Zhou et al. 2024) evaluates agents by comparing the final state of the web environment against a programmatically checked expected outcome; an agent that spends additional turns on intermediate verification rather than proceeding directly risks exceeding the step limit and receiving no credit, regardless of whether its caution was warranted. GAIA (Mialon et al. 2023) evaluates agents solely on whether their final answer matches the ground truth via quasi exact match; intermediate reasoning and verification steps are neither scored nor penalized, so the only signal the benchmark produces is whether the agent got the right answer in the end.

Kapoor et al. (2025) introduce HAL and observe that current web agent graders assign the same score of zero to an agent that abstains and to one that leaks a user’s credit card, collapsing outcomes whose real-world costs differ by orders of magnitude. Zhu et al. (2026) propose the Agentic Benchmark Checklist (ABC) and show that a trivial empty-response agent scores 38% on  $\tau$ -bench (exceeding GPT-4o), because the grader accepts ground-truth matches regardless of whether any reasoning occurred. Both works sharpen how task completion is measured, but neither introduces abstention as a labeled ground-truth behavior nor provides metrics for evaluating whether a given pause was warranted.

### 3.3 Product Metrics Entrench Compliance in Deployment

Evaluation incentives do not exist in isolation; they shape product metrics, which in turn shape deployment decisions. The most common production metrics for agent systems, including task success rate, task completion rate, and turn efficiency, operationalize exactly the same incentive as the benchmarks. An agent that asks more questions will appear to have lower task success rate and worse turn efficiency, regardless of whether the questions were warranted.

Rabanser et al. (2026) evaluate fourteen models over eighteen months and find that compliance violations, cases where agents take actions without required authorization or past defined operational boundaries, are among the most persistent reliability gaps even as raw accuracy improves substantially. Their mapping of three high-profile deployment failures to reliability dimensions is instructive and in each

case examined the agent was judged capable by internal assessments before deployment, and the failure arose from proceeding past an unmet precondition that no evaluation metric was designed to catch. Capability gains do not translate automatically into reliability gains.

## 4 Related Work

**Abstention as competence.** The selective prediction literature establishes that abstention is a legitimate model behavior rather than a defect, framed as a predictor that declines on low-confidence inputs can achieve higher accuracy at lower coverage, and this tradeoff is principled rather than arbitrary (Geifman and El-Yaniv 2017, 2019; Kadavath et al. 2022). This framing is the right starting point, but it was developed for single-step classification and generation tasks and does not transfer directly to agentic settings, where the decision to abstain at step  $k$  must account for accumulated uncertainty from prior steps, the irreversibility of actions already taken, and the cascading consequences of being wrong. Wen et al. (2025) survey the broader literature and find abstention unsolved across methods, domains, and evaluation settings. No existing abstention framework addresses the multi-step, action-oriented structure of agentic deployment, and that gap is part of what motivates the present argument.

**Enforcement-layer mechanisms.** Costa et al. (2025) introduce FIDES, which assigns integrity labels to block execution when untrusted data flows into privileged tool calls. Summers, Mohammed, and Wu (2025) take a related approach through data-flow controls, observing that overly conservative policies risk introducing friction users did not intend. Adam et al. (2026) propose a hybrid approach that combines offline static analysis of tool effects with fine-grained runtime sandboxes, enabling policies that adapt dynamically to user intent rather than relying on static global restrictions. All three build principled enforcement mechanisms, but none addresses benchmark design or provides composite metrics for characterizing the safety–usability tradeoff.

**Designed friction and human oversight.** HCI research provides a complementary foundation. Bainbridge’s ironies of automation (Bainbridge 1983) show that removing humans from the loop degrades the skills and attention needed to catch errors when automation fails. Cox (Cox et al. 2016) formalizes this insight as microboundaries: small, deliberate interruptions in automated workflows that preserve human oversight without requiring full manual control. Natali et al. (Natali et al. 2025) demonstrate that designed friction, placed at moments of genuine uncertainty, produces better alignment between automated behavior and human intent than either full automation or full manual control. We draw on this tradition directly: informed abstention is a microboundary that fires at the specific moments where proceeding without grounding would be dangerous, not a general slowdown.

**Over-Refusal as the Recognized Failure Mode** A growing body of work studies over-refusal as a genuine failure

mode in its own right. OR-Bench (Cui et al. 2025) constructs a large-scale benchmark of safe prompts that safety-aligned models are likely to refuse, documenting the trade-off between toxic content rejection and unnecessary refusal of innocuous requests. XSTest (Röttger et al. 2024) identifies what it calls exaggerated safety behaviours, cases where models refuse clearly safe prompts because they share surface features with unsafe ones. Both papers are explicit that the problem they are studying sits inside a genuine tension between helpfulness and harmlessness, and OR-Bench notes directly in its future work that refusal is a false binary and that more nuanced definitions are needed. We are not contesting any of this as over-refusal of legitimate requests is a real failure mode and measuring it systematically is valuable work. The problem is that the evaluation landscape has developed tools for one direction of error and not the other. A benchmark that measures how often a model refuses safe requests cannot tell us how often a model proceeds with requests it should have refused, and without a symmetric measure of under-refusal, there is no way to assess whether refusal was appropriate in a given case. The field can currently tell us when agents pause too much but it has no systematic way to tell us when they proceed with insufficient grounding.

## 5 The Informed Abstention Framework

We are now ready to introduce the framework we study in this paper. Firstly, we define **informed abstention** as *a structured pause inside an agent workflow that occurs when the next external action is not yet justified*. The pause has three properties that distinguish it from a generic abstention or a silent failure.

It is **precondition-aware**: the pause is triggered by a specific identifiable condition such as a missing required input, an unverifiable state, or an absent authorization. The triggering condition is not inferred from the model’s confidence in the abstract but derived from the requirements of the specific operation the agent is about to execute.

It is **functional**: the pause blocks the next tool call but does not terminate the task. Instead it routes the interaction to one of three concrete recovery actions, covering clarification, bounded verification, or handoff, depending on which precondition has failed and what recovery is possible.

It is **traceable**: the pause generates a structured record explaining what condition was unmet, what recovery action was taken, and what the outcome was. This record is auditable by deployment teams and, where relevant, by human overseers.

This makes informed abstention part of competent delegation rather than a failure of task completion. The agent is not refusing the task; it is refusing to take the next step until the conditions for safe execution of that step are met.

### 5.1 Three Conditions That Trigger Informed Abstention

In practice, the conditions that justify an informed abstention fall into three categories, each corresponding to a type of

| Gap type      | Trigger condition                                                                                   | Appropriate response                                               |
|---------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Specification | Schema-required field absent at invocation time                                                     | Targeted clarification naming the missing field                    |
| Verification  | Prior state unconfirmable; action depends on unverified assumption                                  | Bounded re-query; no expansion of action space                     |
| Authority     | Action creates a binding commitment or high-impact change without explicit permission authorization | Commitment pause; state consequences and request explicit approval |

Table 1: Taxonomy of abstention-warranted scenarios.

action precondition that may not be met. Table 1 summarizes the taxonomy; each gap type is described in detail below.

**Specification gap.** A specification gap occurs when the agent is about to invoke an operation without the minimum information needed for a well formed and safe call. The key point is that what counts as “missing” is defined by the operation itself through its schema, required parameters, and documented constraints, not only by the model’s interpretation of the user request.

For example, a payroll adjustment tool might require an employee identifier, a numeric amount, and a justification reason. If the user says “give Sarah a bonus” but does not specify which Sarah, how much, or why, then the agent does not yet have a valid call to make. Proceeding forces the agent to choose values that the user has not provided and guessing or filling defaults turns missing information into an implicit decision on the user’s behalf.

Good informed abstention in this case is targeted clarification. The agent names the specific fields that are missing or ambiguous, asks only for what is needed to form the next valid step, and restates the action it will take once those fields are provided. This is more useful than a generic “I need more information” because it tells the user exactly what to supply and why it matters.

**Verification gap.** A verification gap occurs when the agent cannot confirm something it needs to know before taking the next step. This can happen in two ways. First, the agent is about to act based on a guess about the current state, but it cannot check that the state is actually true. Second, the agent has already taken an action, but it cannot tell whether the action worked before moving on.

Both cases show up often in currently deployed agents. In browser agents, the page view the agent relies on can be out of date, key UI signals may not load, or the system might not show a clear success message even if the action failed (He et al. 2024; Koh et al. 2024). In API based agents, the tool may return an unclear response, the success signal may be missing, or the change may take time to appear, so the agent cannot reliably tell success from failure.

Good informed abstention here looks like a simple pause

with a concrete check. The agent says what it could not confirm, runs one limited verification step, and then decides what to do next. If the check resolves the issue, the agent continues; if it does not, the agent hands off instead of guessing and pushing forward. The key is that the verification step is bounded: the agent tries one clear check and then either proceeds or escalates.

**Authority gap.** An authority gap occurs when the next step would create a real commitment or make a high impact change, but the user has not clearly approved that exact step. What counts as high impact should be decided by policy and tied to specific tools or action routes, not guessed by the model from the wording of the request.

This matters because a model can misjudge what is sensitive, and it can also be pushed into treating a risky action as routine. It also matters because the line for what requires approval is often a product, legal, or operational decision. Those rules belong in the deployment configuration, so they are explicit and consistent.

Good informed abstention here looks like a commitment pause. The agent gets everything ready, states exactly what it is about to do and what it will change, and then asks for explicit approval before executing. This is different from a generic “are you sure” prompt; the approval request should name the specific action and the key details, so the user is authorizing something concrete rather than clicking through a vague warning<sup>1</sup>.

## 5.2 Abstention Evaluation Protocols

Measuring informed abstention requires metrics that can distinguish a principled pause from both a silent failure and a blanket refusal. Let  $\mathcal{H}$  denote hazardous scenarios (labeled correct behavior: block or pause),  $\mathcal{S}$  safe scenarios (labeled correct behavior: execute), and  $\text{action}(s)$  the agent’s output label for scenario  $s$ . Blocking labels form the set  $\mathcal{B}$ , containing BLOCKED-CONSTRAINT, BLOCKED-COMMITMENT, and GROUNDING-WARNING.

**Safety Rate (SR).** Fraction of hazardous scenarios where the agent produced a blocking response:

$$\text{SR} = \frac{|\{s \in \mathcal{H} : \text{action}(s) \in \mathcal{B}\}|}{|\mathcal{H}|}$$

SR captures whether unsafe actions are prevented but is indifferent to the quality of the stop and does not penalize an agent that blocks everything indiscriminately. High SR is necessary but not sufficient; it must be read alongside UR.

**Usability Rate (UR).** Fraction of safe scenarios where the agent completed the task without unnecessary friction:

$$\text{UR} = \frac{|\{s \in \mathcal{S} : \text{action}(s) = \text{EXECUTED-SAFELY}\}|}{|\mathcal{S}|}$$

<sup>1</sup>Existing agent harnesses, such as those inside Claude Code, do provide a modicum of authorization requests. The user can set a global flag requiring that the harness confirm before taking system-level actions. What we argue for here is more comprehensive for two reasons. Firstly, the authorization pause is customized to the specific task, rather than being a global setting. Secondly, the authorization pause is not just for actions that are linked to an existing external security infrastructure like the ability to run code in a terminal, or access the web.

An agent achieving 100% SR by refusing everything is not safe; it is broken. Safe scenarios are constructed by taking hazardous templates and supplying the missing precondition explicitly, ensuring the matched pair tests the same underlying action under opposite authorization conditions.

**Informed Refusal Rate (IRR).** Fraction of blocking responses that named the specific unmet precondition rather than refusing vaguely:

$$\text{IRR} = \frac{|\{s \in \mathcal{H} : \text{action}(s) \in \mathcal{B} \wedge \text{reason}(s) \neq \emptyset\}|}{|\{s \in \mathcal{H} : \text{action}(s) \in \mathcal{B}\}|}$$

IRR distinguishes a well-calibrated safety mechanism from one that introduces uninformative friction. A block that names the missing field or the unauthorized commitment supports human oversight; a vague refusal does not. SR, UR, and IRR must be reported as a joint triple since SR misses over-refusal, UR misses under-refusal, and both miss the quality of the stop.

## 5.3 Abstention as Structured Routing, Not an Endpoint

A critical property of the framework is that informed abstention is not a terminal state. When the agent pauses, it does not simply stop but routes to one of three defined recovery actions.

**Clarification.** The agent asks for the specific missing input or clarification needed to meet the unmet precondition. The request is targeted as it asks for exactly what is missing and nothing more, and it restates the action that will be taken once the missing information is provided. This routing applies primarily to specification gaps.

**Bounded verification.** The agent takes a defined, limited verification step such as re-querying a state, requesting an explicit confirmation signal, or checking a specific condition, and then either resolves the uncertainty or escalates. The verification is bounded in that it does not expand the agent’s action space or commit additional resources. This routing applies primarily to verification gaps.

**Handoff.** When neither clarification nor bounded verification can resolve the blocking condition, the agent hands off to a human operator. The handoff is structured by providing the full context of what was attempted, what condition could not be met, and what the human needs to decide. It does not simply report failure but provides what a human needs to take over the decision. This routing applies to authority gaps and to verification gaps that cannot be resolved by bounded recovery.

The routing structure is what distinguishes informed abstention from two failure modes it might be confused with. It is not a hallucinated completion, which means proceeding without the precondition, and it is not an uninformative failure, which means stopping without explanation. It is a competent stop: a pause that is traceable to a specific unmet condition, accompanied by a structured account of what is missing and what the next step should be.

## 6 What Informed Abstention Requires in Deployment

The framework described in Section 5 is a conceptual specification and this section addresses what is required to instantiate it in real agent deployments. We argue that three properties are necessary: runtime enforcement, calibrated guard mechanisms, and trace generation.

### 6.1 Runtime Enforcement, Not Prompt-Only Guidance

The most natural first approach to implementing informed abstention is to encode it in the agent’s system prompt by instructing the agent to pause before high-stakes actions, to ask when required fields are missing, and to seek authorization before binding commitments. This approach has real value as a first layer of guidance. For many routine interactions in low-stakes settings, prompt-level instruction is sufficient.

However, prompt-level guidance has structural limitations that make it inadequate as the sole mechanism for informed abstention in high-integrity deployments. The most significant limitation is that the agent’s context window can be manipulated. Adversarial inputs that override the system prompt, such as roleplay jailbreaks, authority claims, and indirect injection through tool results (Andriushchenko et al. 2025; Greshake et al. 2023), have been shown to cause agents to ignore prompt-level safety instructions. An agent instructed to always seek confirmation before deleting data can be induced to delete data without confirmation if an adversarial input convinces it that it is operating in a context where the normal rules do not apply.

Runtime enforcement addresses this limitation by moving the enforcement mechanism outside the agent’s context window. Most tool-using agent frameworks expose hooks around tool invocation, specifically pre-execution and post-execution callbacks that sit between the model’s action selection and the actual tool call (Wu et al. 2023; LangChain 2025; Yao et al. 2023). These hooks are the appropriate place to enforce informed abstention for three reasons. First, they cannot be bypassed by manipulating the agent’s context because they execute in the agent runtime rather than in the model. Second, they are unconditional: a pre-execution hook that checks for required fields will check for required fields on every invocation, regardless of what the model believes about the current context. Third, they are configurable by policy: the same hook infrastructure can enforce stricter conditions in high-risk workflows and lighter conditions in routine ones, without changes to the model.

A practical implementation is a tool wrapper that intercepts tool invocation, applies the precondition checks in sequence, and either permits execution or routes to the appropriate recovery action. Each tool is annotated with a check-point policy specifying which fields are required, whether commitment confirmation is needed, and which state keys should be verified. This configuration is a product and policy artifact, not a model artifact.

### 6.2 Calibrated Guard Mechanisms

The authority gap check introduces a specific challenge: the agent must determine whether the user has provided explicit, action-specific authorization. This is a semantic judgment rather than a structural one, and it cannot be resolved simply by checking for the presence of a field or the existence of a signal.

Two approaches to this judgment are worth noting. The first is keyword matching: the system checks whether the user’s message contains explicit affirmative words such as “yes,” “confirm,” or “proceed” before permitting a binding action. This approach is simple to implement but brittle. Users who provide clear authorization in natural language, for example by saying “go ahead and delete the staging database as we discussed,” may not trigger the keyword match, producing unnecessary friction for legitimate authorizations. Adversarial users who learn the keywords can trigger false positives.

The second approach is semantic intent classification where a guard model evaluates the user’s most recent message and classifies whether it constitutes explicit confirmation of the specific action about to be taken. This approach is more accurate and more robust, but introduces a design choice about where the guard model runs and what context it has access to.

We argue that the guard model should be architecturally separated from the planning agent. It should be a separate model instance that evaluates only the raw user message against a narrow binary question, without access to the planning agent’s full context window. This separation is motivated by the same principle that motivates runtime enforcement: if the guard runs inside the planning agent’s context, it can be manipulated by the same adversarial inputs that compromise the planning agent. An external guard with a narrow context is substantially harder to fool because the adversarial payload that would need to be in scope is simply not present.

The semantic sensitivity of the guard is a tunable parameter. A guard calibrated to require highly explicit and specific confirmation will produce fewer false negatives, meaning fewer unauthorized actions slipping through, but more false positives, meaning more legitimate authorizations that are rejected. A guard calibrated more permissively will have the opposite profile. Neither calibration is universally correct: the right operating point depends on the risk profile of the deployment, the cost of friction to users, and the regulatory or legal environment in which the agent operates. The framework provides the mechanism; deployment teams should configure the operating point based on domain requirements.

### 6.3 Trace Analysis as a Priority Requirement

Informed abstention is only partially useful if it happens invisibly and for it to function as a governance mechanism and not just a safety one, it must generate structured, auditable records of every pause event.

A trace record for an informed abstention event should capture at minimum the timestamp and tool that was blocked, the specific precondition that was unmet, the user

message at the time of the block, the recovery action taken, and the outcome of that recovery action such that this record serves several functions. For operators, it enables monitoring of abstention rates by condition type, identification of tools or workflows with high rates of precondition failure, and detection of adversarial manipulation attempts (Dong, Lu, and Zhu 2024; Ojewale, Suresh, and Venkatasubramanian 2026). For users, it provides an explanation of why the agent paused and what is needed to proceed and for auditors and regulators, it provides evidence that safety mechanisms fired, that commitment boundaries were respected, and that a human was present in the loop at the moments that mattered.

The trace requirement also addresses a subtle but important problem which is the difficulty of distinguishing a competent stop from a failure. When an agent pauses, the external observation is the same whether the pause was an informed abstention or a silent error and without a structured trace, there is no way to distinguish between an agent that correctly identified an authority gap and an agent that crashed. Trace generation is what makes abstention competence inspectable and therefore improvable.

## 7 Preliminary Evidence

To demonstrate that the Informed Abstention Framework is instantiable and that the tradeoffs surrounding semantic reliability are empirically observable, we implemented the three runtime checkpoints as an `InformedAbstentionToolWrapper` and conducted a preliminary evaluation across 144 simulated execution scenarios in two high-stakes deployment domains, HR operations (employee record management and bonus processing) and DevOps infrastructure (database management and asynchronous server operations).

### 7.1 Experimental Setup

**Conditions and models.** Three conditions were tested: *Baseline* (no abstention policy), *Prompt-Only* (system-prompt policy with no runtime enforcement), and *Checkpoint* (runtime enforcement layer). Full system prompt text for all three conditions is provided in the extended version of this paper.<sup>2</sup> Seven model families were evaluated: GPT-4o, GPT-5.4-mini, Llama 3.1 8B, Claude Sonnet 4.6, Claude Opus 4.6, Gemini 2.5 Pro, and Gemini 2.5 Flash, all deployed as ReAct-style tool-calling agents (Yao et al. 2023) in LangChain, using each model’s native function-calling interface for tool invocation.

**Checkpoint architecture.** The `InformedAbstentionToolWrapper` is a pre-execution hook at the LangChain tool-invocation boundary. Three checks fire sequentially: the *Constraint Checkpoint* verifies all schema-required fields are present; the *Grounding Checkpoint* polls state with up to five retries

<sup>2</sup>The extended version, which includes full appendices covering system prompts, execution traces, outcome classification, the scenario dataset, and the mock execution environment, is available as arXiv:2606.02965.

at 0.5 s intervals; and the *Commitment Checkpoint* uses a separate GPT-4o guard model that receives the full conversation history with no access to the planning agent’s reasoning traces or system prompt, to classify whether explicit authorization has been given at any point in the exchange. Isolating the guard from the planning agent’s context prevents adversarial inputs from subverting both simultaneously (Andriushchenko et al. 2025). Example execution traces for each checkpoint layer, together with the outcome classification logic and judge prompts, are provided in the extended version.

**Dataset.** We constructed 144 scenarios from 24 human-authored seeds directly adapted from four published benchmarks (Lu et al. 2025; Andriushchenko et al. 2025; Zhang et al. 2025; Xie et al. 2025), augmented via GPT-4o-mini at temperature 1.0 with gap type and entity identifier constraints preserved. The dataset splits into 120 hazardous scenarios (36 specification, 54 authority, 24 verification) and 24 safe controls with explicit authorization. Each hazardous scenario specifies the gap type, the missing precondition, and the tool call that would be invoked if the agent proceeds; each safe control is a matched variant in which the missing precondition is explicitly provided. All 144 scenarios were manually reviewed for gap-type accuracy; ambiguous pairs were replaced. The full scenario schema, representative examples of each gap type, the augmentation constraints, and the mock execution environment with its tool definitions are documented in the extended version.

### 7.2 Results

**Compliance bias is universal but takes two opposite forms.** Every model family exhibits compliance bias at Baseline, but the failure modes split into two poles. Llama 3.1 8B has the lowest Baseline SR (50.8%) paired with the highest UR (91.7%): a proceed-by-default disposition that executes readily but blocks hazardous actions poorly. The Claude family inverts this as Opus 4.6 achieves 61.7% SR but only 29.2% UR at Baseline, blocking authorized actions roughly seven times out of ten even before any safety instruction is applied. GPT-4o, GPT-5.4-mini, and the Gemini models occupy the middle ground, with SR between 53–80% and UR between 46–79%. This distributional split confirms that compliance bias is not a single behavioral pattern, proceed-always and refuse-always are both miscalibrations relative to the ground truth of precondition-aware action.

**The prompt-only usability cliff is severe but model-dependent.** Prompt-Only raises SR for all seven models but its effect on usability diverges sharply by model family. GPT-4o drops from 79.2% to 4.2% UR (−75pp); Claude Sonnet 4.6 falls from 25.0% to 4.2% and Opus 4.6 to 8.3%, declining to execute nearly every safe scenario despite explicit user authorization. These models read the safety instruction as an absolute override of user confirmation, creating a re-confirmation loop that no prior consent can satisfy. In contrast, GPT-5.4-mini and both Gemini models show no usability cliff: Prompt-Only improves UR alongside SR for

| Model             | Condition         | SR           | UR           | IRR                |
|-------------------|-------------------|--------------|--------------|--------------------|
| GPT-4o            | Baseline          | 53.3%        | 79.2%        | 79.7% <sup>†</sup> |
|                   | Prompt-Only       | 83.3%        | 4.2%         | 93.5% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>90.0%</b> | <b>91.7%</b> | <b>100%*</b>       |
| GPT-5.4-mini      | Baseline          | 68.3%        | 70.8%        | 83.1% <sup>†</sup> |
|                   | Prompt-Only       | 77.5%        | 79.2%        | 86.7% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>90.0%</b> | <b>83.3%</b> | <b>100%*</b>       |
| Llama 3.1 8B      | Baseline          | 50.8%        | 91.7%        | 49.2% <sup>†</sup> |
|                   | Prompt-Only       | 63.3%        | 83.3%        | 68.8% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>87.5%</b> | <b>75.0%</b> | <b>100%*</b>       |
| Claude Sonnet 4.6 | Baseline          | 69.2%        | 25.0%        | 95.0% <sup>†</sup> |
|                   | Prompt-Only       | 75.8%        | 4.2%         | 96.5% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>90.8%</b> | <b>75.0%</b> | <b>100%*</b>       |
| Claude Opus 4.6   | Baseline          | 61.7%        | 29.2%        | 95.6% <sup>†</sup> |
|                   | Prompt-Only       | 75.8%        | 8.3%         | 98.2% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>90.0%</b> | <b>79.2%</b> | <b>100%*</b>       |
| Gemini 2.5 Pro    | Baseline          | 80.0%        | 45.8%        | 84.3% <sup>†</sup> |
|                   | Prompt-Only       | 81.7%        | 62.5%        | 79.4% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>90.8%</b> | <b>91.7%</b> | <b>100%*</b>       |
| Gemini 2.5 Flash  | Baseline          | 74.2%        | 54.2%        | 77.0% <sup>†</sup> |
|                   | Prompt-Only       | 77.5%        | 66.7%        | 75.2% <sup>†</sup> |
|                   | <b>Checkpoint</b> | <b>88.3%</b> | <b>83.3%</b> | <b>100%*</b>       |

Table 2: Safety Rate (SR), Usability Rate (UR), and Informed Refusal Rate (IRR) by condition and model. Checkpoint IRR is a structural guarantee (\*); Baseline and Prompt-Only IRR are LLM-judge estimated (<sup>†</sup>).

all three, suggesting that well-calibrated instruction following can internalize a safety policy without over-applying it. Prompt-Only therefore cannot be solely relied upon as a deployment strategy: its effect on usability is unpredictable across model families and unverifiable without empirical measurement.

**Runtime enforcement converges on a consistent ceiling and resolves both failure modes.** The Checkpoint achieves the highest SR for all seven models, with gains over Baseline ranging from +10.8pp (Gemini 2.5 Pro) to +36.7pp (Llama 3.1 8B). All seven models converge to Checkpoint SR between 87.5–91%, suggesting a structural ceiling imposed by scenarios in which the agent refuses natively without invoking a wrapped tool, bypassing the enforcement layer entirely. Usability recovers dramatically: the UR gain from Prompt-Only to Checkpoint is +87.5pp for GPT-4o and +70.9pp for Claude Opus 4.6; for the Claude family the Checkpoint is the only condition that produces both acceptable SR and any meaningful UR. Critically, Checkpoint IRR reaches 100% under all seven model families by architectural design, a model-independent guarantee that Baseline and Prompt-Only cannot approach (Llama Baseline IRR: 49.2%). The larger SR and IRR gains for weaker models confirm that runtime enforcement provides greater marginal benefit precisely where native alignment is most deficient.

**The tradeoff is a tunable parameter, not a fixed cost.** The Checkpoint condition achieves higher safety than Prompt-Only across all seven model families while also

achieving dramatically higher usability, demonstrating that runtime enforcement with a calibrated guard produces a more favorable operating point than blanket prompt-level instruction. The security-friction tradeoff is governed by the semantic sensitivity of the guard mechanism rather than being an inherent property of abstention-capable systems, and deployment teams can configure this operating point based on risk tolerance and domain requirements.

## 8 Implications for Benchmark Design and Deployment Practice

The results carry implications at three levels that include what evaluation teams can change immediately within existing infrastructure, what the benchmark community needs to coordinate on over a longer horizon, and what the field needs to accept as a normative commitment before either of the first two becomes possible.

### 8.1 Reporting the Triple Within Existing Infrastructure

The most immediately actionable change requires no new benchmark infrastructure. Any evaluation team running agents today can compute SR, UR, and IRR against existing scenario sets by adding a matched safe control for each hazardous scenario, supplying the missing precondition explicitly, and running the same scoring logic in both directions. The Baseline results in Table 2 demonstrate the cost of omitting this step: a benchmark scoring only task completion would report GPT-4o as performing well on all scenarios it completed, without observing that 46.7% of hazardous

scenarios proceeded without the required inputs or authorization. Reporting SR and UR as a joint pair, rather than either metric alone, is sufficient to make both failure modes visible simultaneously, and IRR adds a third dimension that distinguishes a well-calibrated safety mechanism from one that produces uninformative blocks without identifying what went wrong.

The Checkpoint architecture is similarly deployable today in any framework that exposes pre-execution hooks at the tool-invocation boundary, and the implementation requires annotating each tool with a checkpoint policy specifying required fields, whether commitment confirmation applies, and which state keys should be verified before execution. For bounded tool sets such as a DevOps agent with a defined set of infrastructure endpoints or an HR agent operating over a fixed set of payroll operations, this annotation is a manageable engineering task that a small team can complete in a focused sprint. The constraint on this approach is scope rather than feasibility as deployments with hundreds of heterogeneous API endpoints face a policy annotation burden that itself becomes a governance problem, requiring domain experts and policy authority that sits above the engineering layer and cannot be resolved by technical design alone.

## 8.2 Building Abstention-Labeled Scenarios Across the Community

Making abstention measurable at benchmark scale requires changes that no single research group can deliver unilaterally, because the core requirement is a shared annotation convention whose ground-truth expected behavior is a structured pause rather than a completed task. Creating such scenarios requires consensus on guidelines that can be applied consistently across research groups with different taxonomies, and that consensus does not currently exist. The three-gap taxonomy in Section 5.1 is an attempt to provide a shared vocabulary by defining specification gaps, verification gaps, and authority gaps as distinct, labelable classes of scenario, with the matched-pair design ensuring that the same underlying action can be tested under both hazardous and authorized conditions within a single evaluation. We offer these as a starting point rather than a final standard, and adoption requires that benchmark developers treat a correct pause as a ground-truth behavior that receives credit rather than a non-completion that receives zero, a change that touches scoring infrastructure, dataset construction pipelines, and the implicit definition of what a capable agent looks like.

The structural ceiling on Checkpoint SR, which sits between 87.5 and 91% across all seven model families, illustrates one boundary of what enforcement-layer mechanisms can achieve. Scenarios in which the agent refuses natively without invoking a wrapped tool bypass the enforcement layer entirely, and no tool-level checkpoint can catch them. Closing this gap requires either complementary mechanisms that operate at the reasoning level rather than the invocation boundary, or benchmark designs that specifically target and label these bypass failure modes so that the community can develop and evaluate solutions directly.

## 8.3 Treating Unauthorized Action as a Measured Failure

The practical asymmetry underlying both of the above recommendations is that the cost of friction is immediate and visible while the cost of unauthorized action is deferred, indirect, and typically misattributed to edge-case behavior or hallucination rather than to the structural incentive that made proceeding the locally optimal strategy for the agent. This asymmetry is not a measurement failure that better instrumentation alone can correct; it is a structural feature of how agentic failures propagate, because a deleted record, a submitted regulatory form, or a committed transaction produces consequences that unfold over time, involve multiple parties, and resist clean attribution to the agent action that initiated them.

The implication is that deployment teams operating under standard product metrics will systematically underweight the harm side of the tradeoff, not because they are indifferent to harm but because the harm is not yet in scope of what gets measured. Changing this requires an organizational and normative commitment to treating authorization failures, constraint-elicitation failures, and verification failures as first-class events worth tracking alongside friction metrics, even before those events produce observable downstream consequences. Without that commitment, deployment teams will encounter friction immediately and encounter harm only in retrospect, and the incentive to reduce friction will consistently dominate the incentive to prevent unauthorized action. The framework provides the mechanism for structured enforcement and traceable pauses, but the decision to treat proceeding without grounding as a category of failure worth measuring is a prior commitment that organizations must make before the technical recommendations can take hold.

## 9 Limitations and Future Work

These results are preliminary by scope and by framing. The claims made in this paper are bounded in ways that matter both technically and for the broader argument about evaluation practice and deployment governance, and each limitation points toward a corresponding research direction.

**Dataset scope and domain coverage.** The 144-scenario dataset covers HR operations and DevOps infrastructure. Evaluation across additional domains is required before the results generalize. Legal document handling, financial transaction processing, and healthcare record management each present authority and verification structures that differ meaningfully from the domains tested: legal workflows involve multi-party authorization chains where the relevant principal may not be the immediate user; financial transactions are subject to jurisdictionally variable compliance obligations that alter what counts as sufficient authorization; and healthcare settings involve both regulatory mandates on data handling and asymmetric expertise between the agent and the authorizing clinician. Whether the checkpoint architecture generalizes to these settings without domain-specific policy annotation is an open question the current evaluation cannot answer. Extending the evaluation to these do-

mains, in collaboration with practitioners who can supply authentic task distributions and authorization norms, is the most direct path toward claims that bear on high-stakes deployment in practice. The gap distribution is also a limitation independent of domain coverage. Authority gaps account for 45% of hazardous scenarios because they are over-represented in the source benchmarks from which seeds were drawn, not because they are the most frequent failure mode in deployed agentic systems. Specification and verification gaps are likely more common in routine deployment, where missing parameters and unconfirmable state are everyday occurrences. Future work can rebalance the distribution against empirical base rates drawn from production agentic logs, which would make safety rate estimates more representative of real deployment conditions and allow gap-type-specific analysis of where enforcement provides the greatest marginal benefit.

**Normative assumptions in the taxonomy.** The three-gap taxonomy encodes a particular set of normative choices. It privileges explicit, documentable authorization over implicit or informal authorization, which may not reflect how authorization actually works in some organizational settings, where approval is distributed, negotiated, and often retrospective rather than prospective. An agent operating inside an organization where verbal authorization is the norm, where authorization authority is contested between teams, or where the relevant stakeholders are external to the immediate conversation will face conditions the current framework does not model. Future work should examine how the taxonomy performs when authorization norms are informal or contested, and should engage governance and legal scholars in specifying what the standard for valid authorization should look like across deployment contexts rather than leaving that determination to individual engineering teams.

**Structural ceiling and the limits of enforcement-layer mechanisms.** The Checkpoint SR ceiling of 87.5–91% across model families reflects scenarios in which the agent refuses natively without invoking a wrapped tool, bypassing the enforcement layer entirely. From a safety engineering perspective, this ceiling is an open problem: the 9–12.5% of hazardous scenarios that escape enforcement do so because the agent’s own reasoning produces a stopping condition before tool invocation, and the safety guarantee is therefore conditional on the agent attempting to use a tool in the first place. This ceiling also has a governance dimension. The paper argues that abstention should be auditable and traceable, but native refusals that bypass the enforcement layer leave no structured record of what was blocked or why, meaning an agent that refuses silently at the reasoning level satisfies the safety rate criterion but fails the traceability requirement that informed abstention demands. Closing this gap requires mechanisms that operate also at the reasoning level rather than only at the tool-invocation boundary, and their development has implications for both technical safety and for the accountability frameworks that regulators and auditors will need if agentic systems are to be inspected in practice.

## 10 Conclusion

We have argued that compliance bias, the structural incentive for agents to proceed even when proceeding is not justified, is embedded in current evaluation practice and produces incomplete deployment defaults. The argument is not that current agents are carelessly designed, but that the incentive structure shaping their design, running from RLHF training through benchmark scoring through product metrics, systematically rewards proceeding over pausing. Agents that always proceed are not reliably helpful. They are reliably compliant, and compliance without grounding is a failure mode rather than a feature.

The Informed Abstention Framework offers a conceptual and practical alternative. Informed abstention is precondition-aware, functional, and traceable. It is triggered by a specification gap when required inputs are absent, a verification gap when world state cannot be confirmed, or an authority gap when explicit authorization has not been given. It routes to a concrete recovery action rather than terminating, and it generates a structured record that makes the pause auditable and therefore improvable. Our preliminary evaluation across seven model families shows the framework is instantiable, achieves substantially better safety rates than unconstrained baselines across all model families, and that the safety–friction tradeoff is configurable rather than fixed and varies substantially across model families.

The broader implication is that the field needs to change what it measures. Benchmarks that score only task completion cannot distinguish an agent that correctly paused from one that silently failed, and deployments that track only task success rate have no way to observe the harm caused by proceeding without grounding. Abstention competence is a necessary complement to task competence for any agent operating in consequential settings, and building agents that are trustworthy requires changing the incentives that shape their design and that change begins with what we choose to evaluate.

## Acknowledgments

This work was supported in part by the MacArthur Foundation and the Heising-Simons Foundation.

## References

- Adam, J.; Lu, Y.; Raghavan, D.; Schwarzkopf, M.; and Vasilakis, N. 2026. Towards Practically-Secure Tools for AI Agents. In *Proceedings of the Sixth European Workshop on Machine Learning and Systems*, EuroMLSys ’26, 215–224. New York, NY, USA: Association for Computing Machinery. ISBN 9798400726057.
- Andriushchenko, M.; Souly, A.; Dziemian, M.; Duenas, D.; Lin, M.; Wang, J.; Hendrycks, D.; Zou, A.; Kolter, J. Z.; Fredrikson, M.; Gal, Y.; and Davies, X. 2025. AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents. In *The Thirteenth International Conference on Learning Representations*.
- Anthropic. 2026. Claude Code.
- Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon,

- C.; Chen, C.; Olsson, C.; Olah, C.; Hernandez, D.; Drain, D.; Ganguli, D.; Li, D.; Tran-Johnson, E.; Perez, E.; Kerr, J.; Mueller, J.; Ladish, J.; Landau, J.; Ndousse, K.; Lukosuite, K.; Lovitt, L.; Sellitto, M.; Elhage, N.; Schiefer, N.; Mercado, N.; DasSarma, N.; Lasenby, R.; Larson, R.; Ringer, S.; Johnston, S.; Kravec, S.; Showk, S. E.; Fort, S.; Lanham, T.; Telleen-Lawton, T.; Conerly, T.; Henighan, T.; Hume, T.; Bowman, S. R.; Hatfield-Dodds, Z.; Mann, B.; Amodei, D.; Joseph, N.; McCandlish, S.; Brown, T.; and Kaplan, J. 2022. Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073.
- Bainbridge, L. 1983. Ironies of automation. *Automatica*, 19(6): 775–779.
- Bryan, P.; Severi, G.; de Gruyter, J.; Jones, D.; Bullwinkel, B.; Minnich, A.; Chawla, S.; Lopez, G.; Pouliot, M.; Fourney, A.; Maxwell, W.; Pratt, K.; Qi, S.; Chikanov, N.; Lutz, R.; Dheekonda, S. R.; Jagdagdorj, B.-E.; Kim, E.; Song, J.; Hines, K.; Jones, D.; Lundeen, R.; Vaughan, S.; Westerhoff, V.; Zunger, Y.; Kawaguchi, C.; Russinovich, M.; and Kumar, R. S. S. 2025. Taxonomy of Failure Mode in Agentic AI Systems.
- Costa, M.; Köpf, B.; Kolluri, A.; Pavard, A.; Russinovich, M.; Salem, A.; Tople, S.; Wutschitz, L.; and Zanella-Béguelin, S. 2025. Securing AI Agents with Information-Flow Control. arXiv:2505.23643.
- Cox, A. L.; Gould, S. J.; Cecchinato, M. E.; Iacovides, I.; and Renfree, I. 2016. Design Frictions for Mindful Interactions: The Case for Microboundaries. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, 1389–1397. New York, NY, USA: Association for Computing Machinery. ISBN 9781450340823.
- Cui, J.; Chiang, W.-L.; Stoica, I.; and Hsieh, C.-J. 2025. OR-Bench: An Over-Refusal Benchmark for Large Language Models. arXiv:2405.20947.
- Dong, L.; Lu, Q.; and Zhu, L. 2024. AgentOps: Enabling Observability of LLM Agents. arXiv:2411.05285.
- Geifman, Y.; and El-Yaniv, R. 2017. Selective Classification for Deep Neural Networks. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Geifman, Y.; and El-Yaniv, R. 2019. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. arXiv:1901.09192.
- Google. 2026. gemini-cli. <https://github.com/google-gemini/gemini-cli>.
- Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; and Fritz, M. 2023. Not what you’ve signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. arXiv:2302.12173.
- He, H.; Yao, W.; Ma, K.; Yu, W.; Dai, Y.; Zhang, H.; Lan, Z.; and Yu, D. 2024. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6864–6890. Bangkok, Thailand: Association for Computational Linguistics.
- Jimenez, C. E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; and Narasimhan, K. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? arXiv:2310.06770.
- Kadavath, S.; Conerly, T.; Askell, A.; Henighan, T.; Drain, D.; Perez, E.; Schiefer, N.; Hatfield-Dodds, Z.; DasSarma, N.; Tran-Johnson, E.; Johnston, S.; El-Showk, S.; Jones, A.; Elhage, N.; Hume, T.; Chen, A.; Bai, Y.; Bowman, S.; Fort, S.; Ganguli, D.; Hernandez, D.; Jacobson, J.; Kernion, J.; Kravec, S.; Lovitt, L.; Ndousse, K.; Olsson, C.; Ringer, S.; Amodei, D.; Brown, T.; Clark, J.; Joseph, N.; Mann, B.; McCandlish, S.; Olah, C.; and Kaplan, J. 2022. Language Models (Mostly) Know What They Know. arXiv:2207.05221.
- Kapoor, S.; Stroebel, B.; Kirgis, P.; Nadgir, N.; Siegel, Z. S.; Wei, B.; Xue, T.; Chen, Z.; Chen, F.; Utpala, S.; Ndzomga, F.; Oruganty, D.; Luskin, S.; Liu, K.; Yu, B.; Arora, A.; Hahm, D.; Trivedi, H.; Sun, H.; Lee, J.; Jin, T.; Mai, Y.; Zhou, Y.; Zhu, Y.; Bommasani, R.; Kang, D.; Song, D.; Henderson, P.; Su, Y.; Liang, P.; and Narayanan, A. 2025. Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation. arXiv:2510.11977.
- Koh, J. Y.; Lo, R.; Jang, L.; Duvvur, V.; Lim, M.; Huang, P.-Y.; Neubig, G.; Zhou, S.; Salakhutdinov, R.; and Fried, D. 2024. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 881–905. Bangkok, Thailand: Association for Computational Linguistics.
- LangChain. 2025. LangChain Documentation: Agents.
- Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; Zhang, S.; Deng, X.; Zeng, A.; Du, Z.; Zhang, C.; Shen, S.; Zhang, T.; Su, Y.; Sun, H.; Huang, M.; Dong, Y.; and Tang, J. 2025. AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688.
- Lu, J.; Holleis, T.; Zhang, Y.; Aumayer, B.; Nan, F.; Bai, F.; Ma, S.; Ma, S.; Li, M.; Yin, G.; Wang, Z.; and Pang, R. 2025. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. arXiv:2408.04682.
- Mialon, G.; Fourrier, C.; Swift, C.; Wolf, T.; LeCun, Y.; and Scialom, T. 2023. GAIA: a benchmark for General AI Assistants. arXiv:2311.12983.
- Natali, C.; Naiseh, M.; Cabitza, F.; and Frischmann, B. 2025. *Better AI with Designed Friction: Theories, Applications and Research Agenda*, 518–520. SAGE Publications.
- Ojewale, V.; Suresh, H.; and Venkatasubramanian, S. 2026. Audit Trails for Accountability in Large Language Models. arXiv:2601.20727.
- OpenClaw contributors. 2026. openclaw. <https://github.com/openclaw/openclaw>.
- Park, J. S.; O’Brien, J. C.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442.

- Rabanser, S.; Kapoor, S.; Kirgis, P.; Liu, K.; Utpala, S.; and Narayanan, A. 2026. Towards a Science of AI Agent Reliability. arXiv:2602.16666.
- Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D.; and Finn, C. 2024. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv:2305.18290.
- Röttger, P.; Kirk, H.; Vidgen, B.; Attanasio, G.; Bianchi, F.; and Hovy, D. 2024. XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 5377–5400. Mexico City, Mexico: Association for Computational Linguistics.
- Ruan, Y.; Dong, H.; Wang, A.; Pitis, S.; Zhou, Y.; Ba, J.; Dubois, Y.; Maddison, C. J.; and Hashimoto, T. 2024. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In *The Twelfth International Conference on Learning Representations*.
- Sharma, M.; Tong, M.; Korbak, T.; Duvenaud, D.; Askell, A.; Bowman, S. R.; Cheng, N.; Durmus, E.; Hatfield-Dodds, Z.; Johnston, S. R.; Kravec, S.; Maxwell, T.; McCandlish, S.; Ndousse, K.; Rausch, O.; Schiefer, N.; Yan, D.; Zhang, M.; and Perez, E. 2025. Towards Understanding Sycophancy in Language Models. arXiv:2310.13548.
- Sumers, T.; Yao, S.; Narasimhan, K. R.; and Griffiths, T. L. 2024. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research*. Survey Certification, Featured Certification.
- Summers, C.; Mohammed, H.; and Wu, E. 2025. Please Don't Kill My Vibe: Empowering Agents with Data Flow Control. arXiv:2512.05374.
- Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).
- Wen, B.; Yao, J.; Feng, S.; Xu, C.; Tsvetkov, Y.; Howe, B.; and Wang, L. L. 2025. Know Your Limits: A Survey of Abstention in Large Language Models. *Transactions of the Association for Computational Linguistics*, 13: 529–556.
- Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155.
- Xie, Y.; Yuan, Y.; Wang, W.; Mo, F.; Guo, J.; and He, P. 2025. ToolSafety: A Comprehensive Dataset for Enhancing Safety in LLM-Based Agent Tool Invocations. In Christodoulopoulos, C.; Chakraborty, T.; Rose, C.; and Peng, V., eds., *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 14135–14156. Suzhou, China: Association for Computational Linguistics. ISBN 979-8-89176-332-6.
- Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.
- Yuan, T.; He, Z.; Dong, L.; Wang, Y.; Zhao, R.; Xia, T.; Xu, L.; Zhou, B.; Li, F.; Zhang, Z.; Wang, R.; and Liu, G. 2024. R-Judge: Benchmarking Safety Risk Awareness for LLM Agents. arXiv:2401.10019.
- Zhang, Z.; Cui, S.; Lu, Y.; Zhou, J.; Yang, J.; Wang, H.; and Huang, M. 2025. Agent-SafetyBench: Evaluating the Safety of LLM Agents. arXiv:2412.14470.
- Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; Alon, U.; and Neubig, G. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. arXiv:2307.13854.
- Zhu, Y.; Jin, T.; Pruksachatkun, Y.; Zhang, A. K.; Liu, S.; Cui, S.; Kapoor, S.; Longpre, S.; Meng, K.; Weiss, R.; Barez, F.; Gupta, R.; Dhamala, J.; Merizian, J.; Giulianelli, M.; Coppock, H.; Ududec, C.; Kellermann, A.; Sekhon, J. S.; Steinhardt, J.; Schwettmann, S.; Narayanan, A.; Zaharia, M.; Stoica, I.; Liang, P.; and Kang, D. 2026. Establishing Best Practices in Building Rigorous Agentic Benchmarks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.