Tool Execution Hallucination in LLM-based Agents: A Unified Taxonomy with Detection, Mitigation, and Future Directions

Anonymous ACL submission

Abstract

Large Language Model (LLM) agents increasingly rely on external tools to solve complex tasks, yet their failures often arise from erroneous decisions within the agent execution flow rather than tool unavailability. Although hallucination in LLMs has been widely studied, existing surveys largely focus on factual errors or coarse-grained tool usage, overlooking agent-specific hallucinations that emerge from decision failures during tool execution.

In this survey, we systematically characterize tool execution hallucination in LLM-based agents by formalizing failure modes with respect to the agent execution flow. We propose a unified taxonomy that organizes hallucinations according to where they occur in the agent pipeline, with a particular focus on the *tool selection* and *tool calling* stages under both single-turn and multi-turn settings. Based on this taxonomy, we synthesize existing detection and mitigation techniques according to how they intervene in different stages of the agent execution flow, and review representative benchmarks. Our analysis highlights key gaps in current methods, including error propagation across iterations and insufficient awareness of task solvability. By grounding these challenges in the agent execution flow, our survey provides a unifying perspective that connects failure modes, mitigation strategies, and evaluation protocols, and outlines future directions toward more reliable and trustworthy LLM agents.

1 Introduction

Large Language Models (LLMs) have reshaped artificial intelligence by demonstrating strong capabilities in language understanding, generation, and reasoning across diverse domains (Brown et al., 2020; Touvron et al., 2023). Building upon these capabilities, LLM-based agents extend static language generation into interactive and goal-driven settings by integrating reasoning, planning, memory, and external tool usage (Yao et al., 2022; Paranjape et al., 2023; Wei et al., 2022; Kambhampati et al., 2024). Through tool execution, such agents can retrieve up-to-date information from the web, execute code, and interact with software APIs, enabling promising applications in autonomous assistance and complex decision-making systems (Qu et al., 2025).

Despite their growing capabilities, LLM-based agents remain fragile in practice. Empirical and industrial analyses consistently report that failures often arise not from the lack of tools, but from erroneous decisions made during the execution process itself (Wang et al., 2025; Zhu et al., 2025; Duan et al., 2025; Xiong et al., 2025a). Unlike conventional LLM hallucinations that primarily manifest as incorrect or fabricated textual content (Huang et al., 2025; Woesle et al., 2025), agent failures frequently involve incorrect tool-related decisions, such as selecting an inappropriate tool, omitting a necessary tool, invoking tools with incorrect arguments, or propagating early mistakes across multiple interaction steps (Lin et al., 2025; Winston and Just, 2025).

A core reason these failures are difficult to analyze is that agent tool use is inherently *iterative*. A widely adopted abstraction views the agent as repeatedly cycling through *task planning*, *tool selection*, and *tool calling* until the task is completed, followed by response generation (Qu et al., 2025). Errors in any cycle can bias subsequent cycles, accumulate across steps, and become increasingly difficult to detect or correct (Duan et al., 2025; Zhu et al., 2025). In this survey, we refer to this loop as the *agent execution flow* and use it as the organizing backbone for our taxonomy and synthesis.

Positioning of this survey. Recent years have witnessed a growing number of surveys on hallucinations in large language models and agentic systems. However, these works differ substantially in scope, abstraction level, and analytical focus.

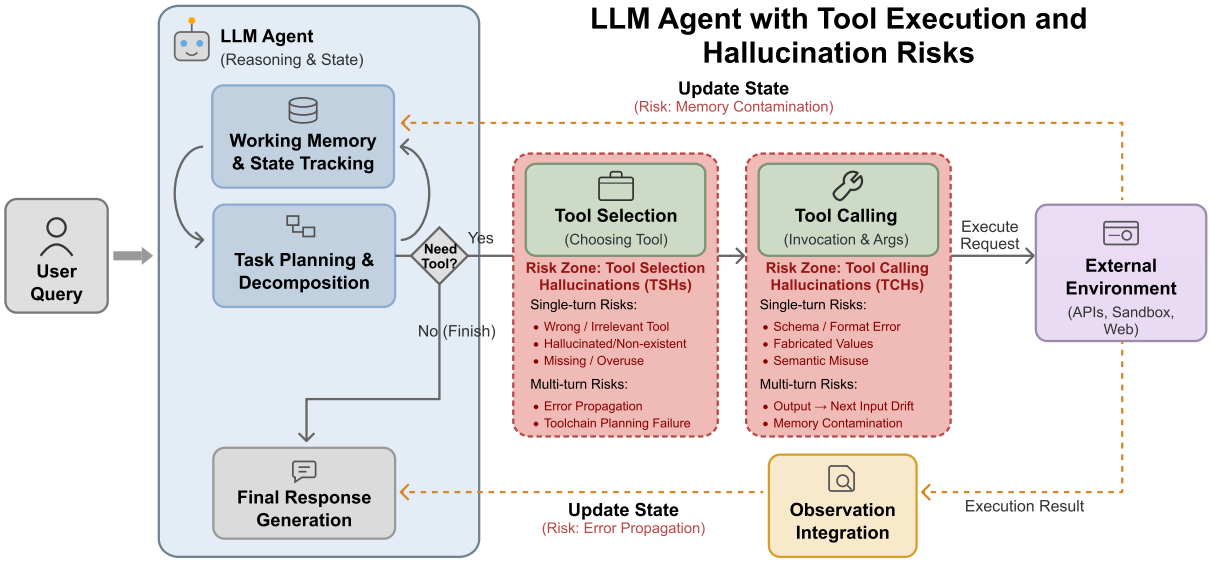

Figure 1: A conceptual workflow of an LLM-based agent's iterative tool execution. The diagram illustrates the interplay between the agent's internal reasoning and state tracking, and its external actions. The red dashed boxes explicitly mark the decision stages – Tool Selection and Tool Calling – that are susceptible to Tool Selection Hallucinations (TSHs) and Tool Calling Hallucinations (TCHs), respectively. Failures in these stages can propagate via the observation-and-update loop (orange dashed lines) to the agent's memory, leading to compounding errors across iterations.

Table 1 compares representative surveys with our work. In contrast to prior surveys that primarily study hallucinations at the LLM or high-level agent abstraction, our survey grounds hallucination phenomena in the *agent execution flow* and provides a stage-aligned taxonomy centered on *tool selection* and *tool calling*. This execution-centric perspective enables a unified synthesis of detection, mitigation, and evaluation methods, while explicitly accounting for multi-turn error propagation and task solvability.

Motivated by these observations, this survey systematically examines hallucinations in LLM-based agents through the lens of failure modes along the agent execution flow. We focus specifically on hallucinations arising during the *tool selection* and *tool calling* stages, where erroneous decisions most directly affect agent behavior and system reliability. We propose a unified taxonomy that categorizes hallucinations based on where and how they occur in the agent pipeline, covering both single-turn and multi-turn settings. Building on this taxonomy, we synthesize existing detection and mitigation techniques according to how they intervene at different stages of the agent execution flow, and review representative benchmarks that stress-test these failure modes.

Scope of literature coverage. We surveyed tool-execution-hallucination and closely related tool-augmented agent literature from 2023 to December 2025, aligned with the emergence and rapid evolution of function-calling, constrained decoding, and tool-use benchmarks for LLM agents (Winston and Just, 2025; Lin et al., 2025).

By consolidating fragmented findings across the literature, our analysis reveals recurring limitations in current approaches, including compounding errors across iterations, weak solvability and tool-necessity awareness, and insufficient semantic/state verification beyond surface-form correctness. Figure 2 illustrates the tool execution hallucination taxonomy used throughout this survey.

2 Definition and Scope

Agent Hallucination. Hallucination in LLMs is commonly defined as generating content that is factually incorrect or unfaithful to inputs or sources (Huang et al., 2025; Woesle et al., 2025). Recent work extends this notion from pure language generation to agentic behaviors. Following the agent-centric view in (Lin et al., 2025), we define *agent hallucinations* as erroneous or fictitious operations arising during the decision-making and execution processes of LLM-based agents. Formally, at execution step t, an agent policy π generates an action $a_t \sim \pi(a \mid s_t, q)$ based on the user query q and

Dimension	(Lin et al., 2025)	(Winston and Just, 2025)	(Tonmoy et al., 2024)	(Huang et al., 2025)	Ours
Execution-stage definition	✓	✓	–	–	✓
Selection vs. calling distinction	○	✓	–	–	✓
Multi-turn propagation analysis	✓	○	–	○	✓
Stage-aligned mitigation synthesis	○	○	✓	✓	✓
Tool-execution benchmarks coverage	–	✓	○	○	✓
Solvability / tool-necessity modeling	○	–	–	○	✓
System-level reliability guarantees	○	✓	–	–	✓

Table 1: Comparison of representative surveys on hallucinations and tool-augmented LLMs. Our survey emphasizes an execution-stage view of hallucinations with an explicit distinction between tool selection and tool calling, and systematically synthesizes multi-turn propagation, solvability/tool-necessity, benchmarks, and system-level reliability mechanisms. Here, ✓ indicates explicit coverage, ○ indicates partial or implicit discussion, and – indicates not addressed.

current state s_t. Such hallucinations are not limited to linguistic outputs, but emerge from interactions among planning, tool use, state tracking, and memory modules.

A key property of agent hallucinations is their endogenous nature: they can be generated within the agent's own decision process and amplified through iterative execution loops, rather than being directly caused by external environment failures (Lin et al., 2025). This cross-step propagation distinguishes agent hallucinations from isolated tool errors and makes them central to agent safety and reliability.

Execution Hallucination. Within agent hallucinations, prior work identifies *execution hallucinations* as a critical subclass (Lin et al., 2025). Execution hallucinations occur when an agent assumes or claims that certain execution sub-stages have been correctly completed, while they have in fact not been executed, have failed, or have been carried out incorrectly. In iterative workflows (Figure 1), an execution hallucination can be viewed as an erroneous state transition where the agent's assumed state s_{t+1} diverges from the true environment state, and these incorrect assumptions about past actions bias subsequent planning and tool decisions.

Tool Selection and Tool Calling Hallucinations. Following the execution-centric taxonomy in (Lin et al., 2025), execution hallucinations can be divided according to where the erroneous decision occurs in the tool-augmented loop: *Tool Selection Hallucinations (TSHs)* and *Tool Calling Hallucinations (TCHs)*.

TSHs arise when an agent selects non-existent, irrelevant, or inappropriate tools for a subtask, or fails to correctly decide whether tool use is necessary (Lin et al., 2025). Let $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_N\}$ be the provided tool inventory and $\tau_\emptyset$ denote the decision to bypass external tools. The selection policy outputs a tool choice $\hat{\tau} \in \mathcal{T} \cup \{\tau_\emptyset\}$. A TSH occurs when $\hat{\tau} \neq \tau^*$, where τ^* is the oracle tool choice. These failures occur prior to execution and are often driven by mismatches between task requirements and tool descriptions, weak solvability awareness, or confusion under large tool catalogs (Winston and Just, 2025).

TCHs occur after a tool has been selected and involve errors in how the tool is invoked. Let Ω_τ define the valid schema and parameter constraint space for a selected tool τ. The agent generates arguments $\hat{x} \sim \pi(x \mid \tau, s_t)$. TCHs encompass syntactic violations where $\hat{x} \notin \Omega_\tau$ (e.g., malformed schemas, missing parameters) and semantic violations where $\hat{x} \in \Omega_\tau$ but the values are invalid or fabricated given the context s_t (Lin et al., 2025; Wang et al., 2025). Such errors may lead to silent failures that contaminate state and memory when incorrect tool outputs are trusted.

In this survey, we focus on hallucinations arising during the *tool selection* and *tool calling* stages, as these stages constitute the primary interface between internal reasoning and external execution in tool-augmented agents.

3 Tool Selection Hallucinations

We initiate the taxonomy by anchoring hallucination phenomena to explicit decision stages in the agent execution flow (Figure 2). Under the widely adopted iterative abstraction, tool-augmented agents repeatedly perform planning, tool selection (deciding whether/which tool to use), and tool calling (deciding how to invoke the selected tool), while feeding observations back into subsequent iterations (Figure 1) (Duan et al., 2025; Zhu et al., 2025). From this execution-centric perspective, *tool execution hallucinations* arise when the agent makes systematically erroneous decisions at these stages, and the iterative loop can amplify such errors via context accumulation, state drift,

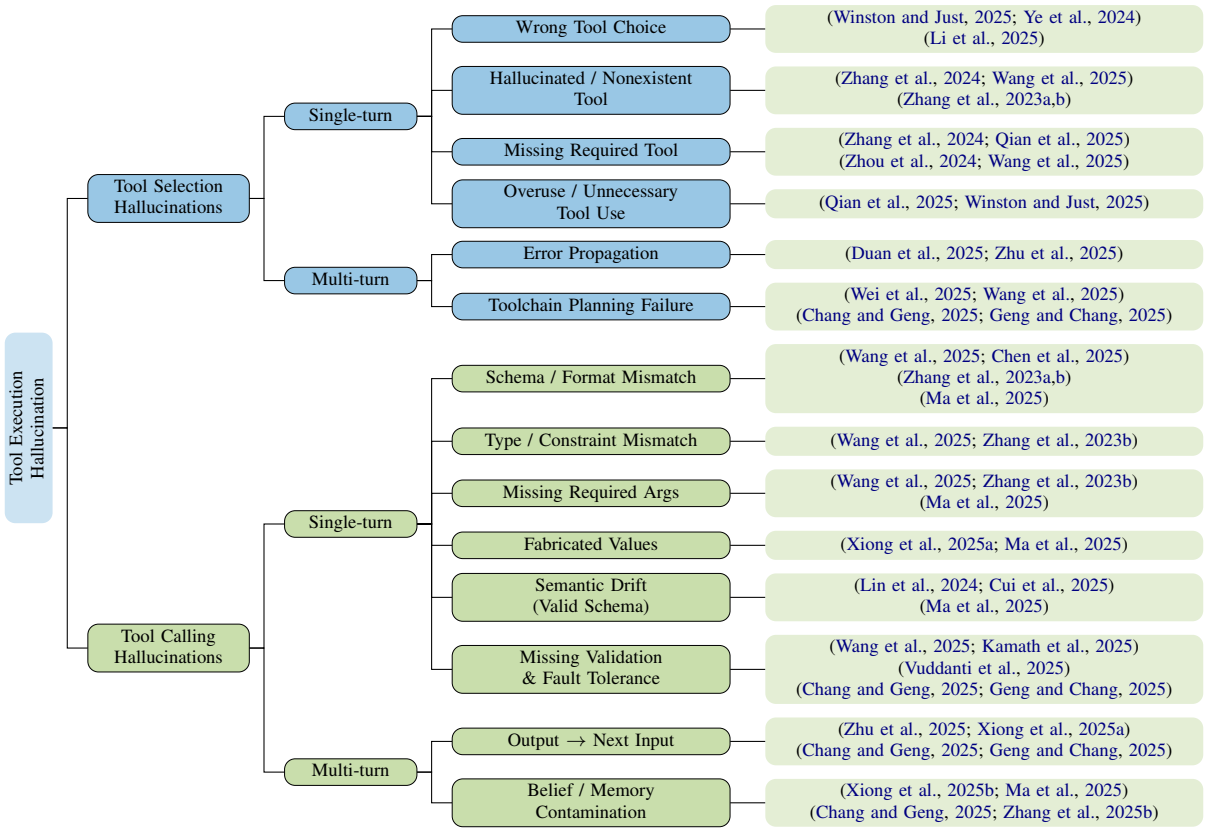

Figure 2: A taxonomy of tool execution hallucinations in LLM-based agents, organized by tool selection and tool calling stages under single-turn and multi-turn settings.

and memory updates.

We begin with TSHs because selection constitutes the first interface between internal deliberation and external execution. Once an agent commits to an inappropriate tool (or omits tool use when necessary), downstream components may remain locally coherent yet globally misaligned with the task objective (Winston and Just, 2025; Zhang et al., 2024). Moreover, selection decisions are intrinsically boundary-sensitive: the agent must reconcile task requirements, tool affordances, and uncertainty about solvability under partial observability, which naturally gives rise to both overuse and underuse behaviors (Winston and Just, 2025). We accordingly organize TSHs into **single-turn** and **multi-turn** settings, reflecting whether a failure is confined to a single decision step or propagates across iterative executions.

3.1 Single-turn TSHs

In single-turn settings, selection failures manifest in four forms: selecting inappropriate tools, proposing hallucinated tools, missing required tools, or overusing tools (Winston and Just, 2025; Qian et al., 2025; Zhang et al., 2024). While these behaviors differ operationally, they can be traced to mismatches among (i) task demands, (ii) the provided tool inventory and its descriptions, and (iii) the agent's calibration of uncertainty and solvability boundaries.

Wrong tool choice occurs when selecting an available but irrelevant tool, often driven by surface-level matching of names or partial descriptions (Winston and Just, 2025). This vulnerability is exacerbated in large catalogs containing naming noise (Ye et al., 2024; Li et al., 2025), or by upstream planning errors cementing incorrect subgoals (Duan et al., 2025). *Detection* relies on auditing tool registries and perturbation-based stress tests (Winston and Just, 2025). Correspondingly, *mitigation* strategies include improving interface discriminability (Winston and Just, 2025), enforcing constrained decoding (Zhang et al., 2023a,b), and employing retrieval-then-select pipelines for large toolsets (Winston and Just, 2025).

Hallucinated/nonexistent tool selection happens when an agent proposes a tool absent from the registry, often reverting to pretraining priors under underspecified prompts (Zhang et al., 2024). Treating tool identifiers as generatable text rather

than closed-registry members amplifies this behavior (Wang et al., 2025). *Detection* is straightforward via existence checks and tool-availability benchmarks (Zhang et al., 2024). *Mitigation* requires inventory-closed constrained decoding (Zhang et al., 2023a,b) alongside explicit abstention policies (Zhang et al., 2024).

Missing required tool refers to failing to invoke external tools when necessary, instead relying on parametric knowledge. This correlates with miscalibrated self-confidence (Qian et al., 2025; Zhou et al., 2024) and pipeline-level intent routing failures (Wang et al., 2025). Fundamentally, this is a failure in solvability assessment (Zhang et al., 2024). Consequently, *detection* leverages solvability-oriented benchmarks (Zhang et al., 2024), while *mitigation* involves calibrating use-thresholds (Qian et al., 2025) and formalizing abstention policies under tool insufficiency (Zhang et al., 2024).

Overuse/unnecessary tool use entails invoking tools when internal reasoning suffices, inflating latency and failure risks (Qian et al., 2025). Often compensating for boundary uncertainty or following rigid templates, mere tool availability can bias decision-making toward unnecessary calls (Winston and Just, 2025). Since end-task success often masks this inefficiency, *detection* requires overuse-aware evaluation (Qian et al., 2025). *Mitigations* adopt uncertainty-gated policies and supervision signals to systematically calibrate the boundary between reasoning and calling (Qian et al., 2025).

3.2 Multi-turn TSHs

In multi-turn execution, selection errors are rarely isolated. Because tool outputs and error messages are fed back into subsequent iterations, early mistakes can distort the agent's belief state, bias later planning, and induce compounding failures (Duan et al., 2025; Zhu et al., 2025). We highlight two dominant failure modes.

Error propagation in tool selection arises when an early wrong selection steers the trajectory into a self-reinforcing loop. Once an inappropriate tool is chosen, its output (or failure response) enters the context and may be treated as evidence, thereby shifting subsequent reasoning and rendering later selections seemingly consistent with a flawed premise (Duan et al., 2025). As context length increases, state tracking becomes noisier and the agent's ability to revisit or override earlier assumptions degrades, further amplifying selection drift (Zhu et al., 2025). This motivates *detection* that is explicitly trajectory-aware: uncertainty propagation analyses can reveal when earlier decisions destabilize later steps, indicating when verification is needed before committing to actions that would propagate error (Duan et al., 2025). In turn, *mitigation* emphasizes turning the execution loop into a recoverable process rather than a one-way trajectory, for example via reflection, rollback, and re-planning mechanisms that reduce compounding failures (Zhu et al., 2025).

Toolchain planning failure captures cases where agents fail to compose a correct sequence of tools (e.g., incorrect ordering, missing intermediate tools, or inability to complete multi-call procedures). Toolchains require global structure: if planning remains purely reactive, agents may lose intermediate information, misuse partial results, or converge to locally plausible but globally inconsistent sequences (Wei et al., 2025). Industrial observations similarly emphasize that reliable multi-call procedures require explicit decomposition and robust orchestration beyond prompt-only control (Wang et al., 2025). Accordingly, *detection and mitigation* are often planner-centric: frameworks that explicitly evaluate and improve toolchain structure reduce the burden on step-by-step prompting (Wei et al., 2025). In practice, toolchain reliability further benefits from transactional execution and validation layers that support safe retries and partial commits, strengthening robustness under real-world failures (Chang and Geng, 2025; Geng and Chang, 2025).

Whereas TSHs determine what external action is attempted, reliability in deployed systems also depends on how the selected tool is invoked. We next examine TCHs, which often dominate in pipelines requiring structured outputs, argument grounding, and post-call validation (Wang et al., 2025; Ma et al., 2025).

4 Tool Calling Hallucinations

TCHs occur after a tool has been selected and encompass errors in invocation structure, arguments, and execution handling (Lin et al., 2025; Wang et al., 2025). Relative to selection, calling is constrained by formal interfaces (schemas, signatures, and value constraints) and by environment-dependent state. Accordingly, failures range from explicitly malformed outputs (often detectable by validators) to semantically plausible yet incorrect

calls (which can evade surface checks), and multi-turn execution can further amplify these failures when tool feedback is misintegrated or stored into memory (Lin et al., 2024; Zhu et al., 2025; Xiong et al., 2025b).

We also organize TCHs into **single-turn** and **multi-turn** settings. Single-turn TCHs describe failure modes within an individual call attempt, whereas multi-turn TCHs capture how tool outputs become future inputs and beliefs, leading to repeated failures or persistent contamination.

4.1 Single-turn TCHs

Single-turn calling failures span schema/format mismatches, type/constraint violations, missing arguments, fabricated values, semantic drift, and absent fault tolerance (Wang et al., 2025; Ma et al., 2025). Requiring progressively stronger correctness – from syntactic to semantic validity – these failures motivate pipelines combining constrained generation, validation, and recovery.

Schema/format mismatch frequently occurs when free-form generation violates strict structured-output constraints (Wang et al., 2025; Chen et al., 2025). *Detection* utilizes structural validators and multi-stage verification to localize errors (Ma et al., 2025). For *mitigation*, constrained decoding restricts generation to syntactically valid states (Zhang et al., 2023a,b), complemented by staged checking and reflection for targeted repair (Ma et al., 2025).

Despite correct formats, *type/constraint mismatches* and *missing required arguments* persist. *Detection* relies on signature validation to expose type violations and missing fields, enabling targeted repair (Wang et al., 2025). *Mitigation* extends constrained decoding to enforce type-correct substructures and mandatory slots (Zhang et al., 2023b), while verification-driven retries bolster recovery against distribution shifts (Ma et al., 2025).

Harder failures emerge at the semantic level. *Fabricated values* involve context-inconsistent inputs (e.g., invalid IDs) within well-formed calls (Xiong et al., 2025a). Their *detection* necessitates state-dependent validators beyond schema compliance (Xiong et al., 2025a), while *mitigation* validates against allowed sets and applies feedback-driven reflection (Ma et al., 2025). Concurrently, *semantic drift under a valid schema* involves misusing parameter semantics or violating interface intents (Lin et al., 2024; Cui et al., 2025; Xu et al., 2025). Escaping structural checks, this drift is *mitigated* by reducing lexical bias via function masking (Lin et al., 2024) and enforcing intent consistency through semantic checklists and verifier agents (Cui et al., 2025; Ma et al., 2025).

Finally, *missing validation and fault tolerance* characterizes brittle post-call handling—failing to validate outputs or apply safe recovery, leading to erroneously trusted states (Wang et al., 2025). Consequently, output validation and robust fallbacks are critical (Wang et al., 2025). Runtime *mitigations* enforce stateful constraints and constraint-based backtracking for safe repair (Kamath et al., 2025; Vuddanti et al., 2025), while transactional frameworks offer commit/rollback semantics to ensure robust multi-step execution (Chang and Geng, 2025; Geng and Chang, 2025).

4.2 Multi-turn TCHs

In multi-turn settings, tool outputs (including failures) become future inputs, and calling hallucinations can persist because the agent misinterprets feedback, fails to update beliefs, or continues acting under a corrupted state (Zhu et al., 2025; Xiong et al., 2025a). As multi-turn contexts grow, state misunderstandings and incorrect integration of tool feedback become dominant sources of repeated invalid calls or cascading argument drift (Zhu et al., 2025). Accordingly, mitigation increasingly relies on execution-time structure: transactional controllers with structured state logs support auditing, rollback, and stable re-planning, thereby reducing cascading errors across iterations (Chang and Geng, 2025; Geng and Chang, 2025). From an evaluation standpoint, robustness benchmarks that inject tool failures and measure recovery behaviors are crucial for exposing propagation vulnerabilities that are otherwise masked by end-task metrics (Xiong et al., 2025a).

A particularly damaging multi-turn failure is *belief/memory contamination*, where incorrect tool outputs or self-generated contexts are written into memory and later retrieved, turning transient mistakes into persistent beliefs (Xiong et al., 2025b). Empirical studies show that memory management choices substantially influence whether agents repeat or correct past errors (Xiong et al., 2025b). Mitigation therefore requires auditable memory actions (e.g., selective writes and versioned updates) and validation-conditioned storage, ideally integrated with execution-time verification (Ma et al.,

2025; Chang and Geng, 2025).

The taxonomy above clarifies how hallucinations arise in selection and calling. However, methodological progress is ultimately constrained by evaluation: benchmarks must operationalize these failure modes and expose them reliably, especially when intermediate errors are later corrected. We next review benchmarks for tool execution hallucinations and discuss how benchmark design choices determine which hallucinations are observable and which interventions appear effective.

5 Benchmarks for Tool Execution Hallucinations

Benchmark design dictates which hallucinations are observable. In tool-augmented agents, hallucinations are often intermediate: agents may commit incorrect actions and recover later, or fail silently through semantic misuse despite syntactically valid outputs (Xiong et al., 2025a; Zhu et al., 2025). Consequently, benchmarks focusing solely on end-task success underdiagnose stage-specific errors, whereas those exposing execution traces and perturbations make them observable.

Early benchmarks primarily assess baseline tool-use competence under controlled conditions. API-Bank (Li et al., 2023) is a representative large-scale benchmark covering realistic APIs and tasks that require deciding whether to use tools, selecting appropriate APIs, and generating valid parameters, thereby implicitly revealing common tool selection and tool calling errors. While these benchmarks established feasibility, they also motivated evaluations that more explicitly diagnose decision-stage failures.

A first class of benchmarks emphasizes boundary decisions of "whether/which tool" to use, directly probing single-turn TSHs. MetaTool evaluates whether and which tools should be invoked (Huang et al., 2023). ToolBeHonest explicitly diagnoses solvability and hallucinations under missing-tool settings (Zhang et al., 2024). SMART targets overuse by quantifying unnecessary tool calls (Qian et al., 2025). By explicitly labeling tool necessity and availability, these benchmarks move beyond end-task success and enable faithful assessment of underuse and overuse.

A second class is trajectory- and state-aware, surfacing multi-turn propagation and recovery behaviors. Toolsandbox provides stateful evaluation (Lu et al., 2025), and the Berkeley Function Calling Leaderboard (BFCL) highlights environment state misunderstandings in multi-turn settings (Patil et al., 2025). TRAJECT-Bench, MIRAGE-Bench, and Tool Playgrounds further localize failures along execution traces (He et al., 2025; Zhang et al., 2025a; Dong et al., 2025). Treating traces as first-class objects makes them essential for evaluating multi-turn TSHs and TCHs.

A third evaluation perspective focuses on robustness and stability under perturbations. Stability-oriented benchmarks deliberately inject failures such as wrong tool choices, malformed calls, or erroneous values, and measure whether agents can recover (Xiong et al., 2025a). This perspective is critical because local validity can mask global failure: schema-correct calls may remain semantically incorrect, and without perturbation or trace-level supervision, such failures are systematically underdiagnosed.

Taken together, existing benchmarks expose two fundamental challenges. First, *error propagation* dominates stateful execution: early mistakes bias subsequent decisions and are unlikely to be corrected without explicit recovery mechanisms (Duan et al., 2025; Zhu et al., 2025). Second, *solvability and tool-necessity awareness* underlies both overuse and underuse, yet remains weakly captured unless evaluations explicitly supervise boundary decisions and safe abstention (Zhang et al., 2024; Qian et al., 2025).

6 Findings and Future Directions

6.1 Key Findings

Viewing tool execution hallucinations through the agent execution flow yields several consistent findings that recur across models, tool ecosystems, and evaluation settings.

Boundary and grounding errors dominate over syntactic failures. Across the literature, the most consequential hallucinations arise not from malformed outputs but from incorrect judgments about tool necessity, tool choice, and argument grounding. Agents therefore frequently produce schema-valid yet semantically incorrect calls, or omit tool use when external execution is required (Lin et al., 2024; Cui et al., 2025; Zhang et al., 2024; Qian et al., 2025). This explains why advances in constrained decoding alone provide limited gains in end-to-end reliability (Zhang et al., 2023a,b).

Error propagation is intrinsic to iterative tool use, making recoverability central. In multi-turn ex-

ecution, early mistakes are recursively incorporated into subsequent reasoning, leading to trajectory-level drift and compounding failures unless explicit recovery mechanisms are present (Duan et al., 2025; Zhu et al., 2025). Empirical results consistently show that robustness hinges less on one-shot correctness than on the ability to detect instability, revise commitments, and prevent unverified intermediate states from polluting belief and memory (Chang and Geng, 2025; Geng and Chang, 2025; Xiong et al., 2025b). As a result, validation, rollback, and auditable state or memory updates emerge as first-order design requirements rather than auxiliary safeguards.

Evaluation protocols lag behind deployment-relevant failure modes. Benchmarks that emphasize final task success systematically underdiagnose intermediate hallucinations and silent semantic misuse that pass structural checks (Xiong et al., 2025a; Zhu et al., 2025). Recent trajectory-aware and perturbation-based benchmarks demonstrate that faithful evaluation requires exposing execution traces and explicitly measuring failure localization and recovery, particularly under missing-tool and adversarial conditions (He et al., 2025; Zhang et al., 2025a, 2024; Qian et al., 2025). This gap in measurement partially accounts for inconsistent conclusions across existing benchmarks.

6.2 Future Directions

Grounded in the above findings and taxonomy, we highlight three directions that directly address the dominant sources of tool execution hallucinations.

Solvability-aware and cost-calibrated tool-use decisions. Both tool underuse and overuse stem from weak awareness of task solvability and tool necessity (Zhang et al., 2024; Qian et al., 2025). Future agents should explicitly model whether a task requires external execution, whether the available toolset is sufficient, and when abstention or clarification is preferable under uncertainty (Zhang et al., 2024). Integrating solvability estimation with explicit cost and risk modeling (e.g., latency, failure surface, and safety) is essential to make tool invocation a calibrated decision rather than a default behavior (Qian et al., 2025; Winston and Just, 2025).

Semantics-first reliability with execution-time guarantees. Many harmful failures are locally valid yet semantically incorrect, including intent-misaligned calls and fabricated but plausible arguments (Lin et al., 2024; Cui et al., 2025; Xiong et al., 2025a). Addressing these failures requires prioritizing semantic validity through intent-aware invocation, state-grounded arguments, and post-call validation against task constraints (Cui et al., 2025; Ma et al., 2025). At the system level, transactional execution with validation and commit/rollback semantics enables recoverable multi-step tool use (Chang and Geng, 2025; Geng and Chang, 2025), while validation-conditioned and auditable memory writes prevent persistent belief contamination (Xiong et al., 2025b; Ma et al., 2025). Runtime verification further provides enforceable consistency and safety guarantees during execution (Kamath et al., 2025; Cruz, 2025).

Failure-centric and trajectory-aware evaluation. Progress in tool-augmented agents is constrained by what benchmarks reveal. Evaluations focused solely on final success obscure intermediate hallucinations, recovery capability, and cross-stage interactions between selection and calling (Xiong et al., 2025a; Zhu et al., 2025). Future benchmarks should therefore emphasize where and how agents fail, using trajectory-level annotations, targeted perturbations aligned with hallucination types, and metrics that reward recovery and safe abstention (He et al., 2025; Zhang et al., 2025a; Xiong et al., 2025a), thereby aligning optimization objectives with deployment-relevant robustness (Duan et al., 2025; Ma et al., 2025).

7 Conclusion

This survey studied *tool execution hallucinations* in LLM-based agents from an execution-centric perspective, grounding hallucination phenomena in an iterative agent execution flow. Focusing on tool selection and tool calling under both single-turn and multi-turn settings, we proposed a unified taxonomy that characterizes where erroneous decisions arise and how they propagate during execution. We further synthesized detection and mitigation strategies and connected them to representative benchmarks, revealing persistent challenges such as error propagation across iterations, weak solvability and tool-necessity awareness, semantic misuse beyond syntactic validity, and fragile state and memory handling. We argue that grounding hallucination analysis in the agent execution flow provides a principled basis for diagnosing failures and guiding the development of more reliable and robust tool-augmented LLM agents.

8 Limitations

While our taxonomy systematically covers tool selection and calling, it deliberately omits hallucinations from pure task planning or multi-agent communication to maintain analytical depth. Additionally, the rapid evolution of foundation models and integration standards (e.g., the Model Context Protocol) means our synthesis inherently represents a temporal snapshot. Finally, the prevalent use of closed-source models restricts mechanistic interpretability, limiting most current detection and mitigation strategies – including those reviewed here – to black-box probing and behavioral observation rather than internal dynamic analysis. However, this survey may have dual-use implications: a taxonomy of tool hallucination can support mitigation, but it may also inform attempts to induce tool-selection errors, invalid tool calls, or misleading tool-use trajectories. We therefore frame failure modes at a high level and emphasize evaluation and mitigation rather than operational attack recipes. We further caution that current defenses should not be treated as complete safeguards in high-stakes tool-use deployments.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Edward Y Chang and Longling Geng. 2025. Sagallm: context management, validation, and transaction guarantees for multi-agent llm planning. *arXiv preprint arXiv:2503.11951*.

Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2025. Octopus: On-device language model for function calling of software apis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 329–339.

Christopher Cruz. 2025. Vigil: A reflective runtime for self-healing agents. *arXiv preprint arXiv:2512.07094*.

Yue Cui, Liuyi Yao, Shuchang Tao, Weijie Shi, Yaliang Li, Bolin Ding, and Xiaofang Zhou. 2025. Enhancing tool learning in large language models with hierarchical error checklists. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16357–16375.

Zhiwei Dong, Ruihao Gong, Yang Yong, Shuo Wu, Yongqiang Yao, Song-Lu Chen, and Xu-Cheng Yin. 2025. Tool playgrounds: A comprehensive and analyzable benchmark for llm tool invocation. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. 2025. Uprop: Investigating the uncertainty propagation of llms in multi-step agentic decision-making. *arXiv preprint arXiv:2506.17419*.

Longling Geng and Edward Y Chang. 2025. Alas: Transactional and dynamic multi-agent llm planning. *arXiv preprint arXiv:2511.03094*.

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, and 1 others. 2025. Traject-bench: A trajectory-aware benchmark for evaluating agentic tool use. *arXiv preprint arXiv:2510.04550*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and 1 others. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. 2025. Enforcing temporal constraints for llm agents. *arXiv preprint arXiv:2512.23738*.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 3102–3116.

Wenjun Li, Dexun Li, Kuicai Dong, Cong Zhang, Hao Zhang, Weiwen Liu, Yasheng Wang, Ruiming Tang, and Yong Liu. 2025. Adaptive tool use in large language models with meta-cognition trigger. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13346–13370.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, and 1 others. 2024. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*.

Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Kun Wang, and 1 others. 2025. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv preprint arXiv:2509.18970*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, and 1 others. 2025. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.

Zhiyuan Ma, Jiayu Liu, Xianzhen Luo, Zhenya Huang, Qingfu Zhu, and Wanxiang Che. 2025. Advancing tool-augmented large language models via meta-verification and reflection learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 2078–2089.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. 2025. Smart: Self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4604–4621.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

SMTI Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 6.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Sri Vatsa Vuddanti, Aarav Shah, Satwik Kumar Chittiprolu, Tony Song, Sunishchal Dev, Kevin Zhu, and Maheep Chaudhary. 2025. Paladin: Self-correcting language model agents to cure tool-failure cases. *arXiv preprint arXiv:2509.25238*.

Maolin Wang, Yingyi Zhang, Bowen Yu, Bingguang Hao, Cunyin Peng, Yicheng Chen, Wei Zhou, Jinjie Gu, Chenyi Zhuang, Ruocheng Guo, and 1 others. 2025. Function calling in large language models: Industrial practices, challenges, and future directions. *ACM Computing Surveys*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xiaolong Wei, Yuehu Dong, Xingliang Wang, Xingyu Zhang, Zhejun Zhao, Dongdong Shen, Long Xia, and Dawei Yin. 2025. Beyond react: A planner-centric framework for complex tool-augmented llm reasoning. *arXiv preprint arXiv:2511.10037*.

Cailin Winston and René Just. 2025. A taxonomy of failures in tool-augmented llms. In *2025 IEEE/ACM International Conference on Automation of Software Test (AST)*, pages 125–135. IEEE.

Christian Woesle, Leopold Fischer-Brandies, and Ricardo Buettner. 2025. A systematic literature review of hallucinations in large language models. *IEEE Access*.

Weimin Xiong, Ke Wang, Yifan Song, Hanchao Liu, Sai Zhou, Wei Peng, and Sujian Li. 2025a. More vulnerable than you think: On the stability of tool-integrated llm agents. *arXiv preprint arXiv:2506.21967*.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. 2025b. How memory management impacts llm agents: An empirical study of experience-following behavior. *arXiv preprint arXiv:2505.16067*.

Hongshen Xu, Zihan Wang, Zichen Zhu, Lei Pan, Xingyu Chen, Shuai Fan, Lu Chen, and Kai Yu. 2025. Alignment for efficient tool calling of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17787–17803.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuan-Jing Huang. 2024. Toolsword: Unveiling safety issues of large language models in tool learning across three stages. In *Proceedings of the*

62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2181–2211.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. 2023a. Don't fine-tune, decode: Syntax error-free tool use via constrained decoding. *arXiv preprint arXiv:2310.07075*.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Yang Wang. 2023b. Tooldec: Syntax error-free and generalizable tool use for llms via finite-state decoding.

Weichen Zhang, Yiyou Sun, Pohao Huang, Jiayue Pu, Heyue Lin, and Dawn Song. 2025a. Mirage-bench: Llm agent is hallucinating and where to find them. *arXiv preprint arXiv:2507.21017*.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, and 1 others. 2024. Toolbehonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11388–11422.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025b. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Kaitlyn Zhou, Jena Hwang, Xiang Ren, and Maarten Sap. 2024. Relying on the unreliable: The impact of language models' reluctance to express uncertainty. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3623–3643.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, and 1 others. 2025. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*.