

Step-Level Hallucination Detection and Error Propagation Analysis in Agentic LLM Systems

Ayurshi Patil¹ Sania Raut² Satwik Kale³ Darpan Thakare⁴

^{1,2,3,4}Student

^{1,2,3,4}Department of Computer Science and Engineering

^{1,2,3,4}Professor Ram Meghe Institute of Technology and Research, Badnera, Maharashtra, India

Abstract — Large Language Models (LLMs) are increasingly used in agentic systems that plan, retrieve, execute, remember, and verify complex workflows involving multiple steps, tools, memory, and intermediate states. In such contexts, errors in intermediate states can be propagated to later steps, and thus to the final answer, when used as part of the state. This work conducts a survey on hallucination detection and error propagation challenges at step, trajectory, and workflow levels. A focused set of ten primary sources is synthesized across three areas of the literature: hallucination detection and mitigation, agentic error propagation, and agentic evaluation and governance. The survey organizes and compares related work by layer of detection, granularity, and methodology, highlighting similarities and differences in output- and step-level detection, error propagation, attribution, evaluation, mitigation, and governance approaches. Notions of error amplification, memory contamination, trajectory attribution, DAG-based dependency modelling, and the distinction between identification of a responsible step and its use for system improvement are discussed. The synthesis outlines several open challenges, notably the lack of a shared benchmark for step-level detection and the diversity of terminology, and provides quantitative evidence of model-specific error propagation properties. It also emphasizes the need to address cost-latency trade-offs, non-DAG workflow architectures, and the integration of technical and governance aspects of hallucination-related system failures.

Keywords: Large Language Models, Agentic AI, Hallucination Detection, Step-Level Detection, Error Propagation, Trajectory Analysis, Agent Evaluation, Memory Contamination, Root Cause Attribution, AI Governance

I. INTRODUCTION

Large Language Models have evolved from single-response generation to planning, reasoning, and tool use with memory and environment interaction. A conventional LLM can often be assessed by comparing the final response to a reference or external facts. An agentic LLM, on the other hand, produces a series of intermediate states and actions. The plausibility of the final outcome therefore depends not just on the final generation but also on the accuracy of the intermediate trajectory.

This distinction defines the problem studied in this paper. Hallucination during retrieval, planning, tool selection, tool argument generation, memory retrieval, or reasoning can be incorporated into subsequent reasoning steps. The resulting error may propagate, amplify, mask, reinforce through memories, or be corrected before final output. Recent work therefore focuses on trajectory analysis, step-level attribution, DAG-structured evaluation, and explicit operational measures for error propagation ([3]; [4]; [7]; [5]).

The objective of this survey is to provide an overview of this area around a central question: *how should hallucinations and errors be detected, attributed, and contained, when an LLM is used as a multi-step agent rather than a single-turn generator?* The paper addresses four research questions (RQs):

- RQ1: What detection methods and granularities are available for hallucinations in LLM and agentic systems?
- RQ2: How does errors propagate, amplify, persist, or get contaminated through multi-step execution and memory?
- RQ3: How could a hallucination be attributed to a responsible step, agent or dependency in an execution trajectory?
- RQ4: What evaluation, mitigation, and governance mechanisms can detect and contain failures before becoming system-level outcomes?

The survey identifies four contributions within the ten-source corpus: (1) a taxonomy that classifies NLP fact-checking tools along detection layers, granularity, and methodology; (2) a comparative analysis of output-level versus step-level detection, propagation, attribution, and evaluation; (3) a proposed three-layer evaluation view covering output, reasoning, and topology; and (4) a set of mitigation directions based on the surveyed corpus's gaps. The three-layer view and mitigation directions represent the authors' synthesis rather than established methods.

II. SURVEY METHODOLOGY

A. Scope

The survey encompasses agentic large language models (LLMs) that include single and multi-agent systems that involve planning, tool use, retrieval, persistent or episodic memory, intermediate reasoning, verification, and execution. The survey scope explicitly focuses on the step-level detection and error propagation rather than hallucination in single-turn generation. The four research questions above form the scope of this survey.

B. Source Selection

There were ten primary sources identified using specific keyword searches followed by citation-chasing in three strands: (i) hallucination detection and mitigation, (ii) agentic error propagation, and (iii) agentic evaluation and governance. The corpus contains the sources mentioned in the reference section. This survey scopes a focused corpus rather than an extensive database search; the coverage is limited by the ten sources mentioned in the reference list.

Search process and reproducibility. Candidate sources were located primarily via Google Scholar, using exploratory keyword searches on the themes of hallucination detection in LLMs, step- and trajectory-level error

propagation in agentic systems, failure attribution, agent evaluation, memory contamination, and governance of agentic systems. These were followed up by citation chaining from the retrieved papers. A formal search log was not kept: exact boolean strings, per-database result tallies, and exact search dates were not noted, and the searches took place during 2026. The corpus is thus a purposive, non-exhaustive selection and the search cannot be replicated in detail. This limitation is declared explicitly, and no systematic-review protocol (such as PRISMA) is claimed.

C. Inclusion and Exclusion Criteria

Sources were included if they described peer reviewed papers, preprints, or theses, which studied hallucination, propagation, attribution, evaluation, mitigation, or governance in any agentic or multi-step large language model (LLM) scenarios. Excluded material comprised papers studying single-turn hallucinations with no step-level or agentic component, non-LLM agents, and papers with a primary contribution focused solely on applications, and therefore not directly relevant to reliability analysis.

D. Categorisation

Each source is considered along three analytical axes:

- 1) Layer: detection, propagation, attribution, evaluation, or governance;
- 2) Granularity: output, step, or trajectory;
- 3) Methodology: observational, controlled, or framework-oriented.

Table I applies the three axes to the ten-source corpus.

Source	Layer	Granularity	Methodology
Ghosh [1]	Detection	Output / trajectory	Observational / framework

Kazlaris et al. [2]	Mitigation	Output	Observational / framework
Zamindar [3]	Propagation	Trajectory	Observational / scoping review
Wang et al. [4]	Attribution	Trajectory	Observational / survey
Shaikh [5]	Propagation	Step / trajectory	Controlled
Bansal et al. [6]	Propagation	Step	Controlled
Guo et al. [7]	Evaluation	Step / DAG	Framework / controlled evaluation
Yehudai et al. [8]	Evaluation	System / node / trace	Framework / observational evaluation
Chang et al. [9]	Evaluation / topology	Output / reasoning / topology	Observational / survey
Valdez-Cantú et al. [10]	Governance	System / trajectory	Observational / structural review

Table I: Classification of the Ten Primary Sources

E. Survey Flow

The review process is summarized in Fig. 1. Due to the lack of a screening log, numbers of records at the stages of identification, screening, and eligibility are not given. Hence, the figure represents a focused-corpus construction workflow rather than the PRISMA-style flow diagram.

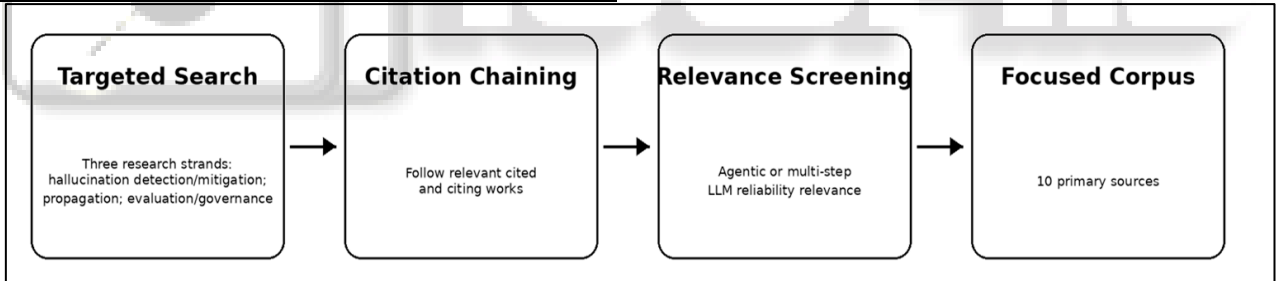


Fig. 1: Focused corpus construction workflow based on targeted keyword searching, citation chaining, relevance screening, and inclusion of ten primary sources.

F. Limitations of the Survey Method

The ten-source corpus was curated by a single reviewer; this refers to the literature-screening and categorisation process, and not the number of authors of the paper. The terminology used across the corpus is heterogeneous. The corpus further consists of a mix of publication venues and publication statuses. The search process was not logged and cannot be reproduced exactly (Section II-B). No quantitative meta-analysis is attempted since the sources employ different datasets, models, operational definitions, evaluation units, and metrics.

III. BACKGROUND

A. Hallucination in LLMs

Hallucination is used in the literature for generative content that takes the form of being false, unsubstantiated, inconsistent with available evidence, or unfaithful to the given context. Ghosh [1] organizes the methods for detecting it around a multidimensional framework that includes the following items: the signal for detection, the model's access, the use of external knowledge, the level of detection, the computational costs, and the ways of structural failure. The hallucination taxonomy could be interpreted from the perspective of both factuality and faithfulness.

An intrinsic/factuality hallucination is a response that contradicts known facts or objective truth. An extrinsic/faithfulness hallucination is when the response

strays from the source or context that it was supposed to stay true to. The corpus also talks about fabrication and context errors, knowledge boundary errors, logical inconsistencies, citation/attribution failures, temporal or out-of-date information, and domain-specific errors. These are not necessarily exclusive categories either, a single step could involve fabrication of a tool argument, violation of retrieved evidence, and then an incorrect final claim could be made as a result.

B. Agentic Architectures

A multi-step agent can be thought of as an execution loop, in which a language model is embedded in a stateful execution

flow. Zamindar's sixteen stage scheme [3] gives an operational breakdown, which we can take inspiration from:

User input → Perception → Intent interpretation → Reasoning → Planning → Task decomposition → Query formulation → Retrieval/grounding → Memory write → Memory retrieval → Tool selection → Tool arguments → Execution → Observation/feedback → Verification/recovery → Final outcome.

The sixteen-stage workflow is complemented by the error-cascade representation in Fig. 2, which makes the propagation path explicit.

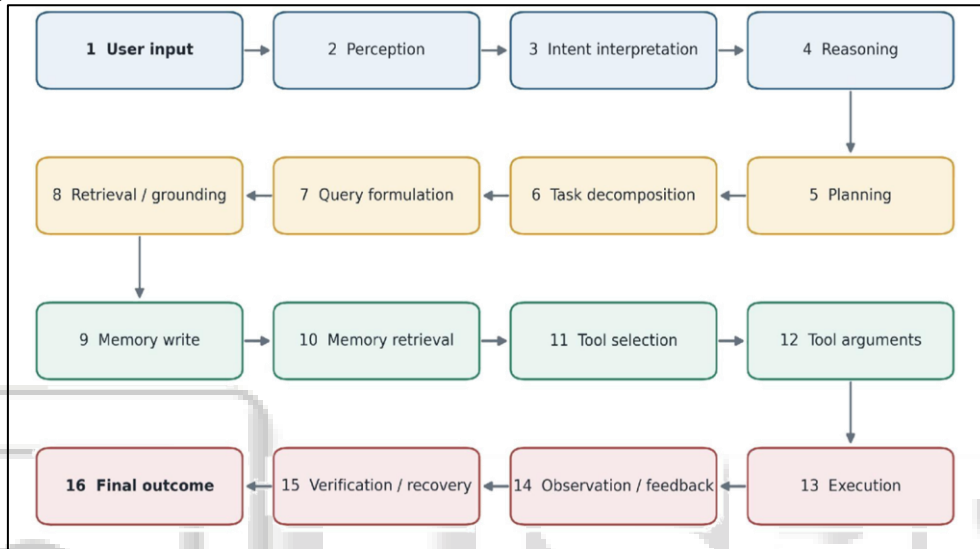


Fig. 2: Sixteen-stage agent error-cascade pipeline. Adapted from Zamindar [3].

The planner-tool-memory-verifier architecture is shown in Fig. 3. It highlights three entry points for faults: reasoning faults, tool faults, and memory faults.

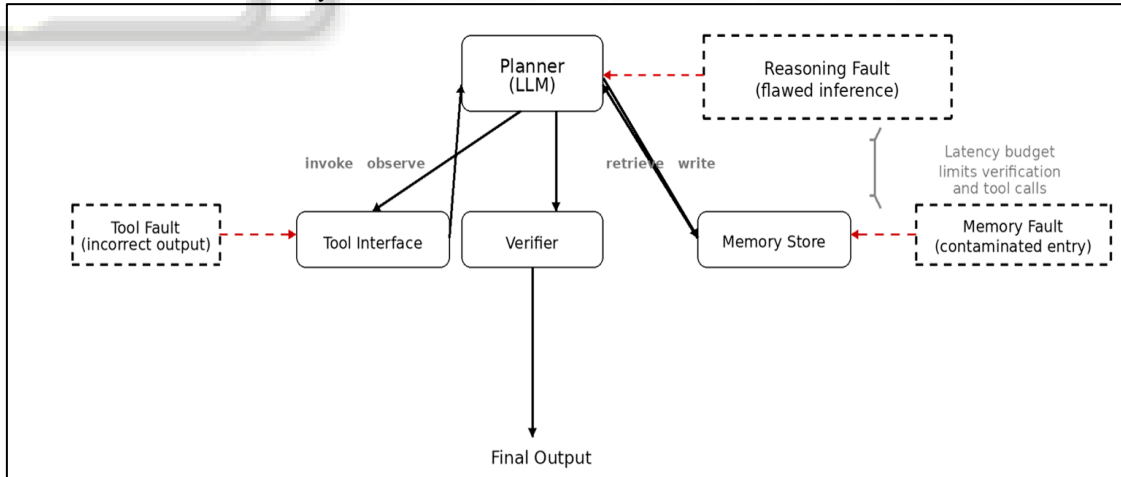


Fig. 3: Tool-using agent architecture showing planner, tool interface, memory store, verifier, final output, and three major error sources. Adapted from Shaikh [5].

C. Why Multi-Step Execution Changes the Failure Mode

In a single-turn setting, incorrect output is primarily an output-level failure, but in an agentic setting, an intermediate error can become *state*, influencing subsequent retrieval, planning, tool selection, arguments, and execution. That is, the causal structure of the system has changed; errors are no longer necessarily independent observations. The final-

output check is also structurally late: it may find that the final result is wrong, but not what the first step that introduced the error was. This is why we are motivated to do step-level detection and trajectory attribution.

D. Terminology

For the purpose of consistency across heterogeneous sources this paper uses the following operational terminology:

- Hallucination: generated content that is false, unsupported, or unfaithful with respect to the relevant evidence or context.
- Error: any incorrect intermediate state, action, decision, tool interaction, or output relative to the task specification.
- Propagation: transmission of the effect of an upstream error to one or more downstream steps.
- Amplification: an increase in downstream error risk or affected steps following an upstream error.
- Contamination: persistence of erroneous information in state or memory so that it can influence later execution or episodes.
- Attribution: localization of an observed failure to a responsible step, agent, component, or dependency.
- Containment: preventing an error from affecting downstream execution.
- Recovery: correction or neutralization of an error after it has entered the workflow.

IV. TAXONOMY OF DETECTION METHODS

A. Output-Level Detection

Ghosh [1] classifies hallucination detection algorithms into nine major families, namely intrinsic, sampling-based, semantic-consistency, knowledge-based, retrieval-based, reference-based, LLM-as-a-judge/verifier-based, multi-agent, and hybrid methods. The characteristics of these families are summarized in Table II.

The families mainly differ in the evidence they consume and what they cost. Under Ghosh’s relative-cost

abstraction, intrinsic detectors require approximately one forward pass; sampling-based and semantic-consistency detectors scale with the number of sampled generations (typically five to ten), retrieval-based detectors add retrieval, re-ranking, and verification calls, judge-based detectors add approximately one additional model call per verdict, and multi-agent detectors scale with the number of agents and rounds of interaction [1]. Intrinsic methods additionally require access to logits or hidden states, which closed API-only models do not provide [1].

Two structural limitations recur across the families [1]. First, any family that uses agreement as a decision criterion, across samples, paraphrases, or agents, is vulnerable to systematic error, as a model that is wrong consistently will pass an agreement check. Second, any family that makes use of an auxiliary model like an entailment model, judge, or verifier, incorporates that model’s knowledge gaps and biases. Judge models in particular exhibit position, verbosity, and self enhancement bias. Hybrid designs that use an agreement signal plus an external check can target complementary blind spots.

For agentic systems these detectors suffer from an additional limitation, that the unit of analysis is a completed response. A final output examination can identify that a result is wrong without pinning down the step when the error was introduced [3]. AgentEval shows the consequence of this constraint: its end-to-end baseline only analyses the final synthesis step, meaning errors in preceding steps are only detected if they visibly impact the final output, and it achieved a failure detection recall of 0.41 compared to 0.89 for DAG-structured step-level evaluation [7].

Family	Signal	External knowledge	Model access	Cost	Structural blind spot
Intrinsic	Uncertainty/representations	No	Often internal	Low	Can miss systematic errors
Sampling	Cross-sample agreement	No	Often black-box	Medium	Expensive for long traces
Semantic consistency	Meaning agreement	Usually no	Black-box/partial	Medium	Agreement can preserve shared error
Knowledge-based	Knowledge consistency	Optional	Varies	Medium	Knowledge coverage/bias
Retrieval-based	Evidence support	Yes	Varies	High	Retrieval quality becomes a dependency
Reference-based	Reference alignment	Yes	Usually black-box	Medium	Requires reliable reference
LLM-as-judge	Judge verdict/rubric	Optional	Black-box/partial	Medium–High	Judge bias/circularity
Multi-agent	Debate/consensus	Optional	Multiple models/agents	High	Correlated agent errors
Hybrid	Combined signals	Varies	Varies	High	Integration and calibration complexity

Table II: Output-Level Hallucination Detection Families

B. Step-Level Detection

Reapplication of an output-level detector to each agent step is not a validated step-level detector. The signal of such a detector has to be calibrated to the semantics of the step, its available context, and its dependency structure. For instance, a retrieval step may be evaluated on the relevance of evidence, a tool-argument step on schema and intent fidelity,

and a synthesis step on how much it is supported by prior observations. As a result, aggregation across steps implies the need for explicit step boundaries and a characterization of failure at each boundary.

C. Granularity Axis

→→→→→Ghosh’s second axis in the taxonomy [1] goes from finer to coarser detection units: token span sentence

atomic claim citation/attribution passage/response reasoning trajectory. The latter is especially interesting for agents as it allows the detector to reason over dependencies between intermediate steps.

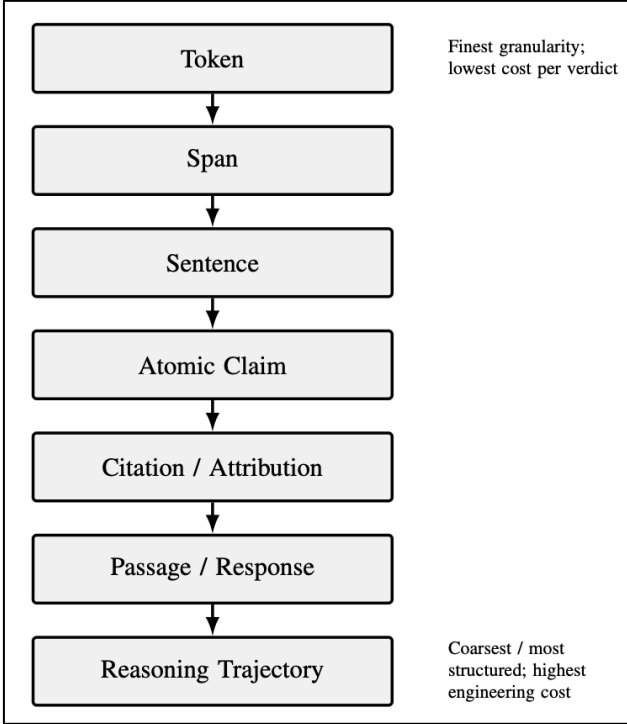


Fig. 4: Detection granularity across hallucination detection systems, from token-level detection to reasoning-trajectory analysis. Adapted from Ghosh [1].

D. Evidence Traceability

Evidence-traceability is amenable to reuse at the step level by associating evidence with each intermediate claim or action, as opposed to just the final response. Ghosh's multidimensional framework [1] provides the detection-signal axis, while the mitigation taxonomy of Kazlaris et al. [2] enables distinction between interventions that modify training, architecture, prompts, post-generation verification, interpretability, or agent orchestration. At the step level, this associates an evidence chain with a decision that grounds it in the context, retrieval result, memory item, or tool observation that supports it.

E. Output-Level versus Step-Level Detection

Table III contrasts the two evaluation units.

Axis	Output-level	Step-level
Unit	Final response	Intermediate action/reasoning step
Localization	Limited	Explicit step localization
Propagation	Usually hidden	Directly observable
Context	Final context	Step-specific context and dependencies
Cost	Lower	Higher due to repeated evaluation
Actionability	Detects final failure	Supports intervention before final output
Blind spot	Late detection	Step-boundary and calibration problems

Table III: Output-Level versus Step-Level Hallucination Detection

V. ERROR PROPAGATION

A. Operational Definitions and Metrics

The paper by Shaikh [5] introduces a reliability framework for tool-using LLM agents, based on metrics related to fault spread, persistence, cumulative divergence, and memory contamination. The metrics used and considered in this survey are as follows:

- State deviation: deviation of the current state from the corresponding fault-free trajectory.
- Error indicator: binary or graded indicator that a step is incorrect.
- Amplification score: extent to which an injected or upstream fault increases downstream error.
- Persistence length: number of downstream steps for which an error remains influential.
- Cumulative divergence: accumulated deviation across affected steps.
- Cross-episode amplification: downstream impact of a contaminated memory state across episodes.
- False-memory rate: proportion of memory writes that encode erroneous information.
- Retrieval reuse rate: proportion of contaminated memory entries that are subsequently retrieved and reused.

B. Propagation Patterns

Zamindar's scoping review [3] identifies ten propagation patterns. Table IV summarizes them from the perspective of detection.

Pattern	Definition	Corpus example	Detection implication
Linear	Error affects a sequence of dependent steps	Incorrect reasoning conditions the next action	Track sequential dependency
Branching	One error influences multiple downstream branches	Faulty plan affects multiple sub-tasks	Detect common upstream causes
Convergent	Multiple paths feed a common erroneous state	Independent errors meet in synthesis	Compare dependency paths
Feedback amplification	Output is fed back into later decisions	Incorrect observation changes the next plan	Monitor recurrent state transitions
Memory contamination	Error is written to persistent memory	False memory entry is retrieved later	Audit writes and retrievals
Retrieval reinforcement	Repeated retrieval strengthens an error	Same incorrect memory repeatedly reused	Track reuse frequency

Cross-agent spread	One agent's error affects another agent	Shared state transfers a false result	Trace inter-agent dependencies
Silent propagation	Error persists without immediate visible failure	Intermediate wrong value appears plausible	Require local verification
Masking	Downstream output hides an upstream error	Later step produces a superficially correct result	Preserve causal trace evidence
Recovery failure	Correction mechanism does not neutralize the fault	Verification misses an incorrect intermediate step	Measure recovery and containment

Table IV: Error Propagation Patterns and Detection Implications

C. Empirical Evidence

$A(\text{early}) = 0.412A(\text{late}) = 0.583U = 0.5893.95 \times$
Shaikh's controlled fault injection experiments on Claude Sonnet 4.6 found an injection position-dependent amplification ranging from $A(\text{early}) = 0.412$ to $A(\text{late}) = 0.583$ [5]. The same work also found cross-episode memory contamination: a memory reuse rate $U = 0.589$ and step-level error rate approximately $3.95 \times$ the no-memory baseline. Note that these are model/experiment-specific rather than universal population estimates.

$AFp_{\text{post}} \approx 0.5$ The complementary cascading-failure study reports an Amplification Factor (AF) of 7.79 for GPT-4o-mini, 3.29 for Llama-3.1-8B and 7.01 for Qwen-2.5-72B [6]. It reports $p_{\text{post}} \approx 0.5$ across the three model families (i.e., approximately half of post-error steps contained new errors under that study's operational definition). The same study used 14,558 step-level records across 2,119 traces and a controlled injection experiment (providing evidence for cascading amplification, but still limited to the models, dataset and error-injection design used in that study)

D. Memory Contamination

$U = 0.589$ Memory changes the scope in time of an error: without memory the erroneous intermediate result may affect only the current execution, whereas with persistent memory, the same erroneous result may be reused in a future execution as an input. Shaikh reports a false-memory rate reduction of 52.8% with confidence-gated memory writes and a latency overhead of approximately 7.9%. The observed rate of reuse $U = 0.589$ demonstrates the rationale to consider memory as a propagation medium rather than a mere storage facility.

E. Recovery

Recovery is not the inverse of amplification. Shaikh reports a recovery rate of approximately 8% for early faults and 41% for late faults in the studied Claude Sonnet 4.6 setting. The paper attributes the pattern to the quantity of coherent context required to detect contradictions. A late fault can thus be corrected more frequently while an uncorrected late fault can impact a higher fraction of the remaining execution window.

F. Divergence Across Sources

Zamindar's scoping review and Shaikh's controlled experiments agree on only two points: propagation is an important characteristic of multi-step execution, and memory can act as a storage medium. They differ in several key points, so the results should be treated as complementary. Zamindar's findings are based on heterogeneity, and individual studies cannot be considered as prevalence

estimates for all types of agents. Shaikh's experiments, in turn, provide specific measurements for individual models with artificially induced failures.

VI. ATTRIBUTION AND TRAJECTORY ANALYSIS

Wang et al.'s survey [4] classifies the failure attribution methods into four types according to their methodology: pattern-based methods, LLM-reasoning-based methods, fine-tuned tracer models, and dynamic runtime-based methods, which mainly differ in the observations they use and the identification of the responsible step.

A. Step-Level versus Agent-Level Attribution

Agent-level attribution finds a faulty agent or subsystem while step-level attribution asks which concrete event first introduced or materially contributed to the failure. Step-level attribution is more useful for debugging since it can identify the point of containment but it can be harder to perform due to scattered evidence and later steps can obscure the original failure.

B. Attribution Accuracy

Wang et al. note that step-level attribution accuracy is largely limited within the assessed settings. The survey then discusses the Who & When results, which are found to be within the 25–52 range. Ghosh's review discusses AgentHallu's results, where the responsible-step localization reaches 41.1 across all steps and 11.6 in the case of tool-use hallucinations, thus proving that responsible-step localization remains largely limited. As the cited results make no reference to a final-output detection baseline that would match step localization, the accuracy of the latter cannot be directly compared to the accuracy of the former in this research.

C. Attribution-to-Enhancement Gap

A key finding in Wang et al. is the attribution-to-enhancement gap: while attribution methods can identify a responsible agent or step, enhancement methods are not obliged to use this information to improve the system. In other words, a closed-loop reliability chain requires more than just localization of the failure cause: it requires a mechanism that turns this information into action, verification, containment, correction, and recovery.

D. Attribution Paradigms

Paradigm	Input	Output	Cost	Characteristics
Pattern-based	Trace patterns/features	Responsible step/agent	Low–Medium	Relies on recurring failure signatures
LLM-reasoning-based	Execution trace + prompt	Natural-language attribution	Medium–High	Flexible but judge-dependent
Fine-tuned tracer	Annotated trajectories	Step-level labels	High upfront	Requires training data and domain coverage
Dynamic runtime-based	Live trace/state	Online attribution/intervention	High	Supports operational monitoring and intervention

Table V: Comparison of Trajectory-Attribution Paradigms

VII. EVALUATION FRAMEWORKS

A. AgentEval

Guo et al.'s AGENTEval [7] formalizes agent execution as an evaluation DAG, where each node contains typed quality metrics and edges encode dependencies between upstream

and downstream steps. The framework achieves 0.89 failure-detection recall (vs 0.41 for end-to-end evaluation) and 72% root-cause accuracy (compared to approximately 81% human ceiling). The removal of the DAG structure resulted in reduced failure-detection recall by 22 percentage-points and reduced root-cause accuracy by 34 percentage-points.

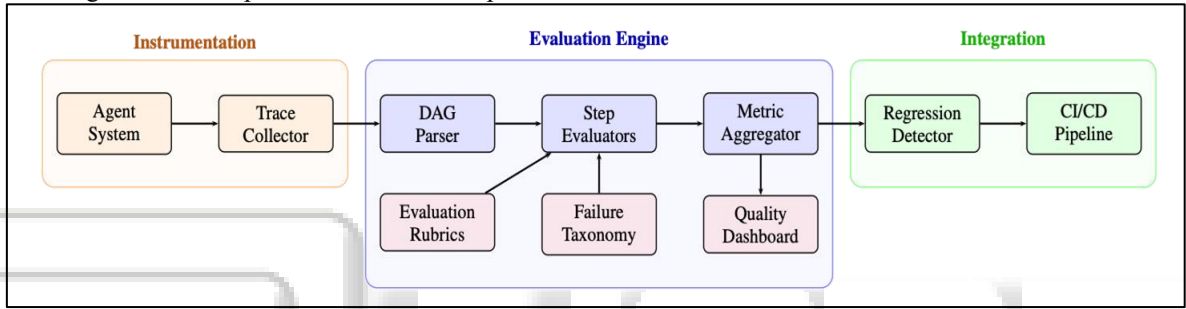


Fig. 5: AGENTEval architecture for DAG-structured step-level evaluation of agentic workflows. Adapted from Guo et al. [7].

B. Agentic CLEAR

Yehudai et al.'s Agentic CLEAR [8] provides dynamic evaluation at system, trace, and node levels. It leverages multi-level per-trace LLM judging with aggregation to system-wide and node-specific patterns and provides an interactive analysis interface. Its focus is on dynamic textual insights from execution traces rather than a fixed taxonomy.

C. Trajectory-Level Metric Set

Zamindar's level of analysis [3] for a trajectory consists of intent fidelity, retrieval fidelity, tool-argument correctness, end state success, propagation rate, amplification by

dimension, detection delay, recovery, and human intervention, and this perspective complements evaluation frameworks that produce, as an output, a final score.

D. Proposed Three-Layer Evaluation View

Chang et al.'s survey [9] organizes reasoning systems by their topology, including direct, linear chain and branch structured reasoning, and discusses the importance of balancing reasoning structure with computation cost and latency. Inspired by their discussion, this paper proposes a three-layer evaluation view of output-level correctness, reasoning-level evidence, and topology-level dependency structure, which is the authors' synthesis rather than a taxonomy by Chang et al.

Framework	Granularity	Taxonomy	Propagation modelling	Reported metrics
AgentEval [7]	Step / DAG	Hierarchical, 3 levels / 21 subcategories	Explicit DAG dependencies	Failure-detection recall, FPR, κ , RCA
Agentic CLEAR [8]	System / node / trace	Dynamic, taxonomy-free	Trace aggregation and recurring patterns	AUC, rubric/step scores, task-success prediction
Zamindar metrics [3]	Trajectory	Propagation-oriented	Explicit propagation and amplification	Fidelity, propagation, amplification, delay, recovery
Chang et al. [9]	Output / reasoning / topology	Topology-oriented	Reasoning topology	Reliability/robustness, cost and latency considerations

Table VI: Comparison of Agentic Evaluation Frameworks

VIII. MITIGATION AND GOVERNANCE

A. Mitigation Families

Kazlaris et al. [2] group hallucination mitigation approaches into six categories: training and learning methodologies,

architectural modifications, input or prompt optimization, post-generation quality control, interpretability and diagnostic techniques, and agent-based orchestration. Relevant approaches for mitigating hallucination at the step-level problem are those that can modify or interrogate an intermediate state before it becomes a downstream

dependency. The technique of confidence-gated memory writing by Shaikh is a prime example: in the experiments reported, the false-memory rate was reduced by 52.8% with approximately 7.9% latency overhead [5].

B. Governance Constraints

Valdez-Cantú et al. [10] derive seven transition gaps, G-G, and fourteen structural constraints, SC-SC, across state

C. Mitigation Mapping

Table VII maps mitigation directions to the stage at which they intervene.

Technique	Stage	Relevant Mechanism	Step-Level Relevance
Training/learning approaches	Prevent	Modify model behaviour before deployment	May reduce recurrent generation errors
Architectural modifications	Prevent / contain	Alter system structure or state handling	Can isolate intermediate dependencies
Prompt optimization	Prevent	Constrain instructions and intermediate behaviour	Can define step-specific requirements
Post-generation quality control	Detect / correct	Inspect generated content before acceptance	Can verify intermediate outputs
Interpretability/diagnostics	Detect / attribute	Expose signals associated with failures	Supports localization of problematic steps
Agent orchestration	Prevent / contain / recover	Coordinate planning, tools, memory, and verification	Provides explicit intervention points

Table VII: Mitigation Mapping

IX. PROPOSED MITIGATION DIRECTIONS

This section is the authors' constructive synthesis of gaps identified in the ten-source corpus. The following directions are proposals, not validated results. None is presented as experimentally established. Each direction is stated together with the gap it addresses and the evidence that would be needed for evaluation.

A. Criteria for Proposed Directions

A direction is included when the surveyed sources identify a consequential mechanism that is not fully operationalized, when the direction can be implemented using mechanisms already present in the corpus, and when its effect can be measured using controlled fault injection, trajectory analysis, memory gating, verification, or DAG-based evaluation. The proposals therefore extend existing mechanisms rather than introducing an unrelated detection principle.

B. State and Memory

- 1) Two-sided memory gating. Shaikh evaluates confidence-gated memory writing and observes decreases in false-memory rate and memory reuse [5]. We propose to extend it by a read-time check ensuring a retrieved entry is provenance-, age-, and consistency-checked before it conditions downstream reasoning; entries that fail the check would be quarantined. The new read-time component is proposed (not reported as implemented).
- 2) Dependency-aware descendant invalidation. Zamindar and Shaikh discuss persistence of contaminated state and the importance of invalidating dependent state not just re-writing a surface output ([3]; [5]). The proposed extension would be to use explicit dependency edges to invalidate downstream memory entries, plan premises, and pending tool calls after a corrected step. Guo et al.'s DAG representation provides the structural mechanism,

isolation, control alignment, and execution governance [10]. The source presents these constraints as analytically derived conditions rather than empirically validated design laws. In this survey, step-level detection is therefore treated as both a diagnostic mechanism and a potential control plane at which state updates, intermediate reasoning, tool use, escalation, and execution can be inspected.

as discussed by Wang et al. with dependency-based attribution ([4]; [7]). The value of this approach in reducing recovery time without excessive false invalidation warrants controlled evaluation.

- 3) Schema-versioned state recalibration. Valdez-Cantú et al. identify schema versioning and recalibration as structural constraints SC3 and SC4 [10]. Their proposed direction is to attach them to structured agent state and to perform recalibration when there is change of schema. No validation result for this proposed mechanism is claimed here.

C. Control and Verification

- 4) Position-aware adaptive verification. Bansal et al. report position-dependent cascading vulnerability while Shaikh reports increasing amplification with later injection positions ([5]; [6]). The proposed direction is a verification scheduler which allocates verification depth as a function of step position and observed recovery characteristics. Chang et al. discuss cost and latency as design considerations for reasoning structures; this paper uses that framing for motivation of an explicit verification budget [9]. The scheduler itself is proposed here and is not reported as an implemented system.
- 5) Contradiction-triggered counter-retrieval. Zamindar identifies retrieval reinforcement as a propagation pattern, whereas Kazlaris et al. discuss the dependence of retrieval based mitigation on the quality of retrieval ([2]; [3]). The proposed mechanism would emit a deliberately disconfirming retrieval query when a retrieved claim is found to be at odds with a current observation. Its efficacy would need to be evaluated by measuring performance on containment and contradiction detection tasks.
- 6) Step-level calibrated abstention. Ghosh identifies calibration as an understudied component of

hallucination detection mechanisms, while Kazlaris et al. discuss selective abstention as a mitigation direction ([1]; [2]). The proposed extension is a step-level policy, which interrupts or promotes continuing the conversation when the calibrated confidence level drops below a threshold value. The calibration threshold and its influence on unsafe continuation should be investigated experimentally.

- 7) Independent evidence-channel verification. Zamindar and Valdez-Cantú argue that independent verification and pre-execution safeguards are vital ([3]; [10]). Guo et al. [7] use cross-family judging, while Yehudai et al. [8] inspect execution traces at multiple levels. The proposed extension is to verify the intermediate outputs of a model against an independent evidence source (as opposed to a judge model) such as an associated tool, schema, or external state.

D. Attribution, Repair, and Evaluation

- 8) Closed-loop attribution-to-repair. Wang et al. point out the gap between attribution outputs and subsequent enhancement [4]. Guo et al. employ counterfactual correction to evaluate whether the detected root causes are consistent with downstream improvement [7]. The direction of the proposal is to feed the attributed root-cause step into a repair operator, re-execute the affected descendants, and reassess the trajectory. The proposal has not been validated.
- 9) Multi-fault interaction testbeds. Shaikh exploits single-fault injection techniques for interpretability, whereas Zamindar discovers convergent propagation to be a propagation pattern in ([3]; [5]). We propose that the new testbed should inject multiple fault types at different steps and measure if their effects were additive or interacting. This is simply a proposed experiment design, and not a result of any empirics.
- 10) Intent-conditioned rubric generation. Yehudai et al. describe failure mode where rubrics generated from the task description reward the completion of adversarial

tasks for which refusal is the correct response [8]. The direction they suggest is to condition rubric generation on the task intent (whether the task is straightforward, ambiguous, adversarial). Its success should be evaluated against the success prediction protocol, rather than assuming it would be effective.

E. Validation Requirements

A proposed direction would be taken as supported only after controlled evaluation. A minimal protocol would compare baseline execution, vs. fault-injected execution, a safeguard-enabled execution under the same budget, and a recovery condition. Presented measures should include propagation, amplification, detection delay, containment, recovery, and the cost or latency of the safeguard. This section does not assert that any proposal has yet satisfied those criteria.

F. Boundary Conditions

The proposals are circumscribed by the evidence in the ten-source corpus. The quantitative results are model- and task-specific, the corpus does not provide cross-model generalization for the proposed mechanisms, and DAG-based directions are contingent on the degree to which the workflow structure can be recovered. Guo et al. find that the DAG advantage becomes marginal beyond approximately 60% of non-DAG traces [7]. The proposals also inherit the definitional variability between hallucination and error in the surveyed literature.

X. COMPARATIVE ANALYSIS

The ten-source corpus has agreed upon the importance of intermediate states, trajectory structure, and dependencies ([3]; [4]; [5]; [7]), but the sources differ considerably in layer of analysis and form of evidence. Detection surveys emphasize signal and granularity ([1]; [2]); propagation studies measure downstream effects ([3]; [5]; [6]); attribution work emphasizes localization [4]; evaluation frameworks operationalized trajectory-level assessment ([7]; [8]); and governance work organizes structural controls [10].

Work	Layer	Granularity	Propagation modelled?	Evaluation method	Key empirical/analytical claim	Stated limitation
Ghosh [1]	Detection	Output / trajectory	No	Multidimensional survey	Organizes detection by signal, access, knowledge, granularity, cost, and failure mode	Survey corpus and definitions vary
Kazlaris et al. [2]	Mitigation	Output	No	Taxonomic survey	Six mitigation families organize intervention strategies	Mitigation evidence is heterogeneous
Zamindar [3]	Propagation	Trajectory	Yes	Scoping review	Defines a sixteen-stage cascade and ten propagation patterns	Heterogeneity limits prevalence claims
Wang et al. [4]	Attribution	Trajectory	Yes	Survey of trajectory analysis	Identifies four attribution paradigms and an attribution-enhancement gap	Step-level attribution remains limited

Shaikh [5]	Propagation	Step / trajectory	Yes	Controlled fault injection	Reports position-dependent amplification and cross-episode memory contamination	Model- and experiment-specific results
Bansal et al. [6]	Propagation	Step	Yes	Controlled error injection	Reports AF across three model families and position-dependent vulnerability	Tied to the tested models, task, and injection design
Guo et al. [7]	Evaluation	Step / DAG	Yes	DAG evaluation and ablation	Reports improved failure detection and root-cause accuracy from dependency modelling	DAG advantage diminishes for highly non-DAG traces
Yehudai et al. [8]	Evaluation	System / node / trace	Partial	Multi-level LLM judging	Produces dynamic evaluation at system, trace, and node levels	Judge-based evaluation remains model-dependent
Chang et al. [9]	Evaluation / topology	Output / reasoning / topology	Structural	Survey	Organizes reasoning by topology and discusses evidence, cost, and latency trade-offs	Focus is time-series reasoning
Valdez-Cantú et al. [10]	Governance	System / trajectory	Structural	Diagnostic review	Derives G1–G7 and SC1–SC14 as structural conditions	Constraints are analytically derived, not empirically validated

Table VIII: Comparative Analysis of the Ten Primary Sources

Across the corpus, the main insight is that final-output analysis alone does not reveal all aspects of multi-step failures ([3]; [4]; [7]). Shaikh and Bansal et al. provide controlled quantitative evidence for such propagation and amplification, while Zamindar provides a higher-level scoping taxonomy for mechanisms of propagation ([3]; [5]; [6]).

The corpus moreover agrees on the importance of dependency-aware evaluation. AgentEval makes dependencies explicit via DAG structure, while Wang et al. stress the importance of trajectory attribution and the gap between identifying a cause and using that information for improvement ([4]; [7]). Agentic CLEAR supplements these methods with multi-level system, trace, and node evaluation [8].

The primary difference lies in the evidentiary scope. Surveys and structural reviews assimilate diverse literature ([1]; [2]; [3]; [4]; [9]; [10]) while controlled studies provide model- and benchmark-specific measurements ([5]; [6]; [7]). These types of evidence should not be conflated as equivalent prevalence estimates.

Another difference is in the meaning of evaluation. Some works emphasize detection or attribution ([1]; [4]; [7]) while governance-oriented work emphasizes controls over state, orchestration, and execution [10]. The three layer evaluation view proposed in this paper aims to connect both without assuming which source has the complete structure.

XI. OPEN CHALLENGES

A. Definitional Instability

The corpus uses notions of hallucination, error, failure, incorrect reasoning, attribution, propagation, and contamination. An operational vocabulary is required to enable direct comparisons across papers' metrics.

B. No Shared Step-Level Detection Benchmark

The surveyed work shows that detailed trajectory annotation is possible, but the ten-source corpus falls short of an agreed-upon benchmark encompassing reasoning, retrieval, memory, tool arguments, execution, verification, and recovery in a single evaluation framework.

C. Model-Specific Quantitative Evidence

Quantitative results are specific to certain models, tasks, and protocols. Bansal et al. evaluate GPT-4o-mini, Llama-3.1-8B, and Qwen-2.5-72B, and Shaikh evaluates Claude Sonnet 4.6 ([5]; [6]). Further replication on other families of models and tasks is necessary before generalization can occur.

D. Attribution–Enhancement Gap

Identifying a responsible step does not necessarily alter system behaviour. The attribution output must be associated with intervention policies for verification, re-planning, memory rollback, execution blocking, correction, or human intervention [4].

E. Cost-Latency Budget

Guo et al. report approximately two seconds of evaluation time per step in their deployment setting, while Shaikh reports approximately 7.9% average latency overhead for confidence-gated memory writing ([5]; [7]). As such, step-level evaluation requires explicit cost and latency modeling.

F. Taxonomy Coverage Across Domains

Guo et al. report decreases of approximately 14 percentage points in root-cause accuracy on tau-bench and 20 percentage points on SWE-bench in their cross-system evaluation [7]. This suggests that a failure taxonomy may need to be extended to apply to novel workflow domains.

G. Non-DAG Architectures

Guo et al. report that the benefit of DAG-based evaluation collapses beyond about 60% non-DAG traces [7]. Consequently, one can apply dependency-based approaches only to structures where dependencies can be recovered with sufficient confidence.

H. Governance Integration

Valdez-Cantú et al.'s G-G transition gaps and SC-SC structural constraints provide a vocabulary for articulating the interplay between technical trajectory analysis and state, control, and execution governance [10]. The remaining challenge is to quantify the interaction of such structural controls with step-level detection mechanisms in practice.

XII. CONCLUSION

This survey explored hallucination detection and error propagation in agentic LLM systems through a selection of ten primary sources. The integration of the literature emphasized detection layers, granularity, and methods, while contrasting output-level detection, step-level detection, propagation, attribution, evaluation, mitigation, and governance.

The corpus suggests that multi-step execution shifts reliability properties by allowing intermediate errors to become part of the state, affect subsequent decisions, and be retained in memory. Controlled studies establish model-specific evidence for the mechanisms through which amplification, memory contamination may occur, while trajectory surveys, evaluation frameworks provide tools for analyzing and attributing reliability with respect to dependencies.([3]; [4]; [5]; [6]; [7]; [8])

The paper's three-layer evaluation view and ten mitigation directions are explicitly proposed synthesized rather than validated methods. Their aim is to relate detection, propagation analysis, attribution, intervention, and governance into a shared research framework. Further empirical studies are needed to evaluate these directions on models, tasks, and execution architectures.

REFERENCES

- [1] A. Ghosh, "Large Language Model Hallucination Detection: A Multidimensional, Evidence-Traceable Framework and Research Agenda," SSRN, 2026. [Online]. Available: <https://ssrn.com/abstract=7299799>. doi: 10.2139/ssrn.7299799.
- [2] I. Kazlaris, E. Antoniou, K. Diamantaras, and C. Bratsas, "From Illusion to Insight: A Taxonomic Survey of Hallucination Mitigation Techniques in LLMs," *AI*, vol. 6, no. 10, p. 260, 2025. doi: 10.3390/ai6100260.
- [3] N. Zamindar, "When Language-Model Errors Become Action Errors: A Systematic Scoping Review of Error Propagation in Agentic AI," 2026.
- [4] J. Wang, Y. Wang, M. Chen, X. Xie, C. Chen, F. Mu, Z. Liu, and Q. Wang, "A Survey for LLM Agent Trajectory Analysis: From Failure Attribution to Enhancement," *IEEE Transactions on Software Engineering*, 2026. doi: 10.1109/TSE.2026.3717765.
- [5] H. Shaikh, "Error Propagation and Memory Contamination in Tool-Using LLM Agents," Thesis, University of Helsinki, 2026.
- [6] A. Bansal, R. Agarwal, and I. Gangwani, "Cascading Failure Amplification in Multi-Step LLM Reasoning: Operational Definitions, Trace-Level Diagnostics, and Intervention Analysis," in *Workshop on Failure Modes of Agentic AI, International Conference on Machine Learning (ICML)*, 2026.
- [7] D. Guo, J. Wu, and S. M. Yiu, "AgentEval: DAG-Structured Step-Level Evaluation for Agentic Workflows with Error Propagation Tracking," 2026. doi: 10.48550/arXiv.2604.23581.
- [8] A. Yehudai, L. Eden, and M. Shmueli-Scheuer, "Agentic CLEAR: Automating Multi-Level Evaluation of LLM Agents," in *Proc. 64th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pp. 750–763, 2026. doi: 10.18653/v1/2026.acl-demo.74.
- [9] C. Chang, Y. Shi, D. Cao, W. Yang, J. Hwang, H. Wang, J. Pang, W. Wang, Y. Liu, W.-C. Peng, and T.-F. Chen, "A Survey of Reasoning and Agentic Systems in Time Series with Large Language Models," *Transactions on Machine Learning Research*, 2026. [Online]. Available: <https://arxiv.org/abs/2509.11575>. doi: 10.48550/arXiv.2509.11575.
- [10] C. Valdez-Cantú, J. A. Cantoral-Ceballos, and J. Alvarado-Uribe, "The Governance Gap in Contemporary LLM-Based Agentic Systems: A Structural Diagnostic Review," *AI*, vol. 7, no. 8, p. 322, 2026. doi: 10.3390/ai7080322.