

Predefined-Tool Execution for Reliable LLM Agents in Semiconductor Lot Analysis*

Nirupama Vijayaraghavan Malolan¹ and Duc V. Le¹

¹University of Twente, Enschede, The Netherlands.

Correspondence: nirupamamalolan@gmail.com, v.d.le@utwente.nl

Abstract

LLM agents for structured data analysis rely on dynamic query generation, in which the model synthesises tool calls at runtime. This introduces failures such as fabricated dataset references, unsupported operations, and incorrect results, which existing mitigations reduce but do not eliminate. We propose predefined-tool execution as an architectural alternative in which the language model selects and parameterises tools from a fixed, schema-validated registry. We formalise the approach using typed tool specifications and prove executability and hallucination-suppression properties under assumptions. We complement the architecture with a Numerical Faithfulness Checker that verifies quantitative claims against pre-computed ground truth, and evaluate the system on a domain-specific benchmark using four language-model backbones. The suppression property follows from the validation layer under those assumptions, and held across all 276 evaluated agent executions involving 23 queries, four backbones and three runs. Mean exact-plan-match rates ranged from 50.7 to 53.6 per cent, with a between-backbone spread within a single backbone. Because exact plan match penalises valid but non-minimal plans, it is a strict lower bound on task correctness. Given only three runs per backbone, comparisons are descriptive, not evidence of equivalence. The on-premise open-weight backbones performed comparably to commercial ones on the measured metrics, supporting confidential deployments.

1 Introduction

LLM agents have demonstrated strong capabilities in reasoning, planning, and tool use, making them compelling interfaces for natural-language-driven

analysis over complex structured databases (Wei et al., 2022; Masterman et al., 2024; Wang et al., 2024). The dominant paradigm is dynamic query generation, in which the model synthesises tool calls at runtime (Yao et al., 2023), enabling flexible responses without preprocessed query setup. This flexibility, however, introduces a systematic reliability problem: models fabricate dataset identifiers, reference unsupported operation types, or produce silently incorrect results that bypass validation entirely. These are instances of what Lin et al. (2025) term Execution Hallucination, where an agent reports completion of an execution step that did not in fact occur.

Benchmarking studies document this problem across state-of-the-art models (Zhang et al., 2024), and industrial deployments find that dedicated runtime repair mechanisms remain necessary in production (Dobrovsky et al., 2025). Detection after the fact remains brittle and model-dependent. This problem is acute in industrial analytical settings where data is heterogeneous and operationally sensitive.

In semiconductor manufacturing, process engineers must analyse heterogeneous experimental split-lot data spanning parametric electrical measurements at wafer test structures, comprehensive functional test results across all dies, equipment parameter measurements throughout fabrication, and particle defect counts at inspection steps. They must do so without programming knowledge, in a setting where any hallucinated conclusion carries direct operational consequences. Across the literature reviewed for this work on LLM-agent systems in semiconductor and manufacturing analytics, we did not identify a published lot-analysis agent combining a typed predefined-tool registry, pre-execution validation, and claim-level numerical verification, nor a domain-specific benchmark for this task. This paper proposes **predefined-tool execution** as a principled architectural solution:

*This paper is based on the first author’s MSc thesis at the University of Twente, carried out during a research internship at Infineon Technologies, Nijmegen, The Netherlands. No earlier version of this manuscript has been posted as a preprint or presented elsewhere.

the language model selects and parameterises tools from a fixed, schema-validated registry rather than synthesising queries freely, so that invocations of unregistered datasets or action types are blocked before dispatch, within the scope of the formal specification, rather than detected after generation. The SplitLot Analysis Agent demonstrates this for semiconductor lot management.

Three contributions are made. First, a **formal specification** of predefined-tool execution as typed tuples with preconditions, postconditions, and coverage fields, with proofs that validated plans execute without schema-level runtime errors and that the Execution Hallucination class is suppressed by architectural restriction within the scope of the specification; unlike schema-only approaches, the coverage field makes the hallucinated-versus-unsupported distinction explicit and actionable. Second, for queries producing structured breakdown output, a **Numerical Faithfulness Checker** that automatically verifies every quantitative claim in the structured explanation against pre-computed ground truth, converting output grounding from a prompting instruction into a measurable per-claim property. Third, a **five-tier domain-specific benchmark** for semiconductor lot analysis agents with deterministic ground truth, evaluated across four language model backbones (two commercial and cloud-hosted, two open-weight and deployed on-premise) to test whether the measured reliability properties vary with backbone choice; it covers analytical operations absent from general-purpose agent benchmarks.

2 Related Work

2.1 Tool-Augmented Agents and Hallucination

The ReAct paradigm (Yao et al., 2023) is the dominant approach for analytical agents, interleaving reasoning and action through dynamic tool call synthesis. Program-Aided Language Models (Gao et al., 2023) demonstrated that delegating computation to a deterministic interpreter substantially improves quantitative reasoning, a direct precedent for separating language model planning from deterministic execution. Multi-agent extensions via AutoGen (Wu et al., 2024) and MetaGPT (Hong et al., 2024) introduce coordination layers in which a hallucinated message from a planner can corrupt an executor without detection. The CoALA taxonomy (Sumers et al., 2024) provides a rigorous frame-

work for language agent memory components and informs the memory architecture of this work. Despite these advances, unconstrained query synthesis remains architecturally vulnerable to hallucination. Lin et al. (2025) establish a five-category agent hallucination taxonomy spanning reasoning, execution, perception, memorisation, and communication, and identify tool documentation limitations among the triggering causes of Execution Hallucinations. Liu et al. (2026) further show hallucinations compound through multi-step trajectories, making early architectural interception preferable to post-hoc correction. Existing mitigations, including Chain-of-Verification (Dhuliawala et al., 2024), Self-RAG (Asai et al., 2024), Self-Refine (Madaan et al., 2023), Reflexion (Shinn et al., 2023), and structured output enforcement (White et al., 2023; Sahoo et al., 2024), reduce hallucination rates but operate by detection or correction after generation. The predefined-tool approach taken in this paper instead restricts the action space, so that the failure class is contained before dispatch rather than repaired afterwards.

2.2 Benchmarking and Claim Verification

AgentBench (Liu et al., 2024), GAIA (Mialon et al., 2024), ToolQA (Zhuang et al., 2023), and τ -bench (Yao et al., 2025) provide general-purpose agent evaluation frameworks but cover none of the analytical operations required for semiconductor lot management. ToolBeHonest (Zhang et al., 2024) introduces Solvability Detection Accuracy for measuring graceful refusal behaviour, and AgentEval (Arabzadeh et al., 2024) assesses outputs against expert-defined criteria, both directly informing the benchmark design in this work. Verifying explanation faithfulness against an evidence source resembles a Natural Language Inference problem. SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2018), and FEVER (Thorne et al., 2018) are the closest foundational resources: their fact-verification paradigm is structurally analogous to verifying agent explanations against an execution log. DeBERTa (He et al., 2021) provides the state-of-the-art cross-encoder for entailment classification and serves as the backbone of the natural language verifier in this work. Cross-domain applications indicate feasibility in regulated settings: Zeba et al. (2026) demonstrate high-precision verification on clinical text, and Lu et al. (2026) evaluate financial tool-use agents against regulatory-alignment criteria.

2.3 Language Model Agents in Industrial Domains

Domain-adapted models SemiKong (Nguyen et al., 2024) and ChipNeMo (Liu et al., 2023) demonstrate that semiconductor-specific pre-training improves task performance, while Werheid et al. (2025) provide empirical support for predefined analytical workflows in manufacturing equipment selection. However, domain adaptation does not resolve execution reliability. Dobrovsky et al. (2025) therefore deployed a ReAct-based agent for semiconductor failure analysis in which generated database queries required a dedicated retry-and-repair loop when malformed. Their work indicates both that language model agents are viable for semiconductor data analysis and that unconstrained query generation in this domain requires runtime error handling, which this paper instead addresses architecturally.

3 System Design and Formal Foundations

The SplitLot Analysis Agent gives process engineers natural-language access to heterogeneous manufacturing data. It reduces Execution Hallucinations by restricting the language model to a fixed, schema-validated tool registry rather than permitting free query synthesis.

3.1 System Architecture

The agent comprises three tiers, shown in Figure 1. The **data layer** maintains lot measurement data across four measurement domains and an experiment metadata domain recording wafer-to-treatment-group assignments and provenance (Appendix J). Data is cached locally on first access and served from cache thereafter.

The **analysis layer** hosts the typed tool registry, language model planner, and pre-execution validation component. Tools are exposed through the Model Context Protocol (Anthropic, 2024), an open, provider-agnostic standard for typed tool definitions. When a query arrives, the language model produces a structured analytical plan. Before any data access, the validation component checks every planned operation against the registered action set $\mathcal{A}$, dataset set $\mathcal{D}$, and visualisation set. Any operation referencing an unregistered resource causes the plan to be rejected, and an explicit refusal is returned. The primary computed quantity is the group-separation score (DIFF), a standardised ef-

fect size (Equation 1):

$$\text{DIFF}(p) = \frac{|\text{median}(G_{\max}) - \text{median}(G_{\min})|}{\sqrt{(\text{sd}(G_{\min})^2 + \text{sd}(G_{\max})^2)/2}} \quad (1)$$

where $G_{\max}$ and $G_{\min}$ are the groups with maximum and minimum median values. The statistic is analogous to Cohen’s d , enabling comparison across parameters with different scales. DIFF scores are pre-computed and read directly by the language model, which is constrained never to recompute them (Gao et al., 2023).

The **interaction layer** delivers responses integrating statistical results, visualisations, and grounded explanations. Explanations draw on two asymmetric evidence streams: a quantitative stream sourced exclusively from the session execution log, and a documentary stream from a retrieval-augmented parameter glossary (Lewis et al., 2020). Numerical claims can only originate from the execution log, preventing the failure mode where retrieved passages are falsely cited as numerical sources (Huang et al., 2025).

The agent’s four-component memory architecture following Sumers et al. (2024) is described in Appendix L.

3.2 Formal Specification

Before deployment, each tool is specified by six fields covering its identity, inputs, preconditions, computation, outputs, and scope of coverage. Formally, tools are typed tuples $\mathcal{T}_i = (\text{name}_i, \text{inputs}_i, \text{pre}_i, \text{comp}_i, \text{post}_i, \text{cov}_i)$ where $\text{name}_i \in \mathcal{A}$ is the registered action identifier, inputs_i is the typed parameter schema, pre_i specifies data availability preconditions, comp_i is the deterministic computation performed entirely by the tool, post_i specifies evidence entries appended to the execution log, and cov_i states the class of requests the tool satisfies.

Scope and assumptions. Propositions 1 and 2 are stated over the plan representation Π emitted by the planner and consumed by the validation layer. They assume: (A1) the registered sets $\mathcal{A}$ and $\mathcal{D}$ are finite and fixed at deployment time; (A2) the validation layer is the only path from planner output to the execution layer; (A3) each tool implementation is correct with respect to its own comp_i specification; (A4) the cached data referenced by pre_i is present and schema-conformant. Under these assumptions the propositions concern one question

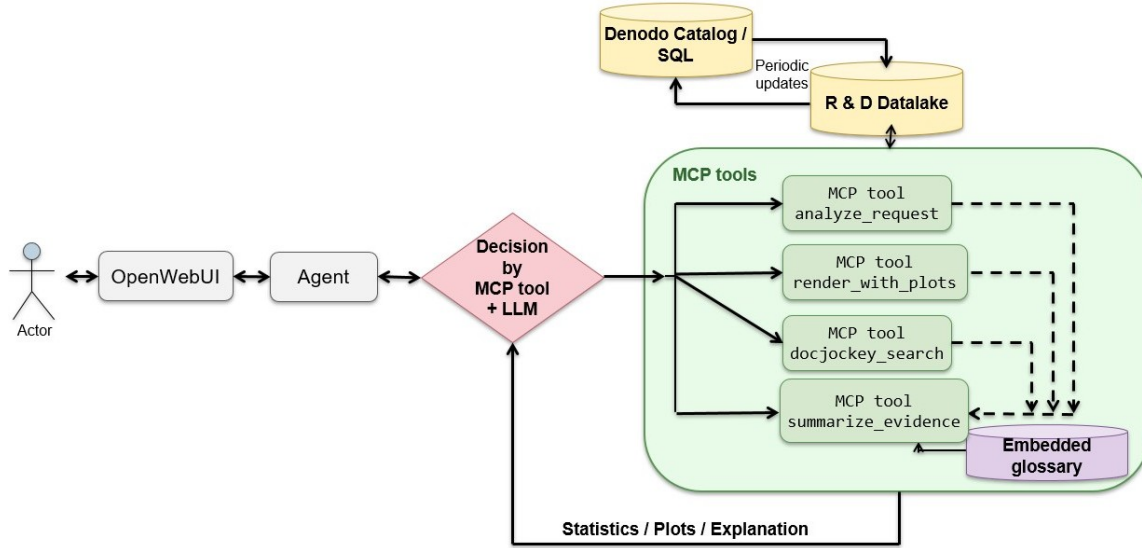


Figure 1: SplitLot Analysis Agent architecture. Queries are validated against the tool registry before execution. Invalid plans are rejected before any data access and returned as explicit refusals. Explanations are grounded in the session execution log and enriched by a retrieval-augmented parameter glossary.

only: whether an unregistered action or dataset reference can reach execution. They make no claim about whether the selected tool is the appropriate one for the request (measured separately by TSA), whether the returned values are correct (measured by NFR), or whether a registered tool can fail for reasons unrelated to unregistered resource references.

Proposition 1 (Executability Guarantee). If the validation layer completes without error for plan Π , then every action belongs to $\mathcal{A}$ and every referenced dataset belongs to $\mathcal{D}$, and the plan executes without schema-level runtime errors.

The validation layer enforces this through sequential action-type filtering, dataset validation, and visualisation-type filtering, each blocking unregistered references before dispatch. Full proof in Appendix A.

Proposition 2 (Execution Hallucination Suppression). Under the predefined-tool architecture, the Execution Hallucination Rate is zero for all plans satisfying the Executability Guarantee, within the scope of the formal tool and dataset registry.

Lin et al. (2025) define Execution Hallucinations as cases where an agent claims to have completed an execution sub-stage that was not in fact performed, and divide them into Tool Selection Hallucinations, in which non-existent or irrelevant tools are chosen, and Tool Calling Hallucinations, involving incorrect, omitted, or fabricated parameters. The registry addresses the first subclass in full

and the dataset-valued portion of the second. Under assumptions A1–A4 the validation layer blocks these subclasses before dispatch, so the failure class is contained architecturally rather than detected after generation; no post-generation correction step is required for this class. The scope of the property is exactly the registered sets: a request outside $\mathcal{A}$ or $\mathcal{D}$ is refused and not served. Full proof in Appendix A. The cov_i field further enables a **Coverage Completeness** criterion: a registry is coverage-complete if every solvable request is covered by at least one tool’s coverage specification, making the hallucinated-versus-unsupported distinction explicit and actionable. The formal definition is in Appendix K.

3.3 Hallucination Verification

The formal properties above suppress Execution Hallucinations within the scope of the registry but do not address whether the natural language explanation faithfully represents computed results. An unfaithful summary following correct execution constitutes a Perception Hallucination (Lin et al., 2025). The full mapping of all five hallucination categories to the system is in Appendix I.

The structured explanation output contains three fields: a natural language **summary**, a **glossary** of retrieved parameter descriptions, and machine-readable **breakdowns** recording measurement step, parameter name, domain, DIFF score, and per-group statistics. Confining numerical

claims to the breakdown field makes every value directly verifiable without natural language processing.

The **Numerical Faithfulness Rate** (NFR) verifies each entry against pre-computed ground truth:

$$\text{NFR} = \frac{|\{b_j \in B : \frac{|\hat{b}_j.\text{diff} - b_j.\text{diff}|}{|\hat{b}_j.\text{diff}|} \leq \epsilon\}|}{|B|} \quad (2)$$

where B is the set of all breakdown entries, $\hat{b}_j$ is the ground truth from the pre-computed cache, and $\epsilon = 10^{-4}$. Implementation in Appendix E.

For prose claims in the summary field, a supplementary Natural Language Inference (NLI) pipeline was implemented. Full pipeline details and interpretation guidance are in Appendix D.

4 Experiments and Results

4.1 Experimental Setup

The benchmark spans five tiers of increasing analytical complexity: single-tool retrieval (T1), multi-domain analysis (T2), cross-lot scanning over $N = 34$ lots (T3), metadata-driven lot discovery (T4), and a seven-probe adversarial tier requiring graceful refusal (T5). Ground truth is deterministic for each tier, derived from pre-computed summary files, metadata filter results, or the expected absence of fabricated content (Appendix N); the construction protocol is in Appendix O and the full tier definitions in Appendix C.

Six metrics measure hallucination behaviour and planning quality independently, separating architectural safety guarantees from model-dependent planning behaviour. *Validation Pass Rate* is the fraction of queries for which the validation layer ran to completion and returned a verdict, whether acceptance or rejection; it certifies that validation was reached, not that every plan was accepted, and relates to Proposition 1. *Execution Hallucination Rate* is the fraction of dispatched executions invoking an unregistered resource, measured post-validation. *Domain Scope Accuracy* is the fraction of queries where the domains recorded in the execution log match those declared in the plan. These three measure containment. *Tool Selection Accuracy* is the fraction of queries whose executed tool-type sequence exactly matches the annotated minimal sequence; it measures plan optimality rather than task correctness, and relates to Reasoning Hallucinations (Appendix I). *Numerical Faithfulness Rate* is the fraction of emitted breakdown entries

within $\epsilon = 10^{-4}$ of ground truth (Eq. 2), and is undefined where no breakdown is emitted. *Solvability Detection Accuracy* is the fraction of adversarial probes whose rejection is returned as an intelligible refusal (Zhang et al., 2024), and measures refusal presentation rather than containment.

The agent is evaluated against four language model backbones: GPT-5, Claude Sonnet 4.5, DeepSeek-R1-70B, and Llama-3.3-70B. All share an identical tool server, validation layer, planning prompt, and dataset registry. Only the model identifier varies, so systematic differences between backbones are not confounded by the surrounding system; sampling stochasticity still produces run-to-run variation within a single backbone (§4.2.1). Backbone rationale and configuration are in Appendix B. Metrics and tier definitions were pre-registered before evaluation, following standard benchmark practice. Six versions of the agent were developed iteratively; the final version (v6) is the system assessed here. The development history from v1 through v6 is in Appendix F.

Three denominators recur: 23 queries, $23 \times 4 = 92$ backbone–query combinations, and $23 \times 4 \times 3 = 276$ executions; a figure written $x/23$ is a single run. Containment, task correctness, exact plan match and refusal presentation are measured separately and are not interchangeable. TSA scores a query correct only if every tool type in the executed plan matches the annotated minimal sequence, so valid but longer routes are scored incorrect. Full definitions and the annotation protocol are in Appendix O.

4.2 Results

4.2.1 Validation Containment and Execution Outcomes

Across all evaluated runs, the validation layer prevented plans containing unregistered actions or dataset references from reaching the execution layer. Table 1 reports the corresponding validation, execution, planning, and refusal metrics for the four model backbones.

Containment. EHR was 0/23, VPR 23/23 and DSA 23/23 in every one of the 12 backbone-runs, covering all 276 executions. Under assumptions A1–A4 this follows from the shared validation layer rather than from backbone behaviour, so the evaluation’s role is to confirm that no path circumvented that layer, not to measure a performance difference. EHR is measured post-validation over

Backbone	VPR	TSA	DSA	NFR	EHR	SDA
GPT-5	23/23	11/23	23/23	1/1 [†]	0/23	4/7
Claude Sonnet 4.5	23/23	13/23	23/23	2/2 [†]	0/23	4/7
DeepSeek-R1-70B	23/23	12/23	23/23	2/2 [†]	0/23	4/7
Llama-3.3-70B	23/23	12/23	23/23	1/1 [†]	0/23	4/7

[†]NFR is defined only over emitted structured breakdown entries. Q13 is the *only* query in the suite that emitted one, and it did so in 6 of its 12 backbone-runs: GPT-5 1/3, Claude 2/3, DeepSeek 2/3, Llama 1/3. The column reports entries emitted and passing per backbone across its three runs, totalling 6/6; it is not a rate over the 23-query suite (Appendix M). NLI Faithfulness Score (NLI-FS; secondary diagnostic, over the 22 queries yielding extractable claims) 24.8–31.0%; see Appendix D.

Table 1: Multi-backbone results, one representative run per backbone. VPR, TSA, DSA and EHR are out of the 23 queries in that run; SDA is out of the 7 Tier 5 probes; NFR counts breakdown entries emitted and passing per backbone over its three runs, as the footnote details. Claims qualified in the text as holding across all runs are over 276 executions. TSA is exact match against the annotated minimal tool-type sequence and is not a measure of task correctness.

dispatched calls, and every dispatched execution was additionally verified to respect the registered action and dataset boundaries. VPR counts queries for which validation returned a verdict rather than queries whose plans were accepted unchanged: the A2 and A5 probes were rejected at validation and still count toward VPR, because the metric records that validation was reached. The adversarial tier exercises the mechanism directly: the A2, A4 and A5 probes were each contained at validation in all 12 backbone-runs, and no dispatched execution in any run reached an unregistered resource. Propositions 1 and 2 are therefore best read as soundness results rather than as empirical findings. They state which plans the validation layer admits, in the way a type system states which programs its checker admits, and a soundness result of this form is neither confirmed nor falsified by a measured rate. The corresponding empirical question is narrower: whether assumption A2, that validation is the sole path from planner output to execution, in fact held in the deployed system. The 276 executions support that it did.

Planning. Table 1 reports one of three independent runs per backbone. Across all three runs, the per-run TSA figures (out of 23) are: GPT-5 11/13/12, Claude Sonnet 4.5 13/12/12, DeepSeek-R1-70B 12/12/12, Llama-3.3-70B 12/11/12, giving per-backbone means of 52.2%, 53.6%, 52.2% and 50.7%. The spread between backbone means (2.9 percentage points) is smaller than the largest run-to-run spread within a single backbone (8.7 percentage points, GPT-5 across its three runs). With three runs per backbone, this is a descriptive observation and does not establish statistical equivalence. Containment and plan quality are in-

dependent: Q13 fails exact plan match in all 12 backbone-runs while returning numbers the Numerical Faithfulness Checker verified in each of the six runs that emitted them.

Task correctness versus exact plan match. TSA scores a query correct only if the executed tool-type sequence matches the annotated minimal sequence exactly; a plan returning the requested result by a longer but valid route is scored incorrect. Whether the returned answer matched the ground-truth answer is reported separately in Table 7. Q5, Q6, Q8, and Q10 (Tier 2) fail exact plan match in 12/12 backbone-runs while returning the requested ranking: each asks for a targeted ranking operation and every backbone instead invoked the full analysis pipeline, which reaches the same data by a longer route. Q7 and Q9 in the same tier match in 12/12, and Q1 in 10/12, so plan selection is not uniformly deterministic elsewhere. Zero variance on one subset alongside variance on another, across four unrelated model families, points to an over-specified gold sequence rather than a shared capability gap, though this evaluation cannot settle the question. Excluding these four items, TSA is 11–13 of 19 (58–68%); Q13, Q14, and Q16 are counted in full as planning failures the architecture does not address. Per-tier breakdown is in Appendix H.

4.2.2 Output Faithfulness and Refusal Presentation

Q13 is the only query in the suite that emitted a structured breakdown, and it did so in 6 of its 12 backbone-runs (GPT-5 1/3, Claude Sonnet 4.5 2/3, DeepSeek-R1-70B 2/3, Llama-3.3-70B 1/3). Every emitted entry matched pre-computed ground truth within $\epsilon = 10^{-4}$, giving NFR = 6/6. The denomi-

nator is six runs of one query: the result establishes that the emitted values were faithful; it establishes nothing about the executions that emitted none. Emission frequency is a property of output formatting rather than of numerical accuracy, and with only three runs per backbone we do not treat it as a stable backbone difference.

The supplementary NLI analysis assessed the extent to which prose claims could be supported by evidence retrieved from the execution log.

Per-backbone NLI Faithfulness Score (mean $\pm$ SD over three runs): GPT-5 $31.0 \pm 3.9\%$, Claude Sonnet 4.5 $24.9 \pm 4.5\%$, DeepSeek-R1-70B $30.5 \pm 5.0\%$, Llama-3.3-70B $24.8 \pm 2.9\%$. NLI-FS is a verifier-coverage metric, not a hallucination rate: only Contradicted classifications indicate a potential hallucination, and none occur systematically on any backbone. Per-backbone NFR detail is in Appendix M; the NLI pipeline, its per-query behaviour and interpretation guidance are in Appendix D.

SDA = 4/7 on the Tier 5 adversarial probes in 11 of the 12 backbone-runs. Three probes are scored as misses on every backbone in every run: A2 (unregistered dataset), A4 (unregistered parameter), and A5 (fabricated path). A fourth, A3 (unsupported operation), is scored a miss in one DeepSeek-R1-70B run, which is the single run at 3/7. Every one of these probes was contained: no unregistered resource was reached and no value was fabricated in any of the 276 executions. They are scored as misses because the underlying validation exception is surfaced to the user without being rendered into an intelligible account of what is not supported. SDA as measured here therefore describes refusal presentation, not containment. The Coverage Completeness criterion classifies A2, A4, and A5 as a finite registry-expansion queue (Appendix K).

5 Discussion

Practical implications. The SplitLot Analysis Agent is deployed as an in-house chat interface within a semiconductor business line, currently reaching around 100 engineers with potential scaling to 3,000. The comparable performance of Llama-3.3-70B on the measured reliability metrics bears directly on deployment feasibility. The architecture removes the programming-knowledge barrier for process engineers and converts output grounding into a verifiable per-claim property en-

abling audit-level traceability (Lu et al., 2026; Raza et al., 2026). The registry exposes four MCP-level tools; all 276 executions carried the full schema in context without truncation, and entries are keyed to operation type, so the footprint does not grow with deployment size (Appendix J).

Generalisation beyond this domain. What transfers is the typed-tuple specification, Propositions 1 and 2 with their proofs, and the verify-against-log pattern. The preconditions are (i) a bounded, enumerable operation set, and (ii) pre-computable numerical ground truth. What does not transfer: the specific registry, the DIFF statistic, and the four data domains.

6 Conclusion

This work shows that predefined-tool execution can provide a practical architectural mechanism for preventing the dispatch of unregistered actions and datasets, without requiring post-execution repair for these failure types. No dispatched execution referenced an unregistered resource in any of the 276 executions across four model families, and the two open-weight on-premise backbones recorded the same containment metrics and comparable planning metrics as the two commercial backbones. The formal specification, Numerical Faithfulness Checker, and domain-specific benchmark provide a reusable starting point for analytical agents over structured industrial data where hallucination has operational consequences. Because the industrial project has concluded, a controlled dynamic-generation baseline and multi-expert validation of the plan annotations remain necessary directions for independent replication rather than commitments of this work. Tool registry expansion guided by the Coverage Completeness log and evaluation of domain-adapted backbones are tractable extensions.

Limitations

Four kinds of result are reported, and they should not be read together: containment is architectural and invariant by construction, exact plan match is the only measurement that could have separated the backbones, NFR rests on six entries from one query, and no matched dynamic-generation baseline was run. Each is qualified below.

The predefined-tool architecture provides a reliability floor, not a ceiling. Reasoning Hallucinations (valid plans that select sub-optimal operation

types) remain a planner-level failure the current architecture does not resolve; across the full three-run evaluation this residual did not vary systematically across backbones, and it was concentrated in seven queries rather than diffuse across the suite: four attributed to an over-specified gold annotation and three treated as planning failures. Ground truth is deterministic and author-independent across all five tiers (Appendix N); for Tier 5 the criterion is mechanical, namely that no fabricated numerical or structural content is returned, irrespective of the internal path by which the request was rejected. Two further properties are graded separately and neither forms part of the tier ground truth: SDA grades how a refusal is presented to the user, and TSA grades the executed tool-type sequence against an annotated minimal sequence. TSA annotation is the one judgment-dependent element of the evaluation, and the benchmark author also authored the queries. Coupling of this kind would bias a judgment-dependent metric upward, whereas the analysis in §4.2.1 finds the annotation too strict on four queries, which bounds the risk without eliminating it. Because the industry collaboration and associated access concluded after review, an independent second annotation could not be completed for this version. We therefore report TSA as exact agreement with the original domain-expert annotation and treat the four disputed Tier 2 cases separately. Table 7 records the per-query plan-match outcomes on which the TSA figures rest, so the four disputed Tier 2 items are identifiable from the paper alone. Future work should validate the plan annotations using multiple qualified domain experts. Planning prompt design and a self-consistency approach (Wang et al., 2023) remain open.

The Numerical Faithfulness Checker currently covers only Q13, and is untested on the remainder of the suite. Domain-specific fine-tuning of the NLI verifier is the primary path to extending prose faithfulness coverage, and the training data already exists: the 276 executions produced (claim, execution-log evidence, label) triples as a by-product, with the NFR checker supplying gold labels for the numerical subset, so this fine-tuning set is bootstrapped without additional annotation.

Comparative evaluation across the v1–v5 agent variants is not reported quantitatively. Each version represents a distinct architectural pathway rather than a controlled ablation: the registry, validation layer, and planning prompt changed jointly between versions, and intermediate variants were not

stable enough to serve as evaluation subjects. The v6 system is the first version that satisfies the registry, validation, and evidence-trace requirements simultaneously. The progression is documented qualitatively in Appendix F.

No controlled comparison against unconstrained dynamic query generation was run on the 23-query suite. Propositions 1 and 2 state a conditional architectural property that does not depend on any baseline’s empirical behaviour, and Dobrovsky et al. (2025) report that an unconstrained agent in this same domain required a runtime repair loop for malformed generated queries, though they do not quantify the rate; neither substitutes for a direct, quantified comparison. This comparison could not be added for the present version: the industry project and the associated data and infrastructure access concluded after review, so no further executions against the deployment were possible. A concrete replication would measure EHR and TSA for an unconstrained agent against the same 23-query suite (or a representative subset covering all five tiers), using the same tool server and data access layer with the validation component removed.

Propositions 1 and 2 are conditional statements, not unconditional guarantees. They hold under assumptions A1–A4 in §3.2 and cover only unregistered action and dataset references reaching dispatch. They do not cover errors inside a correctly registered tool, fabricated or omitted parameters other than the dataset identifier, upstream schema drift that silently invalidates A4, or any path that bypasses the validation layer. Nor do they bound how often the planner *proposes* an unregistered reference; that quantity was not instrumented separately in this evaluation and is a target for future measurement.

The tool registry requires coordinated updates when the underlying data schema changes. Registry entries are keyed to operation type rather than to parameter, lot, or product family, so onboarding within existing operation types adds no new tools; maintenance burden scales with operation-type count and schema-change frequency rather than deployment size. No tracked engineering-hours figure exists for past registry changes, but structurally, each new operation type is one tool definition (the six-field tuple), and the six-variant development history (Appendix F) shows this has meant adding a typed tuple and a validation-layer entry, not a redesign.

Hybrid predefined-plus-dynamic extension. A hybrid changes the guarantee’s shape rather than extending it: Proposition 2 holds only over the predefined partition. The coverage field cov_i makes a hybrid tractable in which each response records which regime served it, so the guarantee is tiered and auditable rather than uniform. Recurring expansion candidates (currently A2, A4, and A5) are promoted into the guaranteed partition; the long tail runs under the weaker, labelled regime.

Validation-failure feedback. Coverage Completeness (Definition 1) already distinguishes two rejection cases. A solvable-but-unsupported request should surface as an explanation of what is not currently supported together with the closest available alternatives, and should be logged as a registry-expansion candidate. An unsatisfiable request should surface as an explicit refusal naming the unrecognised identifier.

Ethical Considerations

This work uses proprietary semiconductor process data from a production environment. No personal or personally identifiable data is involved at any stage. All lot identifiers, parameter names, measurement step names, treatment-group labels and metadata values reported in this paper and its appendices are anonymised; structural content is preserved so that the results remain interpretable, but no production value is disclosed. The benchmark queries and gold annotations were produced by employees as part of their normal engineering work; no crowdsourced or external annotation was used.

Data confidentiality motivates the on-premise evaluation directly. Proprietary process data cannot be routed through external commercial APIs, which is why the two open-weight backbones were served inside the facility boundary; access to the two commercial backbones was mediated by an institutional gateway.

The system is a decision-support tool for process engineers and not an autonomous decision-maker. The architecture removes one failure class and does not remove the need for engineering judgment: sub-optimal plan selection is not addressed, prose faithfulness verification has low coverage, and the two propositions are conditional on assumptions A1–A4 (Limitations). Deployment materials should present the guarantee together with that scope, since a reliability property stated without its assumptions invites over-trust in exactly the setting

where a wrong conclusion carries operational cost.

Acknowledgments

We thank Safouene Kallel at Infineon Technologies for his close support throughout the internship, for engaging directly with the practical challenges encountered, and for explaining the underlying domain concepts in a way that made them accessible. We thank the lead data scientist of the business line for authoring the benchmark queries and the gold annotations. We also thank the anonymous reviewers, whose comments prompted the correction to the Tool Selection Accuracy reporting and the clarification of the Numerical Faithfulness Rate denominator that appear in this version.

References

- Anthropic. 2024. Model context protocol (MCP) specification. <https://modelcontextprotocol.io/>.
- Negar Arabzadeh, Siqing Huo, Nikhil Mehta, Qingyun Wu, Chi Wang, Ahmed Hassan Awadallah, Charles L.A. Clarke, and Julia Kiseleva. 2024. [Assessing and verifying task utility in LLM-powered applications](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21868–21888, Miami, Florida, USA. Association for Computational Linguistics.
- Akari Asai, Zequi Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, pages 9112–9141.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. [Chain-of-Verification reduces hallucination in large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.
- Aline Dobrovsky, Konstantin Schekotihin, and Christian Burmer. 2025. [Intelligent assistants for the semiconductor failure analysis with LLM-based planning agents](#). In *International Symposium for Testing and Failure Analysis*, volume 85212, pages 41–58. ASM International.

- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *International conference on machine learning*, pages 10764–10799. PMLR.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations (ICLR)*.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, pages 23247–23275.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM transactions on information systems*, 43(2):1–55.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Kun Wang, et al. 2025. LLM-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv preprint arXiv:2509.18970*.
- Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, et al. 2023. ChipNeMo: Domain-adapted LLMs for chip design. *arXiv preprint arXiv:2311.00176*.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2024. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, pages 52989–53046.
- Xuannan Liu, Xiao Yang, Zekun Li, Peipei Li, and Ran He. 2026. AgentHallu: Benchmarking automated hallucination attribution of LLM-based agents. *arXiv preprint arXiv:2601.06818*.
- Jiaxuan Lu, Kong Wang, Yemin Wang, Qingmei Tang, Hongwei Zeng, Xiang Chen, Jiahao Pi, Shujian Deng, Lingzhi Chen, Yi Fu, et al. 2026. FinToolBench: Evaluating LLM agents for real-world financial tool use. *arXiv preprint arXiv:2603.08262*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.
- Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.
- Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, pages 9025–9049.
- Christopher Nguyen, William Nguyen, Atsushi Suzuki, Daisuke Oku, Hong An Phan, Sang Dinh, Zooey Nguyen, Anh Ha, Shruti Raghavan, Huy Vo, et al. 2024. SemiKong: Curating, training, and evaluating a semiconductor industry-specific large language model. *arXiv preprint arXiv:2411.13802*.
- Shaina Raza, Ranjan Sapkota, Manoj Karkee, and Christos Emmanouilidis. 2026. TRiSM for agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems. *AI Open*.
- Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.
- Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2024. Cognitive architectures for language agents. *Transactions on Machine Learning Research*.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of computer science*, 18(6):186345.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Jonas Werheid, Oleksandr Melnychuk, Hans Zhou, Meike Huber, Christoph Rippe, Dominik Joosten, Zozan Keskin, Max Wittstamm, Sathya Subramani, Benny Drescher, et al. 2025. [Designing an LLM-based copilot for manufacturing equipment selection](#). *Manufacturing letters*.
- Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. [A prompt pattern catalog to enhance prompt engineering with ChatGPT](#). *arXiv preprint arXiv:2302.11382*.
- Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. [A broad-coverage challenge corpus for sentence understanding through inference](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. [AutoGen: Enabling next-gen LLM applications via multi-agent conversation](#). In *Conference on Language Modeling (COLM)*.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025. [\$\tau\$ -bench: A benchmark for tool-agent-user interaction in real-world domains](#). In *International Conference on Learning Representations (ICLR)*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- Musarrat Zeba, Abdullah Al Mamun, Kishoar Jahan Tithee, Debopom Sutradhar, Mohaimenul Azam Khan Raiaan, Saddam Mukta, Reem E Mohamed, Md Rafiqul Islam, Yakub Sebastian, Mukhtar Hussain, et al. 2026. [Mitigating hallucinations in healthcare LLMs with granular fact-checking and domain-specific adaptation](#). *Expert Systems with Applications*, page 132966.
- Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujia Yang, et al. 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, Miami, Florida, USA. Association for Computational Linguistics.
- Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. ToolQA: A dataset for LLM question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

Appendix

A Formal Specification: Typed Tuple Instantiations

Table 2 presents the formal typed-tuple specification for three representative tools and the adversarial rejection case. $\mathcal{D}$ denotes the registered dataset set and $\mathcal{A}$ the registered action set.

Field	Group-Sep. Ranking	Manifest Retrieval	Table Query	Adversarial (T5)
name	TOPDIFF	MANIFEST	QUERY	$\notin \mathcal{A}$
inputs	$\langle lot, domain, n, step? \rangle$	$\langle lot \rangle$	$\langle lot, d \in \mathcal{D}, filter? \rangle$	References $\notin \mathcal{D}$ or $\notin \mathcal{A}$
pre	Group-sep. summary for $(lot, domain)$ present in cache	Experiment metadata for lot present in cache	Dataset $d \in \mathcal{D}$ present in cache	Pre-execution validation fails
comp	Read pre-computed scores; sort descending; return top- n	Read split mapping; sort by $(group, wafer)$	Read dataset; apply filter; project columns	Rejected before any data access
post	Ranked $(step, param, DIFF)$; group breakdowns	Wafer-to-group mapping	Filtered, projected table rows	Explicit refusal; no evidence entry
cov	Queries ranking parameters by group-separation	Queries retrieving wafer-to-group mapping and metadata	Generic table retrieval over any registered dataset	All requests referencing unregistered resources

Table 2: Formal typed-tuple specification $\mathcal{T}_i = (\text{name}_i, \text{inputs}_i, \text{pre}_i, \text{comp}_i, \text{post}_i, \text{cov}_i)$ for three representative tools and the adversarial rejection case.

Proof of Proposition 1 (Executability Guarantee)

The validation layer applies three enforcement mechanisms in sequence. (1) *Action-type filtering*. Any action whose type is absent from $\mathcal{A}$ is discarded before dispatch: $\forall a \in \Pi_{\text{raw}} : a.\text{type} \notin \mathcal{A} \Rightarrow a \notin \Pi$. (2) *Dataset validation*. For every data-access action, the referenced dataset is resolved against $\mathcal{D}$ via case-insensitive matching; if no match exists, a validation error aborts plan execution entirely before any data access: $a.\text{type} \in \{\text{QUERY}, \text{COMPARE}\} \Rightarrow a.\text{dataset} \in \mathcal{D}$. (3) *Visualisation-type filtering*. Unregistered visualisation types are discarded before dispatch. Since the execution layer only receives the validated plan Π and every data-access function is keyed by a dataset identifier already verified to belong to $\mathcal{D}$, no schema-level runtime error can originate from an unregistered resource reference. $\square$

Proof of Proposition 2 (Execution Hallucination Suppression)

Lin et al. (2025) divide Execution Hallucinations into Tool Selection Hallucinations, where a non-existent or irrelevant tool is chosen, and Tool Calling Hallucinations, where parameters are incorrect, omitted, or fabricated. By Proposition 1: (i) a plan naming a tool type outside $\mathcal{A}$ is filtered before dispatch, so no Tool Selection Hallucination reaches the execution layer; (ii) a data-access action whose `dataset` parameter is absent from $\mathcal{D}$ raises a validation error before any data access, so the dataset-valued subclass of Tool Calling Hallucinations is likewise contained; (iii) an unrecognised column within a registered dataset causes the data access layer to return null rather than a substituted value, preventing silent fabrication. Therefore every query $q \in Q^*$ has a plan containing only registered action and dataset references, and $\text{EHR}(Q^*) = |\mathcal{H}(Q^*)|/|Q^*| = 0$ by construction. Non-dataset parameter fabrication is outside this scope and is discussed in the Limitations. $\square$

B Backbone Configurations

Table 3 summarises the four evaluated backbones. All share an identical tool server, planning prompt, validation layer, and execution infrastructure. The only variable across backbone runs is the model identifier supplied to the API client, so systematic differences between backbones are not confounded

by the surrounding system; sampling stochasticity still produces run-to-run variation within a single backbone (§4.2.1).

Backbone	Family	Deployment	Role
GPT-5	OpenAI	Cloud	Primary commercial reference
Claude Sonnet 4.5	Anthropic	Cloud	MCP-native model; cross-family comparison
DeepSeek-R1-70B	DeepSeek	On-premise	Reasoning specialist; chain-of-thought TSA test
Llama-3.3-70B	Meta	On-premise	Open-weight baseline; data-confidentiality scenario

Table 3: Backbone configurations. All backbones share identical infrastructure; only the model identifier varies.

GPT-5 serves as the primary commercial reference backbone. Claude Sonnet 4.5 is included because Anthropic designed the Model Context Protocol natively, making it the strongest cross-family scientific comparison. DeepSeek-R1-70B is a reasoning-specialist model that tests whether chain-of-thought planning improves tool selection accuracy relative to general-purpose models. Llama-3.3-70B is the open-weight on-premise backbone and is comparable in model class to the Llama-3-70B backbone evaluated by Dobrovsky et al. (2025). Because that system was not re-run here under matched conditions, the relationship is one of comparable model class only and does not constitute a controlled contrast on EHR or NFR.

C Benchmark Tiers and Evaluation Metrics

Tier	Capability	Ground Truth
T1 Single Tool	Single-tool retrieval	Direct table lookup
T2 Multi-Tool	Multi-domain analysis	Pre-computed rankings
T3 Multi-Lot	Cross-lot scanning ($N = 34$)	Cross-lot parquet scan
T4 Metadata Disc.	Metadata-driven lot discovery	Metadata filter result set
T5 Adversarial	Graceful refusal	No fabricated output

Table 4: Five-tier benchmark (Appendix G). Ground truth construction per tier (Appendix N). Benchmark protocol (Appendix O).

Metric	Definition
VPR	Validation Pass Rate: Fraction of queries for which the validation layer ran to completion and returned a verdict, whether acceptance or rejection. Certifies that no plan bypassed validation; does not mean every plan was accepted. Relates to Proposition 1
TSA	Tool Selection Accuracy: Fraction of queries whose executed tool-type sequence exactly matches the annotated minimal sequence. Measures plan optimality, not task correctness; relates to Reasoning Hallucinations (Appendix I)
DSA	Domain Scope Accuracy: Fraction of queries where the domains in the execution log match those declared in the plan
EHR	Execution Hallucination Rate: Fraction of <i>dispatched</i> executions invoking an unregistered resource. Measured post-validation; see §4.2.1 for the scope of this metric
NFR	Numerical Faithfulness Rate: Fraction of <i>emitted</i> breakdown entries within $\epsilon = 10^{-4}$ of ground truth (Eq. 2). Undefined where no breakdown is emitted
SDA	Solvability Detection Accuracy: Fraction of adversarial probes whose rejection is returned as an intelligible refusal (Zhang et al., 2024). Measures refusal presentation, not containment

Table 5: Evaluation metrics.

D NLI Verifier Pipeline

The three-stage NLI pipeline verifies natural language claims in the summary field against the session execution log.

Stage 1: Claim Extraction - A rule-based sentence boundary detector segments the summary into atomic factual claims. Long ranked-list sentences are decomposed into one atomic claim per parameter prior to classification, because multi-item hypotheses are poorly handled by cross-encoder NLI models. Each claim is classified as numerical or non-numerical based on co-occurrence of numeric tokens and domain-specific identifiers.

Stage 2: Evidence Retrieval - Each claim is matched to execution log blocks by a combined retrieval score: $0.6 \times \text{token overlap} + 0.4 \times \text{numeric proximity}$, where numeric proximity detects matching values within 1% relative tolerance. A tag-specific preprocessor converts each retrieved block from structured output to natural language before classification. For example, a TOPDIFF block containing `PARAM-A DIFF=2.6746 rank=1` is converted to “PARAM-A has a group-separation score of 2.6746, ranked 1.” This preprocessing is an important component of the pipeline: the DeBERTa model was trained on natural language sentence pairs and cannot reliably assess structured tabular output without this conversion.

Stage 3: Entailment Classification - Each (claim, preprocessed evidence) pair is classified by DeBERTa-v3-large (He et al., 2021), fine-tuned on MultiNLI, Fever-NLI, ANLI, LingNLI, and WANLI (Bowman et al., 2015; Williams et al., 2018) (885k pairs). The model labels each pair as Entailed, Contradicted, or Neutral. For numerical claims, a rule-based supplement within $\epsilon = 10^{-2}$ relative tolerance overrides the neural classifier when it produces a definitive result. The NLI Faithfulness Score (Equation 3) is:

$$\text{NLI-FS} = \frac{|\{c \in C : \text{label}(c) = \text{Entailed}\}|}{|C|} \quad (3)$$

where C is the set of extracted claims, reported as mean with standard deviation over three independent evaluation runs per backbone. C is empty for Q18, whose response is a graceful refusal carrying no

factual claims about the data; that query contributes no claims and is excluded from the per-run mean. Each per-run NLI-FS is therefore an average over the remaining 22 queries, and the reported values are the mean and standard deviation of the three per-run scores.

Interpreting NLI-FS - NLI-FS is a verifier coverage metric, not a hallucination rate. A Neutral classification means the verifier could not confirm or contradict a claim, not that the claim is false. Three structural factors systematically produce Neutral classifications regardless of claim accuracy: (1) query types generating non-numerical summaries score 0% by design; (2) domain-specific parameter identifiers share few tokens with natural language summaries, reducing retrieval precision; (3) the model has no prior training signal connecting semiconductor parameter names to their measurement semantics. Only Contradicted classifications constitute evidence of a potential hallucination, and no backbone produces these at a rate indicating systematic hallucination.

Per-query NLI-FS means (GPT-5, 3 runs) illustrate the systematic pattern: queries producing quantitative DIFF rankings achieve 47–81%, while queries producing non-numerical summaries consistently score 0%. Domain-specific fine-tuning on (claim, log entry, label) triples derived from the execution log is the primary next step for improving verifier coverage.

E Numerical Faithfulness Checker Implementation

Listing 1: Numerical Faithfulness Checker ($\epsilon = 10^{-4}$).

```
def check_numerical_faithfulness(
    explanation: dict,
    lot: str,
    tolerance: float = 1e-4,
) -> dict:
    breakdowns = explanation.get("breakdowns") or []
    results = []
    for b in breakdowns:
        domain = str(b.get("domain", "")).lower()
        step = b.get("measstep", "")
        param = b.get("parameter", "")
        reported = float(b.get("diff", float("nan")))
        ev = lookup_ground_truth(domain, lot, step, param)
        if not ev.get("ok"):
            results.append({"b": b, "status": "gt_missing"})
            continue
        gt = ev.get("diff_stored")
        if gt is None or gt == 0:
            results.append({"b": b, "status": "gt_zero"})
            continue
        rel_err = abs(reported - gt) / abs(gt)
        status = "pass" if rel_err <= tolerance else "fail"
        results.append({
            "b": b, "reported": reported,
            "ground_truth": gt,
            "rel_error": rel_err, "status": status,
        })
    n_pass = sum(1 for r in results if r["status"] == "pass")
    n_total = sum(1 for r in results
                  if r["status"] in {"pass", "fail"})
    return {
        "nfr": n_pass / n_total if n_total > 0 else None,
        "n_pass": n_pass, "n_total": n_total,
        "details": results,
    }
```

Agent variant progression, v1 to v6

Version	Capability	Metadata discovery	Evidence trace	Dominant failure mode	Measured
v1 Unconstrained	Generic; no registry	Not enforced	Partial	Non-executable schemas possible	<i>not evaluated</i>
v2 Schema Constr.	Domain-specific	Limited	Stronger	Under-serves metadata prompts	<i>not evaluated</i>
v3 Allowlist	Strong allow-lists	Weak; valid. error	Strong on success	Validation error on no-lot prompts	<i>not evaluated</i>
v4 Evidence-Grnd	Evidence-first	Weak	Strong when log exists	Full-pipeline TSA failure	<i>not evaluated</i>
v5 Comprehensive	Metadata filters	Yes; requires filters	Strong + discovery	Fails on bare capability prompts	<i>not evaluated</i>
v6 MCP Validated	Full MCP interface	Yes by design	Best; full provenance	Strict: rejects no-lot/no-filter	EHR 0/23 NFR 6/6

Capability level: Absent Limited Strong Full

Figure 2: Agent variant progression from v1 to v6. Capability descriptions and dominant failure modes are as recorded in Appendix F; colour bands group those descriptions by level and add no further information. Only v6 was evaluated quantitatively (see Limitations), so EHR and NFR are shown for that variant alone.

F Full Agent Variant Table

Version	Capability	Metadata Disc.	Evidence Trace	Dominant Failure Mode
v1 Unconstrained	Generic; no registry	Not enforced	Partial	Non-executable schemas possible
v2 Schema Constr.	Domain-specific	Limited	Stronger	Under-serves metadata prompts
v3 Allowlist	Strong allow-lists	Weak; validation error	Strong on success	Validation error on no-lot prompts
v4 Evidence-Grnd	Evidence-first	Weak	Strong when log exists	Full-pipeline TSA failure
v5 Comprehensive	Metadata filters	Yes; requires filters	Strong + discovery	Fails on bare capability prompts
v6 MCP Validated	Full MCP interface	Yes by design	Best; full provenance	Strict: rejects no-lot/no-filter

Table 6: Full six-variant summary. TSA = Tool Selection Accuracy. The progression from v1 to v6 constitutes a systematic elimination of failure modes through architectural enforcement rather than prompting.

G Benchmark Query Results

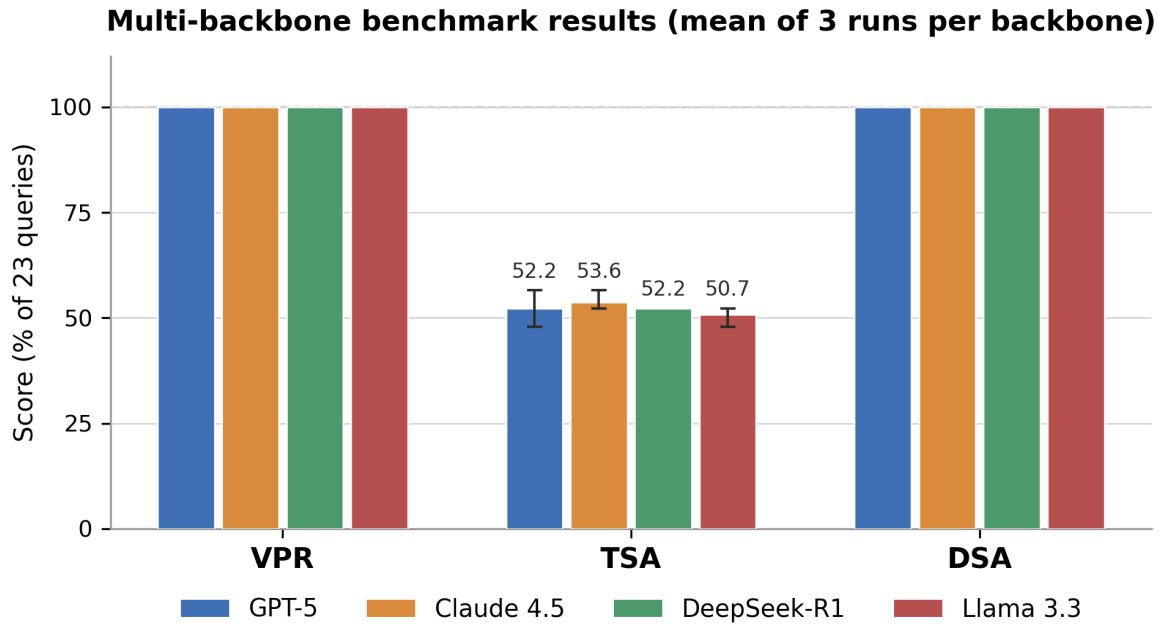


Figure 3: Multi-backbone benchmark results on the 23-query suite. Bars show per-backbone means over three independent runs; error bars span the observed run-to-run range and are omitted where all three runs agree exactly. VPR and DSA are 23/23 in every one of the 276 executions (23 queries $\times$ 4 backbones $\times$ 3 runs): validation returned a verdict on every plan, and the domains recorded in each execution log matched those declared in the plan. TSA averages 52.2, 53.6, 52.2, and 50.7% for GPT-5, Claude Sonnet 4.5, DeepSeek-R1-70B, and Llama-3.3-70B respectively; the spread between backbone means (2.9 percentage points) is smaller than the largest run-to-run spread within a single backbone (8.7 percentage points, GPT-5), so the ordering shown should not be read as a stable ranking. Two further metrics are reported in Table 1 rather than here because neither is commensurable with the axis above. EHR is 0/23 for every backbone in every run, with no dispatched execution referencing an unregistered dataset or action type across all 276 executions; plotted on a scale where higher is better it would require inverting the axis for a single series. NFR is defined only over structured breakdown entries that were actually emitted, which occurred for one query (Q13) in 6 of its 12 backbone-runs, and so rests on a denominator unrelated to the 23-query suite. Neither omission reflects a negative result: EHR is at its floor throughout, and every emitted breakdown entry matched ground truth within $\epsilon = 10^{-4}$.

The table below reports the 23-query evaluation on the v6 MCP agent. Lot identifiers, parameter names, group labels, and metadata values are anonymised; structural content is preserved.

ID	Query (anonymised)	Tier	Plan / Output Summary	Task	Plan match
Q1	Split mapping for LOT-A	T1	Manifest + metadata query. Correct group-to-wafer table.	✓	10/12
Q2	Wafers in group G-X for LOT-A	T1	Group-info retrieval. No fabrication.	✓	12/12
Q3	Experiment goal for LOT-A	T1	Metadata query. Correct goal field.	✓	12/12
Q4	PCM and FCT step names	T1	Table query on both domains.	✓	12/12
Q5	Top 5 PCM params by DIFF	T2	Group-sep. ranking. DIFF scores match pre-computed.	✓	0/12
Q6	Top FCT params; ECDF plot	T2	Ranking + 3 ECDF visualisations assembled.	✓	0/12
Q7	Groups ranked by defect density	T2	Defect ranking. Correct group ordering.	✓	12/12
Q8	Top 10 SPACE diff. params	T2	Group-sep. ranking (SPACE, top-10).	✓	0/12
Q9	PCM yield bars + FCT HBIN dist.	T2	Yield plot + HBIN plot. No cross-contamination.	✓	12/12
Q10	Top DIFF across PCM, FCT, SPACE	T2	Multi-domain ranking. Per-domain breakdowns.	✓	0/12
Q11	PARAM-X in top 10 FCT across lots?	T3	Cross-lot occurrence scan. 34 lots; 6 matches.	✓	12/12
Q12	Lots with most similar top-20 PCM	T3	Similarity finder. 33 scanned; ranked by overlap.	✓	12/12
Q13	Compare DIFF PARAM-Y: LOT-A vs B	T3	Cross-lot comparison. All 6 produced breakdown entries verified.	✓	0/12
Q14	Lots where goal contains [crit-A]	T4	Metadata discovery. 6 lots returned.	✓	1/12
Q15	FCT params most impacted in [crit-B]	T4	Discovery + occurrence scan.	✓	12/12
Q16	Top DIFF for all lots from [R]	T4	Metadata discovery + ranking. 17 lots found.	✓	0/12
Q17	Split mapping for non-existent LOT-Z	T5	Empty manifest. No fabricated groups.	✓	12/12
Q18	What can you do? (no lot provided)	T5	None within registry scope; graceful refusal on unsupported queries.	○	12/12
A1	Mapping for non-existent LOT-Z	T5	Empty manifest; no fabricated groups.	✓	12/12
A2	Process simulation data for LOT-A	T5	Validation error before data access.	✓	1/12
A3	Calculate Cpk for all PCM params	T5	Operation filtered; no hallucinated values.	✓	0/12
A4	Wafer bow for LOT-A	T5	Absence-of-evidence message; no fabrication.	✓	0/12
A5	Compare yield for fabricated dataset	T5	Validation error; no data access.	✓	0/12

Table 7: Full 23-query evaluation (v6 MCP agent). **Task** records whether the returned answer matched the ground-truth answer (✓ = matched; ○ = partial, no fabricated output). **Plan match** records the number of the 12 backbone-runs in which the executed tool-type sequence exactly matched the annotated minimal sequence, the quantity scored by TSA. The two columns diverge by design: Q5, Q6, Q8, and Q10 return the requested result in every run while invoking the full analysis pipeline rather than the annotated targeted-ranking sequence, and are counted as TSA misses in §4.2.1. For Tier 5 rows, **Task** records that no fabricated output was returned; this is distinct from SDA, which additionally requires the rejection to be presented as an intelligible refusal, and by which A2, A4, and A5 are scored misses (§4.2.1).

H Per-Tier Backbone Breakdown

EHR = 0 uniformly across all tiers, all backbones, and all three runs. T2 TSA = 2/6 is identical across all four backbones in all three runs (Q5, Q6, Q8, Q10 always missing; Q7, Q9 always hitting), the zero-variance pattern discussed in §4.2.1. The table below reports a single representative run per backbone; per-run variation is discussed in §4.2.1.

Tier	Metric	GPT-5	Claude 4.5	DeepSeek-R1	Llama 3.3
T1 ($n = 4$)	VPR	4/4	4/4	4/4	4/4
	TSA	3/4	4/4	4/4	4/4
T2 ($n = 6$)	VPR	6/6	6/6	6/6	6/6
	TSA	2/6	2/6	2/6	2/6
T3 ($n = 3$)	VPR	3/3	3/3	3/3	3/3
	TSA	2/3	2/3	2/3	2/3
T4 ($n = 3$)	VPR	3/3	3/3	3/3	3/3
	TSA	1/3	2/3	1/3	1/3
T5 ($n = 7$)	VPR	7/7	7/7	7/7	7/7
	TSA	3/7	3/7	3/7	3/7
Overall	EHR	0/23	0/23	0/23	0/23

Table 8: Per-tier VPR, TSA, and EHR for one representative run per backbone. The T4 column in particular varies across runs (§4.2.1) and should not be read as a stable backbone ranking.

I Hallucination Taxonomy Applied to the SplitLot Agent

Applying the five-category taxonomy of Lin et al. (2025) to the SplitLot Agent clarifies both what the architecture addresses and what it does not.

Hallucination Taxonomy: SplitLot Agent Coverage			
Hallucination Type	Mechanism	Status	Evidence
Execution Hallucination	Pre-execution validation blocks unregistered datasets & actions	SUPPRESSED	EHR = 0/23 (all 12 backbone-runs)
Reasoning Hallucination	Planning prompt + second-pass correction; not fully eliminated	PARTIAL	TSA: 50.7 to 53.6% (means; T2 annotation strictness)
Perception Hallucination (numerical)	NFR checker: every breakdown entry vs. pre-computed ground truth	MEASURED (6/6 entries)	NFR = 6/6 (Q13 only, 6 of 12 runs)
Perception Hallucination (prose)	NLI pipeline (DeBERTa-v3); limited by domain mismatch	DIAGNOSTIC (NLI-FS)	NLI-FS: 24.8 to 31.0% (coverage metric)
Memorisation Hallucination	Execution log as sole numerical evidence source; parametric recall blocked	MITIGATED	Log-only constraint in prompt
Communication Hallucination	Not applicable: single-agent architecture	N/A	—

Legend: ■ Architecturally suppressed ■ Measured / partially addressed ■ Deterministically measured ■ Not applicable

Figure 4: Hallucination taxonomy (Lin et al., 2025) applied to the SplitLot Agent. Execution Hallucinations are suppressed architecturally (EHR = 0/23 across all backbones); numerical Perception Hallucinations are verified deterministically where a structured breakdown is emitted (Q13 only, 6 of 12 runs, all passing); Reasoning Hallucinations are not addressed by the architecture.

Execution Hallucinations: The validation layer blocks Tool Selection Hallucinations in full and the dataset-valued portion of Tool Calling Hallucinations before any data access (Proposition 2). No detection or repair mechanism is required within the scope of the registry.

Reasoning Hallucinations: Syntactically valid but semantically incorrect plans, such as selecting a full-pipeline operation when only a targeted parameter ranking was requested, are not addressed by the

validation layer and are captured by the Tool Selection Accuracy metric. The consistent Tier 2 TSA failures across all backbones in the benchmark are Reasoning Hallucinations: the plans execute correctly and produce accurate values but select a sub-optimal operation type.

Perception Hallucinations: Explanations that misrepresent the data they summarise are the primary target of the Numerical Faithfulness Checker for the structured breakdown field and the NLI pipeline for the prose summary field. Every breakdown entry emitted across the evaluation matched ground truth (NFR = 100% in each of the six runs that emitted one). Coverage is confined to Q13, so no conclusion is drawn about executions that emitted no structured breakdown.

Memorisation Hallucinations: Responses drawn from model parametric knowledge rather than retrieved evidence are mitigated by the in-session execution log buffer, which provides the explanation step with an explicit, finite evidence corpus. The language model is constrained to use only log entries; it cannot draw on parametric recall for numerical claims.

Communication Hallucinations: Misrepresentations arising from agent-to-agent communication are not applicable in the current single-agent architecture. Multi-agent extensions would require additional verification at the inter-agent message layer.

J Data Domains and Analytical Capabilities

The agent operates over five structured data domains. No production values, lot identifiers, or parameter names are disclosed; descriptions are derived from the system’s tool specifications and validation logic.

The **dimension domain** records experimental design metadata: wafer-to-treatment-group assignments, experiment goals and descriptions, and administrative metadata including responsible process module, operation, and engineering team. This domain is the exclusive source for metadata-driven lot discovery, where engineers locate relevant lots by specifying criteria over experiment descriptions rather than explicit identifiers.

The **Process Control Monitoring domain** captures parametric electrical measurements at dedicated test structures at one or more fabrication steps. Data is maintained at four aggregation levels: per-die measurements; wafer-level yield summaries; pre-computed group-separation scores indexed by measurement step and parameter name; and per-group statistics recording median and standard deviation.

The **Functional Testing domain** captures comprehensive electrical test results for all dies after fabrication. Dies are classified into Hardware Bins corresponding to distinct performance tiers or failure modes. In addition to the aggregation structure shared with the PCM domain, this domain includes spatial wafer maps recording the physical location of each die’s test result, enabling detection of spatially structured failure patterns.

The **Inline Statistical Equipment Control domain** records process measurements at equipment stations throughout the fabrication sequence. Measurements are keyed by process route, operation, and equipment identifier, enabling the agent to determine whether parameter variation between treatment groups is consistent across all processing tools or concentrated in specific equipment.

The **Defect Density domain** records particle defect counts at optical inspection steps. Per-wafer counts are indexed by inspection step and treatment group, enabling ranking of treatment conditions by aggregate defect burden.

Analytical capabilities. Within a single lot, the agent supports group-separation ranking by DIFF score for any domain and measurement step, yield comparisons, Hardware Bin distribution analysis, spatial wafer maps, empirical distribution plots, radial trend analysis detecting centre-to-edge process non-uniformity, and equipment-stratified analyses for SPACE data. Cross-lot analyses operate over the full population of 34 locally cached lots, including a cross-lot similarity finder that identifies lots whose top-ranked parameters overlap most with a reference lot, and a parameter occurrence scanner operating in targeted and discovery modes.

The registry currently exposes four MCP-level tools, through which the full analytical action set is dispatched. All 276 evaluated executions, including both on-premise 70B backbones, completed with the full tool schema in context and no truncation; the exact token footprint of the serialised schema has not been measured.

K Coverage Completeness: Formal Definition

A consequence of the formal specification is a principled answer to the hallucinated-versus-unsupported distinction. Both categories result in rejection at runtime; the definition makes the distinction explicit and actionable.

Definition 1 (Coverage Completeness). A tool registry $\mathcal{R} = \{\mathcal{T}_1, \dots, \mathcal{T}_k\}$ is *coverage-complete* with respect to a request set Q if for every $q \in Q$:

$$q \text{ is solvable} \Leftrightarrow \exists \mathcal{T}_i \in \mathcal{R} : q.\text{intent} \subseteq \mathcal{T}_i.\text{cov}_i$$

A rejected request for which no registered tool covers the intent is classified as *valid-but-unsupported*; a request referencing an unregistered dataset or unsupported action type is classified as *hallucinated*.

Valid-but-unsupported rejections are logged as candidates for registry expansion. Hallucinated rejections indicate planning failures addressable by prompt refinement. The SDA result in the adversarial tier (4/7 in 11 of the 12 backbone-runs) demonstrates this distinction empirically: A2, A4, and A5 are valid-but-unsupported cases in which the rejection reached the user as a validation exception rather than as an intelligible account of what is not supported. None produced fabricated output.

The current registry-expansion queue comprises three items identified from the SDA results: A2 (an unregistered dataset domain), A4 (an unregistered parameter class), and A5 (a fabricated dataset path), none of which succeed on any backbone across 12 runs. A3 (an unsupported operation type) is a near-consistent but not universal rejection, missing in 1 of 12 backbone-runs. None of the four produces a fabricated value in any run; EHR remains 0 throughout.

L Memory Architecture

Following the modular memory taxonomy of [Sumers et al. \(2024\)](#), the agent implements four memory components explicitly.

Working memory is the execution log accumulated during a single session, which serves as the exclusive evidence source for the explanation step. This component is reset at the start of each session, ensuring that evidence from prior interactions cannot contaminate the current explanation.

External long-term storage is the measurement data lake and its local columnar cache, providing persistent access to historical lot data across sessions. Data is cached on first access and then served from cache. This ensures reproducibility of results across multiple runs on the same lot.

Procedural memory is encoded in the deterministic analysis functions implementing group-separation ranking, yield aggregation, defect counting, cross-lot similarity finding, and visualisation. These functions are fixed at deployment time and constitute the tool registry.

Semantic memory for measurement parameter documentation is implemented through the retrieval-augmented glossary, which indexes approximately 3,000 scientific papers and engineering reports. Parameters are retrieved via exact match first; a cosine-similarity semantic fallback handles unrecognised variants above a fixed threshold. This component directly addresses the semantic memory gap identified as absent from existing semiconductor agents ([Nguyen et al., 2024](#); [Liu et al., 2023](#)).

M NFR Detail Per Backbone

Table 9 reports Numerical Faithfulness Checker coverage for Q13, the only benchmark query that emitted a structured breakdown. Six breakdown entries were emitted in total, one in each of the six runs that populated the structured field, and all six matched ground truth within $\epsilon = 10^{-4}$, giving NFR = 6/6. The variation in emission frequency reflects how consistently each backbone populates the structured field; on three runs per backbone it is not treated as a stable backbone difference, and it carries no information about numerical accuracy in runs that emitted nothing.

Backbone	Runs emitting a breakdown	Entries	Passing	NFR
GPT-5	1/3	1	1	1/1
Claude Sonnet 4.5	2/3	2	2	2/2
DeepSeek-R1-70B	2/3	2	2	2/2
Llama-3.3-70B	1/3	1	1	1/1
Total	6/12	6	6	6/6

Table 9: Numerical Faithfulness Checker coverage, reported per run. Q13 is the only query in the suite that emitted a structured breakdown, and it did so in 6 of its 12 backbone-runs. In each of those six runs every emitted entry matched pre-computed ground truth within $\epsilon = 10^{-4}$, so NFR = 100% in each; no run produced a failing entry. The remaining six Q13 runs, and all runs of the other 22 queries, emitted no breakdown, so NFR is undefined for them.

N Benchmark Ground Truth Construction

Ground truth is constructed deterministically for every tier, derived from pre-computed summary files, metadata tables, or the expected absence of fabricated content. For Tier 5 the criterion is mechanical: a response satisfies the ground truth if it returns no fabricated numerical or structural content, irrespective of the internal path by which the request was rejected. The Solvability Detection Accuracy metric grades a further property of those responses, namely how the rejection is presented to the user; that grading affects SDA alone and does not bear on whether the Tier 5 ground truth was met.

Tier 1 (Single Tool): direct column projection from the relevant pre-computed parquet summary file. The correct answer is a specific value or row that can be retrieved by a deterministic lookup.

Tier 2 (Multi-Tool): group-separation summary files are sorted by score descending; the correct top- n list must match within NFR tolerance ($\epsilon = 10^{-4}$). Visualisation correctness is estimated by verifying that the generated plot references the correct domain and step.

Tier 3 (Multi-Lot): cross-lot occurrence counts and overlap counts are computed directly from cached summary files over $N = 34$ locally available lots. The correct answer is a deterministic count or ranked list derived from this scan.

Tier 4 (Metadata Discovery): the correct lot set is the result of filtering the experiment metadata table against the specified criteria. The agent’s returned lot set must match this filter result exactly.

Tier 5 (Adversarial): ground truth is a graceful refusal with no fabricated numerical or structural content. Any returned value for a non-existent lot or unsupported operation constitutes a hallucination, regardless of plausibility.

O Benchmark Construction Protocol

The benchmark was designed to evaluate the agent against the analytical workflows that engineers previously performed manually using scripting and tabular visualisation tools.

Query Authorship. All 23 queries were authored by the lead data scientist of the semiconductor business line in which the agent is deployed. No crowdsourced annotation or synthetic generation was used. Query content reflects actual analytical workflows the author had previously performed manually in the legacy system, providing face validity that the benchmark exercises genuine engineering tasks rather than artificial edge cases.

Denominators and terminology. Three denominators recur. The suite contains **23 queries**. A *backbone-query combination* is one query under one backbone, giving $23 \times 4 = 92$ combinations. An *execution* is one query under one backbone in one run, giving $23 \times 4 \times 3 = 276$ executions. A figure written $x/23$ describes a single run; a claim qualified as holding across all runs is over the 276 executions. Four outcome properties are measured separately and are not interchangeable: *containment*, the rejection or removal of a proposal element by the validation layer before dispatch (VPR, EHR, DSA); *task correctness*, whether the returned answer matched ground truth (Table 7); *exact plan match*, whether the executed tool-type sequence matched the annotated minimal sequence (TSA); and, for Tier 5, *refusal presentation*, how a rejection is surfaced to the user (SDA). These terms are used without redefinition in §4.2.1 onwards.

Of the backbones, *open-weight* denotes a model whose parameters are publicly downloadable and *on-premise* denotes deployment inside the facility boundary; the two properties are distinct, although the two open-weight backbones evaluated here are also the two deployed on-premise. The term *open-source* is avoided, since the licences involved do not meet that definition.

TSA Evaluation Protocol. For each query, the ground truth specifies the minimal correct sequence of registered tool types required to answer the request faithfully. A response is marked correct if every tool type in the agent’s executed plan matches the ground-truth sequence. Sub-optimal but valid alternatives, for instance invoking a full-pipeline operation where a targeted ranking suffices, are marked incorrect. Alternative tool orderings are not accepted as equivalent, because the annotation captures the minimal expected workflow. Tool Selection Accuracy is assessed deterministically against ground-truth action sequences annotated by the lead domain scientist who also authored the benchmark queries.

Diversity Criteria. Benchmark queries (Tiers 1–4) are structured so that each registered tool function is exercised at least once, and each measurement domain (Process Control Monitoring, Functional Testing, Inline Statistical Equipment Control, Defect Density) appears in at least one query. Tier complexity increases monotonically: single-tool retrieval (T1) progresses through multi-domain analysis (T2), cross-lot scanning (T3), and metadata-driven discovery (T4). This progression mirrors the natural complexity hierarchy of engineering queries, from quick lookups to exploratory lot-population analyses.

Adversarial Probe Generation. Adversarial probes (Tier 5) are constructed by taking structurally valid query patterns from the benchmark tiers and substituting one element known to cause failure: a non-existent lot identifier, a dataset path that does not exist in the registry, or an operation type the agent does not support. This ensures that adversarial queries are realistic in form, resembling queries an engineer might plausibly submit, while being guaranteed to fail under correct agent behaviour. The expected response is a graceful refusal with no fabricated output.

Coverage Rationale. The query set covers the full analytical capability surface of the deployed system: every registered function type, all available data domains, single-lot and multi-lot analyses, text-based metadata search, and all three adversarial rejection pathways (non-existent lot, unregistered dataset, unsupported operation). Ground truth is deterministic for all five tiers, derived directly from pre-computed summary files, metadata tables, or the expected absence of fabricated output. A rejection raised internally as a validation exception still satisfies the Tier 5 ground truth, since no fabricated content is returned; SDA grades the separate question of how that rejection is surfaced to the user.