FROM TRACES TO EXPLANATIONS: A STRUCTURED FRAMEWORK FOR AUDITING AI AGENT BEHAVIOR

Anonymous authors
Paper under double-blind review

ABSTRACT

AI Agents are increasingly deployed in real-world settings, where they interact with external tools and make sequential decisions with limited human oversight. This creates a pressing need for reliable and auditable explanations of what an agent did, how its behavior unfolded, and what evidence supported it. However, traditional Explainable AI (XAI) methods fall short of providing the process-level transparency required for such interactive, multi-step systems. To address this gap, we introduce a trace-grounded structured representation of agent execution that reconstructs behavioral trajectories from lengthy execution traces and provides an auditable basis for faithful natural-language explanations. The representation explicitly separates behavioral commitments from the internal and external evidence, enabling the identification of behavior–evidence mismatches. Relying solely on execution traces, our framework generalizes across agent architectures and environments. Human and automated evaluations across multiple benchmarks and architectures show that our framework produces high-quality, trace-faithful explanations while reliably identifying unsupported claims, unjustified actions, and evidence gaps, outperforming direct LLM-generated explanations.

1 INTRODUCTION

Recent advances in Agentic AI are reshaping the capabilities of AI systems by enabling autonomous goal pursuit, adaptive decision-making, and complex task execution with minimal human oversight (Acharya et al., 2025). Unlike traditional AI models, AI Agents[1] interact with external tools, maintain internal state, coordinate multiple actions, and adapt their behavior over extended execution horizons. While these capabilities unlock significant opportunities across critical domains, they also introduce new challenges for transparency, accountability, and governance (Zhu et al., 2026; Shah et al., 2026), making surface-level transparency no longer sufficient to support user trust or regulatory compliance (Khalid et al., 2026). Recent studies further highlight novel failure modes, including error cascades, responsibility gaps, flawed execution monitoring, and multi-step error propagation that obscures the origins of system behavior and outcomes (Zhu et al., 2026; Shah et al., 2026), highlighting the need for a rethinking of current XAI practices.

Existing explainability approaches, in fact, were primarily designed for static, single-step predictive models and are increasingly inadequate for explaining the behavior of autonomous agents operating in dynamic environments (Chakrabarty, 2025; Raza et al., 2026; Chaduvula et al., 2026). This challenge is further amplified by the interaction paradigm of Agentic AI, which introduces a complex triadic relationship among users, autonomous agents, and external digital resources, requiring explanation mechanisms that account for multi-layered interactions across interconnected systems (Ehsan et al., 2026). For these reasons, researchers have called for a shift from traditional feature-level explanations toward system-level interpretability capable of capturing agent trajectories, reasoning processes, tool interactions, and execution histories (Zhu et al., 2026; Raza et al., 2026; Collaco et al., 2026). However, execution traces are not explanations by themselves: their length, heterogeneity, and low-level structure make them difficult to inspect and interpret directly, motivat-

[1] In this paper, we use *AI agents* and *agentic AI systems* interchangeably, focusing on LLM-based agents capable of autonomous planning and tool use. While some findings may generalize to other autonomous systems, our focus is on tool-using LLM agents, the dominant paradigm of Agentic AI.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ing intermediate representations that preserve the structure of agent behavior while abstracting away implementation-specific details.

In this context, emerging work calls for lifecycle-oriented, human-centered frameworks that explain not only what decisions were made, but why, how, when, and by whom they were influenced (Raza et al., 2026), while cautioning that Chain-of-Thought disclosures may create an illusion of explainability rather than meaningful understanding (Ehsan et al., 2026). Yet, to date, XAI research on Agentic AI remains limited and largely industry-driven, focusing primarily on failure detection and debugging, leaving broader interpretability and accountability needs across regulators, deployers, and end users insufficiently addressed.

For these reasons, we introduce a post-hoc framework centered on a trace-grounded structured representation that reconstructs an agent's behavioral trajectory from logged actions and environmental evidence, providing an auditable basis for faithful natural-language explanations. By operating solely on execution traces, the framework exposes the action–observation sequences underlying agent behavior while remaining applicable across agent architectures, tool ecosystems, and task domains. Overall, our work makes the following contributions:

1. **A trace-grounded structured representation of AI agent behavior.** We introduce an intermediate representation that transforms execution traces into a structured, step-by-step account of an agent's behavioral trajectory. The representation organizes the user goal and its requirements together with the agent's actions, observations, state transitions, and supporting evidence, providing an auditable abstraction of the execution process. Human evaluation shows that the resulting structured reports are substantially more faithful to the underlying traces than equivalent reports generated directly by an LLM.
2. **A taxonomy of agent behavior grounded in behavior–evidence alignment.** We develop a taxonomy that characterizes agent behavior according to the relationship between the agent's behavioral commitments and the internal and external evidence available in the execution trace. The taxonomy captures aligned and mismatched behavior–evidence configurations, providing a principled basis for identifying different forms of agent behavior from execution traces.
3. **Faithful natural-language explanations.** We use the structured representation as an auditable substrate for generating concise natural-language explanations of agent behavior. Across multiple benchmarks and agent architectures, these explanations are judged more faithful, exhaustive, and satisfactory than direct trace-to-text LLM explanations.

2 Related Work

While recent work explores agentic systems for explainability tasks (He & Martens, 2026; Mehta, 2026; Vineis et al., 2026), to the best of our knowledge, no prior work provides a general framework for making agent decision processes transparent to end users. Nevertheless, the need for agent behavior auditability has increasingly emerged across both industry and academia. Recent agentic security incidents provide a concrete illustration: in its investigation of the 2026 Hugging Face incident, OpenAI conducted retrospective reviews of Chain-of-Thought, actions, and final outputs to understand the models' misaligned behaviors (OpenAI, 2026). In parallel, commercial and open-source observability platforms such as Langfuse (Rawert et al., 2023) provide structured traces of multi-step agent workflows, primarily to support developer-side debugging, monitoring, and evaluation. Academic research similarly leverages execution traces to reveal failures missed by outcome-only evaluation (Mazhar et al., 2026) and demonstrates the challenge of automatically diagnosing agent failures (Deshpande et al., 2025). Other approaches leverage traces to verify procedural compliance (Sharma et al., 2026), detect and correct execution errors (Mulian et al., 2026), reconstruct causal graphs for failure localization (Wang, 2026), or support structured issue localization and root-cause analysis (Yang et al., 2026). Although these methods establish the value of execution traces, they remain primarily developer-facing and focus on *why an agent failed*, rather than communicating *how it reached a decision* in a faithful, user-understandable form.

Our framework, instead, treats execution traces as the basis for trace-grounded explanations while retaining a structured representation for inspection and auditing. This perspective aligns with Hu et al. (2026), who identify explicit execution provenance as key infrastructure for responsibility

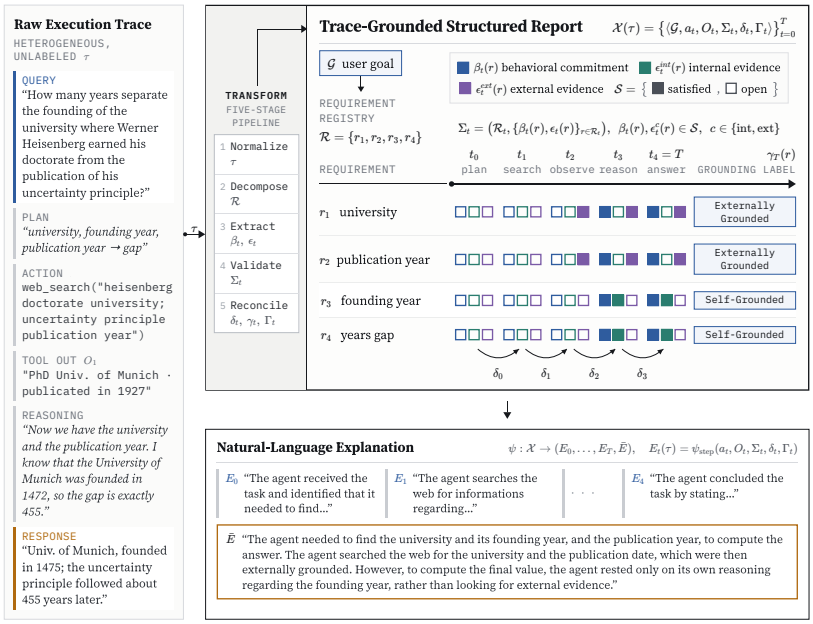

Figure 1: Framework overview. The raw agent execution trace is transformed into a structured report $\mathcal{X}(\tau)$ through trace normalization, requirement decomposition, LLM-based extraction of behavioral commitments and evidence, and deterministic validation and reconciliation. The resulting state transitions and grounding reports $(\Sigma_t, \delta_t, \Gamma_t)$ are rendered by the constrained narrator ψ into per-step explanations $E_1, \ldots, E_T$ and an overall summary $\bar{E}$ (see also example in Appendix D).

attribution in Agentic AI. Their proof-of-concept, however, does not provide a general mechanism for generating user-facing explanations, which is the gap addressed by our work.

3 Trace-Grounded XAI Methodology

In this section, we describe our XAI framework for AI agents, illustrated in Figure 1, introducing its core components and relationships, and its outputs: a trace-grounded structured report and a natural-language explanation rendered from it.

3.1 Structured Representation

Objective. The central objective of our framework is to obtain a structured, trace-grounded representation of the agent's behavioral trajectory, which we call the *structured report* $\mathcal{X}(\tau)$. The report is organized around the goal and its requirements, the actions and observations recorded during execution, and requirement-level states that capture both the agent's behavioral commitments and the evidence available to support them. These components distinguish what the agent appears to commit to from what can be established from the execution trace, while preserving the temporal structure needed to reconstruct how requirements evolve throughout the trajectory. The resulting representation supports both human-readable natural-language explanations $\bar{E}(\tau)$ that summarize the agent's trajectory, which are referred to as *XAI Narratives* in the literature (Silvestri et al., 2025), and grounding diagnostics that identify behavioral–evidence mismatches.

Goal and Requirements. Consider a tool-using LLM agent tasked with achieving a goal G, specified by a user query. We assume that G can be decomposed into a set of atomic, independently verifiable requirements $\mathcal{R} = r_1, r_2, \ldots, r_N$, where each requirement r_i represents a minimal criterion that must be satisfied for the goal to be considered complete. The agent pursues G through iterative interactions with external tools and the environment. Crucially, the framework does not seek to assess the objective correctness of the agent's final answer, but rather to make transparent the process that led to it. Failure detection is likewise not its primary objective. Although the re-

sulting artifacts can help localize potential errors and characterize inconsistencies exhibited by the agent, these signals concern mismatches between the agent's behavior and the evidence available in the trace. They do not, by themselves, establish failures in the underlying tools or the incorrectness of the agent's knowledge.

Event Trace. The fundamental input to our framework is an agent-specific execution trace, generated through the agent's interaction with its environment. This raw trace records agent actions, tool invocations and outputs, together with heterogeneous execution metadata, including timestamps, call identifiers, intermediate states, and framework-specific information. While the structure and granularity of such traces vary across agent implementations, we assume that the recorded information is sufficient to reconstruct the underlying behavioral trajectory. We denote the resulting epistemically meaningful *event trace* by $\tau = (e_1, e_2, \ldots, e_T)$, where each event e_t captures an action, observation, or other interaction that may alter the information available to the agent at that point in the trajectory. By convention, T denotes the terminal step at which the agent produces its final response to the user; all preceding steps $t < T$ correspond to intermediate actions and observations along the trajectory. This representation abstracts from implementation-specific logging details while preserving the information needed to characterize the agent's progression from query to goal.

Behavior–Evidence States and Transitions. The event trace τ provides the basis for constructing a structured representation of the agent's behavioral trajectory. We denote by a_t the identifier of the agent action at step t and by $o_{t,k}$ the identifier of the k-th observation associated with that step. Intuitively, for each requirement and at each point in the execution, we track two complementary questions: whether the agent's behavior treats the requirement as resolved, and whether the trace contains evidence supporting its resolution. Let $\mathcal{S} = \{\texttt{open}, \texttt{satisfied}\}$ denote the set of possible statuses. At each step t, we reconstruct a requirement-indexed knowledge–action state

$$\Sigma_t = \left(\mathcal{R}_t, \{\beta_t(r), \epsilon_t(r)\}_{r \in \mathcal{R}_t}\right), \qquad \beta_t(r), \epsilon_t^c(r) \in \mathcal{S}, \quad c \in \{\text{int}, \text{ext}\}, \tag{1}$$

where $\mathcal{R}_t \subseteq \mathcal{R}$ contains the requirements introduced up to t, $\beta_t(r)$ denotes the behavioral commitment expressed with respect to r, and $\epsilon_t^c(r)$ denotes the corresponding trace evidence. The behavioral status is inferred from the next relevant agent action as $\beta_t(r) = f_\beta(a_{t+1}, r)$, where f_β determines whether that action indicates that the agent treats r as resolved. The evidence status is further decomposed by provenance as $\epsilon_t(r) = \epsilon_t^{\text{int}}(r), \epsilon_t^{\text{ext}}(r)$, with each channel independently taking values in $\mathcal{S}$. The internal channel captures evidence from the agent's surfaced information (e.g., reflections and tool inputs), whereas the external channel captures observations returned by the environment (e.g., tool outputs and execution feedback); hence, the two channels may disagree for the same r and t. Notably, all `satisfied` statuses are trace-grounded by design. For each $r \in \mathcal{R}_t$, the behavioral commitment $\beta(r)$ is grounded in the action a_{t+1} (or goal G) that expresses it, while each evidential component $\epsilon^c(r)$ is grounded in at least one observation $o_{t,k}$ from provenance channel c whose content entails r. If no admissible grounding exists, the component remains `open`. Thus, every positive assertion in Σ_t is grounded in explicit trace evidence and directly auditable against the execution trace.

So far, Σ_t captures the reconstructed execution state at a single time point. To model how the trajectory evolves, we record changes between consecutive states. As the agent proceeds through the action–observation sequence, Σ_t evolves with changes in requirements, behavioral commitments, and available evidence. We represent each transition as $\delta_t = \Delta(\Sigma_t, \Sigma_{t+1})$, capturing changes in requirement status across β and ϵ, including requirements becoming satisfied, reopening, or being newly introduced.

Behavior-Evidence Reconciliation. For each requirement $r \in \mathcal{R}_t$, we derive a grounding label from the joint status of its behavioral commitment and internal and external evidence:

$$\gamma_t(r) \;=\; f\big(\beta_t(r),\, \epsilon_t^{\text{int}}(r),\, \epsilon_t^{\text{ext}}(r)\big)\,. \tag{2}$$

The label indicates whether the agent's behavioral commitment is consistent with the evidence available in the trace and, when inconsistent, characterizes the discrepancy. Table 1 reports the complete 2^3 configuration space, organized into *concordant* configurations, where the commitment is consistent with available evidence, and *mismatches*, where it diverges from the evidence. Concordant cases include requirements supported by internal and/or external evidence and the *genuine gap*, where the requirement remains unresolved without supporting evidence. The four mismatch categories constitute the core diagnostic outputs of γ. Aggregating γ_t over all requirements yields a

Table 1: Behavior–evidence reconciliation for a requirement r at step t. β, ϵ^{int}, and ϵ^{ext} indicate whether the requirement is behaviorally committed, supported by internal evidence, and supported by external evidence, respectively.

β	ϵ^{int}	ϵ^{ext}	**Diagnosis Label**	**Meaning**
Concordant: behavioral commitment is consistent with available evidence				
✓	✓	✓	*Fully Grounded*	r_t resolved with internal and external evidence.
✓	✓	–	*Self-Grounded*	r_t resolved from internal evidence only.
✓	–	✓	*Externally Grounded*	r_t resolved from external evidence only.
–	–	–	*Genuine Gap*	r_t unresolved and unsupported by available evidence.
Mismatch: behavioral commitment diverges from available evidence				
✓	–	–	*Ungrounded*	r_t resolved without supporting trace evidence.
–	✓	–	*Missed Internal Evidence*	Internal evidence is available but not reflected in the behavior.
–	–	✓	*Missed External Evidence*	External evidence is available but not reflected in the behavior.
–	✓	✓	*Missed Recognition*	Both evidence sources available but unreflected in behavior.

per-step grounding report:

$$\Gamma_t \;=\; \big\{\, \big(r, \gamma_t(r)\big) \;:\; r \in \mathcal{R}_t,\; \gamma_t(r) \in \mathcal{L} \,\big\}, \tag{3}$$

where $\mathcal{L}$ is the label set of Table 1. By construction $\Gamma_t = \emptyset$ when every requirement is warranted, so Γ_t acts as the step-level record of behavior-evidence discrepancies.

Structured report. Putting these components together yields the structured report $\mathcal{X}(\tau)$, aggregating the requirement-level states, transitions, and grounding reports across the full trajectory:

$$\mathcal{X}(\tau) \;=\; \Big(\, G,\; \big\{\big(a_t,\, \mathcal{O}_t,\, \Sigma_t,\, \delta_t,\, \Gamma_t\big)\big\}_{t=0}^{T} \Big), \tag{4}$$

where $\mathcal{O}_t = \{o_{t,k}\}_{k=1}^{K_t}$ denotes the set of observations associated with step t.

For each step, the structured report records the action taken, observations received, reconstructed knowledge–action state, transition summary, and grounding report. $\mathcal{X}(\tau)$ is directly inspectable and serves as the auditable input to the natural-language explanation described next.

XAI Narratives. As a downstream use of $\mathcal{X}(\tau)$, a constrained narrator ψ generates step-level explanations and an episode summary:

$$\psi : \mathcal{X} \to (E_1, \ldots, E_T, \bar{E}), \qquad E_t(\tau) = \psi_{\text{step}}(a_t, \mathcal{O}_t, \Sigma_t, \delta_t, \Gamma_t). \tag{5}$$

Each $E_t(\tau)$ verbalizes the epistemic change and any divergence recorded in Γ_t, while $\bar{E}(\tau)$ summarizes the trajectory at the episode level.

Operational Pipeline. The structured representation is constructed from the raw execution trace through a five-stage pipeline reported in Figure 1: trace normalization, requirement decomposition, local behavior–evidence extraction, and deterministic validation and reconciliation. These stages respectively produce the canonical trace τ, requirement registry $\mathcal{R}$, and grounded state and transition variables $(\Sigma_t, \delta_t, \Gamma_t)$ that constitute $\mathcal{X}(\tau)$. Narrative generation then operates on this audited representation through the constrained narrator ψ. Full implementation details and validation procedures are described in Appendix A.

4 Experiments

We evaluate the framework through a controlled study examining the reliability and trace-grounded fidelity of the structured report and the quality of the natural-language explanations. This section describes the experimental setup, evaluation dimensions, and protocols (see Figure 3 in the Appendix).

4.1 Experimental Setup

The evaluation is designed along two complementary dimensions. First, we assess the contribution of the proposed structured analysis by comparing our XAI artifacts with outputs generated directly

by an LLM under matched input and model conditions. Second, we evaluate the quality of the resulting artifacts using both human assessment and LLM-as-a-judge evaluation.

Trace Corpus. We focus our empirical evaluation on tool-empowered QA tasks, which provide well-defined goals and explicit action–observation sequences. This enables systematic assessment of requirement satisfaction and explanation faithfulness to observable agent behavior, while reducing reliance on subjective judgments and annotation costs. The choice of QA is specific to our empirical evaluation, not a limitation of the framework, which extends to other agentic tasks given suitable execution traces.

Among existing QA resources, TRAIL (Deshpande et al., 2025) is, to our knowledge, the only dataset providing publicly available execution traces. We therefore use TRAIL as an external trace source and build our evaluation around its underlying GAIA (Mialon et al., 2024) questions. Since TRAIL's agent and environment are unavailable and its trajectories lack explicit tool calls, we complement it with two independently implemented agents. The first is a *ReAct* agent (Yao et al., 2022), which interleaves reasoning and tool use within a single execution loop; the second is a *plan-and-act* agent that follows an iterative plan–act–reflect loop with explicit action routing. Both use *gemini-3.1-flash-lite*[2] and share a heterogeneous tool set comprising Wikipedia, Tavily web search, calculator, Python execution, PDF reader, and text editor, covering retrieval, browsing, computation, programmatic analysis, and document processing. For direct comparison with TRAIL, we use the 116 GAIA questions for which executions are available and run both agents on the same queries. We also consider 100 HotpotQA (Yang et al., 2018) questions sampled from the distractor validation split, providing a complementary multi-hop setting to further assess generalization. The resulting corpus comprises five benchmark–agent subsets for a total of 548 execution traces.

Experimental Configurations. We consider two complementary comparisons to isolate the contributions of the structured representation and its trace-grounded construction. We use paired notation (S, N), where S and N denote the sources of the structured report and narrative, respectively. The OUR–OUR configuration is the complete proposed pipeline, combining our trace-grounded construction with the constrained narrator ψ. The LLM–OUR configuration replaces only the structured-report construction: the LLM receives the same canonical trace representation, requirement vocabulary, and target schema, and is prompted to generate the report directly, which is then passed unchanged to ψ. This isolates whether the benefits of the proposed representation depend on its trace-grounded construction rather than on the schema alone. Finally, LLM-ONLY removes the intermediate representation and prompts the LLM to generate the narrative directly from the original trace, providing an end-to-end baseline aligned with existing approaches to LLM-based XAI narrative generation (Silvestri et al., 2025)[3]. All configurations use *gemini-3.1-flash-lite* for LLM-based generation and operate on the same input trace. Each artifact is generated with up to 20 attempts; unsuccessful generations are excluded from artifact-quality evaluation. Our pipeline produced no generation failures, whereas both LLM-OUR and LLM-ONLY exhibited failures, concentrated in longer GAIA traces. For LLM-ONLY, failures may additionally result from context-window limitations of the free Gemini plan. A complete breakdown is provided in Appendix B.1. Overall, our experimental setup yields 1,096 structured reports and 1,644 natural-language explanations across all configurations.

4.2 EVALUATION DIMENSIONS AND METRICS

We evaluate the framework at two complementary levels: the fidelity of the intermediate structured representation $\mathcal{X}(\tau)$ and the quality of the generated narratives $\bar{E}(\tau)$.

Structured Report Fidelity. We assess the trace-grounded faithfulness of $\mathcal{X}(\tau)$ using an ad hoc metric, *Normalized Faithfulness of the Intermediate Representation* (NFI). For each step t, let R_t^{GT} denote the set of ground-truth requirements and R_t^{gen} the corresponding requirements encoded in the generated report. We define

$$\mathrm{NFI} = \frac{\sum_t \mathrm{matched}\big(R_t^{gen}, R_t^{GT}\big)}{\sum_t |R_t^{GT}|}, \tag{6}$$

[2] https://deepmind.google/models/model-cards/gemini-3-1-flash-lite/

[3] Prompts are reported in Appendix E.

Figure 2: NFI (mean $\pm$ SD) by agent (color) and number of steps (x-axis); line style marks the method (solid: OUR, dashed: LLM).

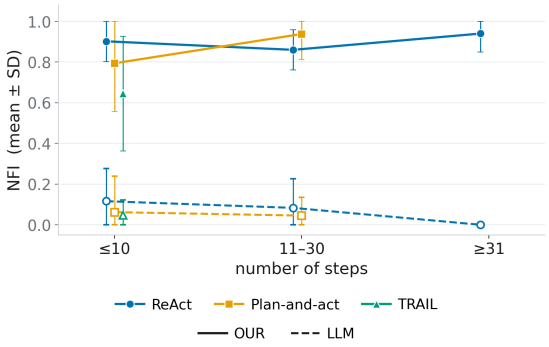

Table 2: Hyp-D direction consistency for the LLM-as-Judge corpus. Cell = mean paired difference within each agent architecture.

Hyp	Measure	ReAct	Plan-act	TRAIL
B	M2	+0.08	+0.03	+0.15
B	M3	+1.18	+1.11	+1.37
B	M4	+0.63	+0.55	+0.77
C	M2	+0.45	+0.59	+0.39
C	M3	-0.01	-0.01	+0.03
C	M4	+0.76	+0.93	+0.73

where $\mathrm{matched}(\cdot,\cdot)$ counts the ground-truth requirements correctly represented in the generated report. NFI therefore measures the proportion of trace-grounded requirements recovered by the intermediate representation, with scores in $[0, 1]$. We adopt a strict matching criterion: any step-level discrepancy is considered an error, including omitted requirements, missed δ_t or γ_t, incorrect classifications of β_t or ϵ_t, and incorrect assignments of supporting evidence.

Narrative quality. Following established practices in narrative XAI (Silvestri et al., 2025), we evaluate generated narratives along four dimensions[4]. *Faithfulness* (M1) is assessed at two granularities. *Narrative-to-report faithfulness* (M1-a) measures whether the narrative accurately reflects its source structured report, without unsupported claims, critical omissions, or semantic alterations. This criterion applies to the OUR-OUR and LLM-OUR configurations, for which the source report is available. *Trace-level faithfulness* (M1-b) instead evaluates whether the narrative accurately captures the agent's behavior in the raw execution trace. The remaining dimensions capture complementary aspects of explanatory quality, namely *exhaustiveness* (M2) measures whether the narrative conveys the information necessary to understand the agent's behavior; *linguistic quality* (M3) assesses clarity, coherence, and conciseness; and *overall quality* (M4) provides a holistic assessment of the narrative's explanatory utility.

Diagnostic Alignment. Although not primarily designed for failure detection, we assess whether behavioral–evidence mismatches identified by the grounding report correspond to independently annotated failures in the TRAIL benchmark, highlighting a potential secondary application beyond the framework's primary auditability objectives.

4.3 EVALUATION PROTOCOLS

Human Evaluation. Human evaluation is conducted on a stratified subset of 32 raw traces across the three agent systems and two benchmarks, stratified by task difficulty and number of tool interactions (see Appendix B.2). For each trace, a human annotator independently constructs the structured representation following the schema in Section 3.1, providing an independent reference for both structured-report variants. Excluding failed artifact-generation cases, the evaluation covers 63 structured reports and 89 narratives across the two structured-output methods and three narrative-generation settings. Evaluators assess all the dimensions described in Section 4.2.

LLM-as-a-Judge Evaluation. For scalable evaluation, we employ an LLM-as-a-Judge protocol (Zheng et al., 2023), using *Muse-Spark-1.2-Contributor* to evaluate M2, M3, M4. Because reliably assessing trace-level faithfulness requires the direct inspection of lengthy execution traces, we exclude NFI and M1-a,b from automated evaluation and assess them exclusively via human review. To ensure the reliability of LLM-judge evaluations, we first measure their alignment with human ratings on the common subset. Detailed metric-level results and agreement statistics are reported in Table 6 (Appendix B.4). Raw agreement is highest for narrative-to-report faithfulness

[4]Full metric definitions and the evaluation rubric are provided in Appendix B.3.

Table 3: Hypothesis tests on the LLM-as-judge corpus, with human effects for reference. Each row is a one-sided paired contrast on the same traces (Hyp-B: OUR $>$ LLM baseline; Hyp-C: OUR $>$ LLM-structured). Measures: M1-a (narrative vs. its own report), M1-b (vs. the trace), M2 (content), M3 (delivery), M4 (overall; primary, **bold**). n: paired blocks; $W/T/L$: wins/ties/losses; Δ: mean paired difference with cluster-bootstrap 95% CI over queries; p: exact one-sided query-level permutation with joint sign flips (p_{BH}: Benjamini–Hochberg within hypothesis). M1-a and M1-b are assessed exclusively by human evaluators.

		Human		LLM judge						
Hyp	Measure	n	Δ [95% CI]	n	W	T	L	Δ [95% CI]	p	p_{BH}
B	M1-b	26	1.11 [0.79, 1.39]	–	–	–	–	–	–	–
B	M2	26	0.85 [0.62, 1.07]	417	18	395	4	0.07 [0.02, 0.12]	0.0031	0.0031
B	M3	26	1.04 [0.65, 1.36]	418	353	65	0	1.18 [1.12, 1.25]	0.0001	0.0001
B	**M4**	26	**1.35 [1.00, 1.64]**	416	206	209	1	**0.62 [0.55, 0.69]**	0.0001	0.0001
C	M1-a	31	0.55 [0.32, 0.79]	–	–	–	–	–	–	–
C	M1-b	31	0.52 [0.28, 0.71]	–	–	–	–	–	–	–
C	M2	31	1.03 [0.61, 1.45]	500	124	373	3	0.49 [0.41, 0.58]	0.0001	0.0001
C	M3	31	0.39 [-0.03, 0.82]	500	30	438	32	-0.00 [-0.03, 0.03]	0.6418	0.8022
C	**M4**	31	**1.00 [0.58, 1.42]**	497	186	305	6	**0.82 [0.72, 0.93]**	0.0001	0.0001

(94%), followed by linguistic quality (84%) and exhaustiveness (79%), and is lowest for overall quality (67%). Automatic evaluation was also used to assess the correspondence between grounding mismatches and TRAIL failure annotations[5].

5 Main Results

We evaluate four directional hypotheses concerning structured-representation fidelity (Hyp-A), end-to-end performance (Hyp-B), the contribution of the structured representation (Hyp-C), and robustness across agent architectures (Hyp-D). In addition, we report a diagnostic analysis examining whether behavioral–evidence mismatches identified by the grounding report align with independently annotated failures in the TRAIL benchmark. Full methodological details are provided in Appendix B.5, while more extensive human-evaluation results are reported in Appendix C.1.

5.1 Structured-Representation Fidelity (Hyp-A)

Hypothesis. Our structured report is more faithful and complete as a step-by-step representation of the execution trace than the freely LLM-generated report.

Results. Figure 2 reports NFI for the two structured-report variants. Our structured reports substantially outperform those generated by directly prompting the LLM: across 63 reports, our ($\mathrm{SD} = 0.124$) for the LLM baseline. The advantage is consistent across all three agent architectures and shows no evident degradation with trace length, whereas the LLM-generated reports decline as the number of steps increases: under the strict matching criterion 74% of them score zero, versus none of ours.

5.2 End-to-end superiority (Hyp-B)

Hypothesis. Narratives produced by the full pipeline, Our-Our, receive higher ratings than narratives generated by directly prompting an LLM to explain the execution trace, LLM-only, across all narrative-quality criteria.

Results. Table 3 shows positive effects across all metrics in the LLM-as-a-Judge evaluation. The effect is smallest for M2 ($\Delta = 0.07$) but remains statistically significant, while substantially larger

[5]See Appendix C.2 for details on judge calibration and disaggregated results.

effects are observed for M3 ($\Delta = 1.18$) and M4 ($\Delta = 0.62$), both with $p < 0.0001$. Human-evaluation estimates show the same ordering for M3 and M4, with larger effect magnitudes. For faithfulness, the human evaluation shows definitely positive effect on trace-level faithfulness (M1-b), with $\Delta = 1.11$. Therefore, experimental results fully support our hypothesis.

5.3 Value of the Structured Representation (Hyp-C)

Hypothesis. Holding the narrator fixed, Our-Our outperforms LLM-Our across M1-a, M1-b, M2, M3, and M4.

Results. This hypothesis isolates the contribution of the structured representation by comparing configurations that differ in how the structured representation is obtained. The results in Table 3 confirm the hypothesis across all relevant metrics. In the LLM-as-a-Judge evaluation, Our-Our significantly outperforms LLM-Our on M2 ($\Delta = 0.49$, $p < 0.0001$) and M4 ($\Delta = 0.82$, $p < 0.0001$), while M3 shows essentially no difference, consistent with the fact that both configurations share the same narrator. In the human evaluation, Our-Our also shows positive effects on faithfulness at both granularities, with $\Delta = 0.55$ for narrative-to-report faithfulness (M1-a) and $\Delta = 0.52$ for trace-level faithfulness (M1-b). Positive effects are likewise observed for M2 ($\Delta = 1.03$) and M4 ($\Delta = 1.00$), with slightly larger effect sizes than those obtained by the LLM evaluation.

5.4 Invariance across agent architectures (Hyp-D)

Hypothesis. The direction of the Hyp-B and Hyp-C effects is preserved across the three source agent architectures, such that the observed advantages of Our-Our over LLM-Only and Our-Our over LLM-Our do not depend on the architecture generating the execution trace.

Results. This hypothesis examines whether the effects observed under Hyp-B and Hyp-C are consistent across agent architectures. Table 2 reports the mean paired differences separately for ReAct, Plan-and-act, and TRAIL agent. Under Hyp-B, all three measures show a positive effect across all architectures, yielding consistent directions. The same pattern holds under Hyp-C for M2 and M4, with positive effects across agent configurations. In contrast, M3 remains effectively null under Hyp-C across all three architectures ($\Delta = -0.01$, -0.01, and $+0.03$, respectively). This behavior is expected given that the two configurations compared under Hyp-C share the same narrator across architectures, while M3 evaluates linguistic realization produced by that narrator. Thus, changing the underlying agent architecture does not introduce a systematic M3 difference between the two configurations.

5.5 Failure Detection

Across the 569 error instances identified by TRAIL annotators, our method detected 433, corresponding to an overall recall of 76.1%. Detection is particularly high for errors that are most relevant to our scope, including tool selection (97.6%), tool-related errors (95.5%), task orchestration (93.2%), and goal deviation (90.3%). Detection is lower for formatting errors (55.7%), instruction non-compliance (38.5%), and resource- or environment-related errors, which often require greater contextual or interpretive judgment and are therefore less directly identifiable from the trace alone. The full disaggregated results are reported in Appendix C.2, Table 10.

6 Conclusion

We presented a post-hoc framework for explaining AI agents from execution traces. By transforming heterogeneous traces into an auditable representation that separates behavioral commitments from supporting evidence, the framework enables step-level diagnostics and trace-grounded explanations. Across three agent architectures and two QA benchmarks, structured reports were more faithful to human references than direct LLM generation, while their resulting narratives achieved higher faithfulness, exhaustiveness, and overall quality than direct trace-to-text explanations. Future work will extend evaluation to broader agentic tasks and more complex execution scenarios.

AI USE STATEMENT

In this work, we used generative AI tools to edit the paper's text to improve English-language readability and clarity. We also sporadically used generative AI tools to support coding activities and to assist in generating Figures 1 and 3. We have not used generative AI tools for any other tasks requiring disclosure. All AI-assisted text and code were reviewed and revised by the authors. We take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our experiments throughout the main paper and Appendix, including the experimental setup, model and framework specifications, datasets and preprocessing procedures, evaluation protocols, and implementation details. The Appendix further provides the prompts and configuration details used in our experiments. In addition, we release a documented codebase as supplementary material, including the code required to reproduce the reported results together with detailed instructions for installation, execution, and reproduction of the experimental pipeline.

REFERENCES

Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. *IEEE Access*, 13:18912–18936, 2025.

Sindhuja Chaduvula, Jessee Ho, Kina Kim, Aravind Narayanan, Mahshid Alinoori, Muskan Garg, Dhanesh Ramachandram, and Shaina Raza. From features to actions: Explainability in traditional and agentic ai systems. *arXiv preprint arXiv:2602.06841*, 2026.

Pradipta Kishore Chakrabarty. Causal inference in agentic ai: Bridging explainability and dynamic decision making. *International Journal of Science and Research (IJSR)*, 14(4):10–21275, 2025.

Sangwoo Cho, Kushal Chawla, Pengshan Cai, Zefang Liu, Chenyang Zhu, Shi-Xiong Zhang, and Sambit Sahu. Ask, don't judge: Binary questions for interpretable LLM evaluation and self-improvement. In *2nd Workshop on Compositional Learning: Safety, Interpretability, and Agents*, 2026. URL https://openreview.net/forum?id=tJytD1JIkR.

Bernardo G Collaco, Srinivasagam Prabha, Cesar A Gomez-Cabello, Syed Ali Haider, Ariana Genovese, Nadia G Wood, Narayanan Gopala, Raghunath Raman, Erik O Hester, and Antonio Jorge Forte. Explainable agentic artificial intelligence in healthcare: A scoping review. *Bioengineering*, 13(5):513, 2026.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

Upol Ehsan, Amal Alabdulkarim, Kenneth Holstein, Min Kyung Lee, Andreas Riener, and Justin D Weisz. Human-centered explainable ai (hcxai): Re-examining xai in the era of agentic ai. In *Proceedings of the Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems*, pp. 1–6, 2026.

Yifan He and David Martens. An agentic approach to generating xai-narratives. *arXiv preprint arXiv:2603.20003*, 2026.

Jinwei Hu, Xinmiao Huang, Qisong He, Youcheng Sun, Yi Dong, and Xiaowei Huang. Responsible agentic ai requires explicit provenance. *arXiv preprint arXiv:2605.17169*, 2026.

Omer Khalid, Ammad Ul Haq Farooqi, and Muhammad Bilal. Agentic ai: a review, applications and open research challenges. *Computing*, 108(6):80, 2026.

Anna Mazhar, Huzaifa Suri, and Sainyam Galhotra. Trace-level analysis of information contamination in multi-agent systems. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 483–496, 2026.

Sushant Mehta. Matrag: Multi-agent transparent retrieval-augmented generation for explainable recommendations. *arXiv preprint arXiv:2604.20848*, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Hadar Mulian, Sergey Zeltyn, Ido Levy, Liane Galanti, Avi Yaeli, and Segev Shlomov. Agentfixer: From failure detection to fix recommendations in agentic systems. In *Proceedings of the 2026 International Workshop on Agentic Engineering*, pp. 96–103, 2026.

OpenAI. The hugging face incident and the road ahead. https://openai.com/index/hugging-face-incident-and-the-road-ahead/, August 2026. Accessed: 2026-09-18.

Clemens Rawert, Marc Klingen, and Maximilian Deichmann. Langfuse—open-source llm engineering platform. *Software available from https://langfuse. com*, 2023.

Shaina Raza, Ahmed Radwan, Sindhuja Chaduvula, Mahshid Alinoori, and Christos Emmanouilidis. Transparency in agentic ai: A survey of interpretability, explainability, and governance. 2026.

Mehil B Shah, Mohammad Mehdi Morovati, Mohammad Masudur Rahman, and Foutse Khomh. Characterizing faults in agentic ai: A taxonomy of types, symptoms, and root causes. *arXiv preprint arXiv:2603.06847*, 2026.

Reshabh K Sharma, Shraddha Barke, and Benjamin Zorn. Willful disobedience: Automatically detecting failures in agentic traces. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 584–596, 2026.

Matteo Silvestri, Vittoria Vineis, Edoardo Gabrielli, Flavio Giorgi, Fabiano Veglianti, Fabrizio Silvestri, and Gabriele Tolomei. A survey on explainable ai narratives based on large language models. 2025.

Vittoria Vineis, Matteo Silvestri, Lorenzo Antonelli, Filippo Betello, and Gabriele Tolomei. Ponte: Personalized orchestration for natural language trustworthy explanations. *arXiv preprint arXiv:2603.06485*, 2026.

Zhaohui Geoffrey Wang. Agenttrace: Causal graph tracing for root cause analysis in deployed multi-agent systems. *arXiv preprint arXiv:2603.14688*, 2026.

Shu-Xun Yang, Cunxiang Wang, Haoke Zhang, Wenbo Yu, Lindong Wu, Jiayi Gui, Dayong Yang, Yukuo Cen, Zhuoer Feng, Bosi Wen, et al. Tracesir: A multi-agent framework for structured analysis and reporting of agentic execution traces. *arXiv preprint arXiv:2603.00623*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=tvI4u1ylcqs.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Judy Zhu, Dhari Gandhi, Himanshu Joshi, Ahmad Rezaie Mianroodi, Sedef Akinli Kocak, and Dhanesh Ramachandran. Interpreting agentic systems: Beyond model explanations to system-level accountability. *arXiv preprint arXiv:2601.17168*, 2026.

A Detailed Operational Pipeline

Following the definition of the structured representation in Section 3.1, we now describe how it is constructed from a raw execution trace. These stages correspond to the transformation block on the left of Figure 1: normalization produces the canonical trace τ, requirement decomposition produces $\mathcal{R}$, local extraction proposes the components of Σ_t, and validation and reconciliation produce the final δ_t and γ_t states.

Trace Normalization. The raw execution trace is mapped to the canonical step sequence required by the framework. Each step preserves the action, its observations, their semantic types, and the terminal condition. The representation uses a fixed observation vocabulary—user query, tool input, tool output, reflection, error, and final response—and separates tool inputs from tool outputs so that agent-emitted information remains distinguishable from environment-returned information. The resulting canonical sequence is the common interface to all subsequent stages, decoupling the analysis from the underlying agent architecture and enabling the same pipeline to be applied across heterogeneous paradigms.

Requirement Decomposition. The normalized goal is decomposed into the requirement registry $\mathcal{R}$ using a summarize-then-decompose strategy inspired by Cho et al. (2026). The decomposition targets atomicity at the level required for independent status assessment: requirements concerning different entities are kept distinct whenever their satisfaction can be established independently. Each requirement is assigned a stable identifier and classified as either a *content requirement* (information that must appear in the answer) or a *constraint requirement* (formatting or structural condition). Depending on the application, requirements may be generated upfront, incrementally, or through a hybrid strategy that admits newly discovered subgoals during execution.

Local Behavior–Evidence Extraction. At each step, an LLM produces local updates to the requirement-level state. Given the information available at step t together with the subsequent action a_{t+1}, the extractor proposes behavioral commitments and evidence items, each accompanied by pointers to the specific trace elements that support them. We recall that behavioral commitment is inferred from a_{t+1}; evidence proposals are assigned to the internal or external channel according to the provenance of their cited observations.

Validation and Grounding. Candidate updates are subjected to deterministic validation. Behavioral supports must correspond to the action designated as the behavioral basis, and evidence supports must reference observations present in the trace and admissible for their assigned channel; ill-typed or unsupported references are discarded. When entailment verification is enabled, an evidence channel is marked `satisfied` only if the cited content additionally entails the corresponding requirement. Validated updates are then incorporated into the state subject to the framework's consistency and monotonicity constraints.

Deterministic Reconciliation. The validated state sequence is processed deterministically to obtain the per-step summaries. Consecutive states yield the transition δ_t, and the joint status of behavioral commitment and the two evidence channels determines the label $\gamma_t(r)$ for every requirement, which aggregate into the step-level report Γ_t. The resulting $(\Sigma_t, \delta_t, \Gamma_t)$ tuples complete the structured report $\mathcal{X}(\tau)$.

Narrative Generation. The narrator ψ is instantiated as a constrained LLM call over $\mathcal{X}(\tau)$. Because every belief, evidence item, and diagnostic has already been grounded in the trace, narrative generation reduces to verbalizing an audited execution history rather than reconstructing reasoning from raw logs. The narrator is constrained to reference only elements of its input—requirement identifiers, action identifiers a_t, observation identifiers $o_{t,k}$, and their statuses—so that every generated statement remains trace-grounded and no content is introduced beyond $\mathcal{X}(\tau)$. The resulting explanation reports what the agent attempted, what information it acquired, which requirements became satisfied, and where potential failures or inconsistencies occurred.

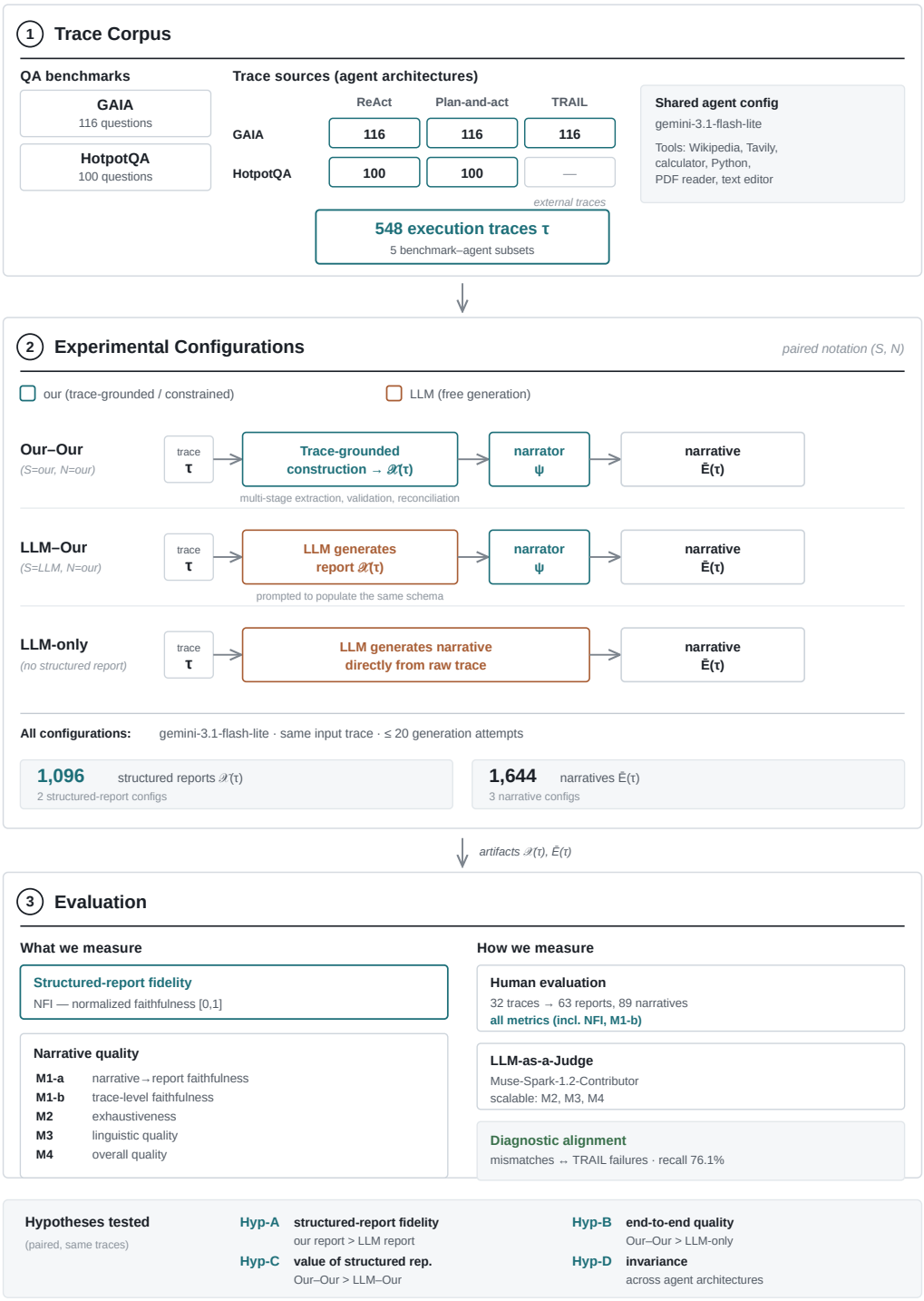

Figure 3: Overview of the experimental setup and evaluation.

Table 4: Evaluation corpus and pipeline reliability. For each of the five (benchmark × agent) subsets we report the number of traces processed and the number of *failed* generations per pipeline configuration (percentages relative to the row total). The *Structured report* block covers the intermediate representation ($\mathcal{X}(\tau)$), produced either by our deterministic pipeline (OUR) or by a single-shot LLM prompt with the same schema (LLM). The *Narrative* block covers the final natural-language explanation ($\bar{E}$), produced by: (i) OUR–OUR, our narrator applied to our structured report; (ii) LLM–OUR, our narrator applied to a directly LLM-generated version of the structured report; and (iii) LLM-ONLY, an LLM prompted to summarize the raw trace directly in natural language.

			Struct. report — fails (%)		Narrative — fails (%)		
Benchmark	Agent	# traces	OUR	LLM	OUR–OUR	LLM–OUR	LLM-ONLY
GAIA	Plan-and-act	116	0	1 (0.9)	0	—	25 (21.6)
GAIA	ReAct	116	0	9 (7.8)	0	—	40 (34.5)
GAIA	TRAIL	116	0	3 (2.6)	0	—	28 (24.1)
HotpotQA	Plan-and-act	100	0	1 (1.0)	0	—	5 (5.0)
HotpotQA	ReAct	100	0	0	0	—	1 (1.0)
Total		**548**	**0**	**14 (2.6)**	**0**	—	**99 (18.1)**

B ADDITIONAL DETAILS ON THE EXPERIMENTAL SETUP

B.1 GENERATION FAILURES

Table 4 reports the generation failures observed across the evaluation corpus and pipeline configurations. For each benchmark–agent subset, we report failures in structured-report and narrative generation, together with the corresponding percentages relative to the number of traces in the subset. A generation is counted as failed when the requested artifact is not successfully produced within the allowed 20 attempts.

B.2 HUMAN EVALUATION SUBSET AND REFERENCE ANNOTATION.

We conduct human evaluation on a stratified subset of the execution corpus, comprising 32 traces from 12 queries across three agent systems (ReAct, plan-and-act, and TRAIL agent). The ReAct and plan-and-act agents contribute 12 traces each, covering 9 GAIA and 3 HotpotQA questions, while TRAIL contributes 8 GAIA traces[6]. GAIA questions are selected through stratified random sampling over difficulty and number of tool interactions, following the characterization provided by the original GAIA benchmark paper. For HotpotQA, whose original difficulty characterization is based on query length and is not meaningful to our setting, we assign each query a difficulty level and tool-use stratum using the same criteria adopted for GAIA.

For each selected trace, a human annotator independently constructs a structured representation directly from the original execution trace, following the representation schema defined in Section 3. This human-annotated representation serves as the reference for evaluating the two structured-report variants: the report produced by our full pipeline and the report obtained by prompting the LLM to instantiate the same schema. Because the reference is constructed directly from the trace and independently of both generated reports, it provides an independent basis for assessing the fidelity of the intermediate representation. Considering the two structured-output methods (OUR and LLM) and the three narrative-generation settings (OUR-OUR, LLM-OUR, and LLM-ONLY), and excluding cases in which artifact generation failed (see Table 4), the human evaluation covers 63 structured reports and 89 narratives.

[6]One sampled GAIA question has no corresponding TRAIL execution and is therefore excluded from the TRAIL subset.

Dimension	Strongly agree	Agree	Disagree	Strongly disagree
M1 – Faithfulness	The narrative accurately reflects the structured output, with all relevant claims and interpretations fully supported by the available information.	The narrative is generally faithful to the structured output, with only minor inaccuracies, omissions, or unsupported claims.	The narrative contains noticeable inaccuracies, unsupported claims, or misrepresentations of the structured output.	The narrative substantially misrepresents the structured output or contains many unsupported claims.
M2 – Content / Exhaustiveness	The narrative contains all essential information needed to fully understand the model's behavior, without relevant omissions.	The narrative contains most of the essential information needed to understand the model's behavior, with only minor gaps.	The narrative omits important information needed to understand the model's behavior, resulting in noticeable gaps.	The narrative lacks substantial essential information and does not enable understanding of the model's behavior.
M3 – Delivery / Linguistic Quality	The narrative is clear, logically structured, concise, and easy to understand, with effective communication throughout.	The narrative is generally clear and understandable, with only minor issues in organization, conciseness, or language quality.	The narrative has noticeable issues with clarity, structure, conciseness, or readability that hinder understanding.	The narrative is unclear, poorly structured, overly verbose, or difficult to understand.
M4 – Holistic / Overall Satisfaction	Overall, the narrative provides a complete, appropriate, and satisfactory standalone explanation of the model's behavior.	Overall, the narrative is a useful explanation of the model's behavior, with only minor shortcomings.	Overall, the narrative provides only a partially satisfactory explanation, with significant limitations.	Overall, the narrative does not provide a satisfactory explanation of the model's behavior.

Table 5: Evaluation rubric used for human and LLM-as-a-Judge assessment of narrative explanations.

B.3 Evaluation Rubric

The following rubric operationalizes the four narrative evaluation dimensions introduced in Section 4.2. For each dimension, evaluators assign an ordinal rating based on the level of agreement with the corresponding statement.

B.4 Human-LLM evaluator agreement analysis.

Before using LLM-as-a-judge for scalable evaluation, we assess its alignment with human judgments, on the common scored subset. For each criterion rated by both evaluators, we report raw percentage agreement after collapsing the ordinal scale into positive and negative judgments, together with quadratic-weighted Cohen's κ (κ_w), which preserves the ordinal structure of the ratings and penalizes larger disagreements more strongly than near-misses.

Both measures are interpreted in conjunction with the marginal distribution of the ratings. In particular, when narratives from stronger configurations receive positive judgments almost uniformly, the negative class becomes small and chance-corrected agreement can be attenuated: high raw agreement may therefore coexist with a low κ_w when little disagreement remains for chance correction

Table 6: Agreement between the human annotator and the LLM judge, pooled over conditions (bold) and split by condition. H/J = percentage of ratings on the agree side for the human and judge, respectively; Agr. = raw agreement on the agree/disagree collapse; κ_w = quadratic-weighted Cohen's κ; Neg. rec. = share of human-negative ratings also rated negative by the judge, with counts in parentheses. † marks rows with fewer than three human-negative ratings, for which κ_w is not informative; raw agreement and negative recall should be preferred.

Measure	Condition (n)	H/J (%)	Agr. (%)	κ_w	Neg. rec.
M1-a	**All (63)**	**95/98**	**94**	**-0.09**	**0% (0/3)**
	Our–Our (32)	100/97	97	$-0.06^\dagger$	n/a
	LLM–Our (31)	90/100	90	-0.05	0% (0/3)
M2	**All (86)**	**71/87**	**79**	**0.28**	**36% (9/25)**
	Our–Our (32)	94/94	94	$0.22^\dagger$	50% (1/2)
	LLM–Our (31)	48/71	71	0.36	50% (8/16)
	LLM-only (23)	70/100	70	0.00	0% (0/7)
M3	**All (86)**	**81/86**	**84**	**0.52**	**44% (7/16)**
	Our–Our (32)	94/100	94	$0.27^\dagger$	0% (0/2)
	LLM–Our (31)	87/100	87	0.16	0% (0/4)
	LLM-only (23)	57/48	65	0.40	70% (7/10)
M4	**All (86)**	**62/80**	**67**	**0.41**	**33% (11/33)**
	Our–Our (32)	97/91	94	$0.13^\dagger$	100% (1/1)
	LLM–Our (31)	55/68	68	0.48	50% (7/14)
	LLM-only (23)	22/83	30	-0.06	17% (3/18)

to capture. We therefore additionally report, for each criterion, the proportion of positive ratings assigned by each evaluator and the judge's *negative recall*, defined as the proportion of cases rated negatively by humans that are also rated negatively by the judge. Negative recall is particularly relevant for scalable evaluation, as it measures whether the judge identifies the failures that human evaluators would flag, independently of a shared tendency to assign positive ratings. Statistics are pooled across the three narrative configurations. They are computed over the paired ratings available for each criterion; consequently, the effective sample size varies across criteria because of missing or inapplicable annotations.

Table 6 reports comprehensive agreement results disaggregated by evaluation dimensions. Raw agreement is highest for OUR-OUR (94–97%), intermediate for LLM-OUR (61–90%), and lower for LLM-ONLY (up to 70%). These values must be read together with the marginal distributions: where ratings are highly skewed—OUR-OUR has fewer than three human-negative ratings per criterion—high raw agreement carries little information and κ_w is correspondingly attenuated. For criteria with sufficient variation, agreement is substantial: pooled κ_w is 0.63 for cross-level faithfulness, 0.52 for linguistic quality, and 0.41 for overall quality. The main discrepancy sits in the calibration, since the judge is systematically more positive than humans, most visibly on the weaker configurations (e.g., for M4 the LLM-ONLY positive rate is 22% for humans but 83% for the judge), which compresses the differences between conditions, particularly for exhaustiveness. Consequently the judge recovers only $9/25$ human-negative M2 cases and $11/33$ M4 cases, so M2-based judge scores in particular cannot replace human assessment. This may reflect differing weightings of surface fluency, apparent completeness, and trace-grounded evidence between human and model-based evaluation.

B.5 STATISTICAL TESTING

For hypothesis tests, all comparisons are paired on the same execution traces. Because the human-evaluated subset comprises 12 queries, with each query executed by up to three agents, ratings associated with the same query are not treated as independent observations. For each directional comparison, we therefore obtain p-values from a one-sided paired permutation test in which the sign of the paired difference is flipped at the query level and applied jointly to all agents associated with that query. With 12 queries, the resulting null distribution contains 2^{12} possible sign assignments, which can be enumerated exhaustively to obtain exact p-values. Multiplicity is controlled using the Benjamini–Hochberg procedure across criteria within each hypothesis. M4 (overall satis-

Table 7: NFI: mean (std) and n by method and agent, overall and by number-of-steps group.

		Overall		≤ 10		11–30		≥ 31	
Method	Agent	n	Mean (SD)	n	Mean (SD)	n	Mean (SD)	n	Mean (SD)
OUR	ReAct	12	.904 (.092)	5	.902 (.099)	3	.860 (.098)	4	.940 (.090)
OUR	Plan-and-act	12	.842 (.212)	8	.794 (.237)	4	.938 (.125)	–	–
OUR	TRAIL	8	.644 (.282)	8	.644 (.282)	–	–	–	–
LLM	ReAct	11	.075 (.131)	5	.116 (.161)	3	.083 (.144)	3	.000 (.000)
LLM	Plan-and-act	12	.057 (.149)	8	.062 (.177)	4	.045 (.090)	–	–
LLM	TRAIL	8	.046 (.077)	8	.046 (.077)	–	–	–	–

faction) is designated as the primary confirmatory outcome, while the remaining criteria are treated as secondary. Each comparison is reported with the mean paired difference, a cluster-bootstrap confidence interval, and win, tie, and loss counts.

For analysis, ordinal categories are mapped to consecutive integers in ascending order (1–3 for faithfulness and 1–4 for M2–M4). Accordingly, Δ denotes the mean paired difference under this coding and should be interpreted as an average movement in rating categories rather than as an interval-scale quantity. The permutation tests do not depend on this coding: under the null, the sign-flip distribution is exact for any statistic computed from the paired differences. Similarly, win, tie, and loss counts are invariant under any order-preserving recoding of the categories.

Confidence intervals for Δ are obtained using a nonparametric cluster bootstrap with $B = 5{,}000$ replicates and a fixed seed. The query is the resampling unit: each replicate samples with replacement k queries, where k is the number of queries contributing to the contrast, and retains all query–agent blocks associated with each sampled query. Thus, the number of clusters per replicate is fixed at k, while the number of blocks is random, with expectation equal to the observed number of blocks n. The statistic is the unweighted mean over blocks in each replicate, matching the point estimate. Reported intervals are the 2.5th and 97.5th percentiles of the bootstrap distribution. The same procedure, with the same B and resampling unit, is used for the intervals accompanying the correlations.

Because the human subset contains only approximately 12 query-level clusters, percentile bootstrap intervals may under-cover. These intervals are therefore treated as indicative, and no inferential conclusions rest on them.

C Additional Results

C.1 Human-Evaluation Results

End-to-end superiority (Hyp-B). This hypothesis compares narratives produced by the full pipeline (Our-Our) with those generated by directly prompting an LLM to explain the execution trace (LLM-only). The full pipeline significantly outperforms the direct explainer on all four criteria, with $p_{\mathrm{BH}} = 0.0019$ throughout (Table 8). The mean paired differences are $\Delta = 1.35$ for overall quality, 1.11 for trace-level faithfulness, 1.04 for linguistic quality, and 0.85 for exhaustiveness. Across the four criteria and 26 paired traces, Our-Our receives higher ratings in 80 of 104 comparisons, ties in 23, and loses only once.

The distribution of ratings (Figure 4) further clarifies the difference. For exhaustiveness, LLM-only receives positive ratings in 73% of cases but only 8% reach the highest category. This apparent strength does not extend to trace-level faithfulness: 46% of LLM-only narratives receive an outright *Disagree*. Overall quality follows the faithfulness pattern: only 31% of LLM-only narratives are rated positively, versus 97% for Our-Our. Thus direct prompting can produce narratives perceived as reasonably complete and fluent, but these qualities do not translate into faithful explanations of the trace, consistent with the benefit of grounding generation in the structured, step-level representation.

Value of the Structured Representation (Hyp-C). Hyp-C holds the narrator fixed and varies only the structured report, isolating the contribution of the structuring stage. OUR-OUR significantly outperforms LLM-OUR on four of five criteria (Table 8): exhaustiveness ($\Delta = 1.03$), overall quality ($\Delta = 1.00$), faithfulness to the source report ($\Delta = 0.55$), and trace-level faithfulness ($\Delta = 0.52$). Linguistic quality does not significantly differ ($\Delta = 0.39$), consistent with both configurations sharing the same narrator. The effect is smaller and more frequently tied than in the end-to-end comparison, with 75 wins, 70 ties, and 10 losses across 155 paired comparisons.

The distribution of ratings (Figure 4) shows how the deficit in the LLM-generated report propagates: narratives built on it remain largely faithful to the report itself (90% positive M1-a) but exhibit substantially lower content quality (exhaustiveness positive in 48% vs. 94% for OUR-OUR; overall quality 55% vs. 97%), while linguistic quality stays similar (87% vs. 94%). The primary effect of the structuring stage is thus on the content available to the narrator rather than on linguistic realization. For M1-a, 17 of 31 paired comparisons are ties while OUR-OUR wins the remaining 14 and loses none: when the LLM report is adequate both configurations produce faithful narratives, and our pipeline's advantage emerges primarily when the intermediate representation becomes limiting.

Invariance Across Agent Architectures (Hyp-D). Table 9 reports the Hyp-B and Hyp-C contrasts separately for each agent architecture. All subgroup effects are positive: none of the 27 architecture-specific effects reverses direction or is zero, so Hyp-D is supported in terms of effect-direction consistency. The subgroups contain only 8–12 queries, insufficient for a reliable condition-by-architecture interaction test, so we report directions and magnitudes without p-values. The confirmatory outcome, overall quality, is particularly stable (Hyp-B: $+1.33$, $+1.33$, $+1.38$; Hyp-C: $+1.00$, $+1.08$, $+0.88$ across ReAct, Plan-and-act, and TRAIL). Effect magnitudes vary for individual criteria—Hyp-C faithfulness to the source report ranges from $+0.27$ on ReAct to $+1.12$ on TRAIL, and linguistic quality shows the smallest effects ($+0.64$, $+0.33$, $+0.12$). At the judge level, the same contrasts stay direction-consistent across architectures (Table 2), the one exception being Hyp-C on M3, which is ≈ 0 everywhere as expected under a fixed narrator.

Relations among criteria. Figure 5 shows strong associations between overall quality (M4) and trace-level faithfulness ($\rho = 0.85$), exhaustiveness ($\rho = 0.84$), and delivery ($\rho = 0.81$), suggesting that human holistic evaluation reflects substantive content and faithfulness rather than fluency alone. Faithfulness to the structured report (M1-a) is comparatively weakly associated with the narrative criteria ($\rho = 0.23$–0.40): a narrative may faithfully reflect an imperfect report, so M1-a alone is insufficient to characterize explanation quality. NFI shows moderate associations ($\rho = 0.33$–0.48), indicating that the structured representation constrains narrative quality without fully determining it.

C.2 FAILURE DETECTION ON TRAIL CORPUS

Failure detection is performed through LLM-as-a-Judge evaluation comparing TRAIL annotations over agentic traces for the GAIA dataset queries with the matching structured reports. The task ID is fetched from the structured report to find the original trace, from which the related TRAIL annotation is retrieved. For each TRAIL annotation in a file, the *Muse-Spark-1.2-Contributor* model takes as input the system prompt and the bodies of both the report and the annotation itself. The prompt leverages role-playing and prefix caching to reduce cache misses while retaining accuracy and detail. After illustrating the input structure and contents, it asks to find whether the error described in the annotation is present within the structured report and details the output format.

The output is a JSON file containing a boolean indicating if the error is present, the location ID of the error within the report, the error type according to the annotation, the requirement $r \in \mathcal{R}_t$ processed at the error location, and the corresponding label $\gamma_t(r)$. The TRAIL annotation and the report use different taxonomies and levels of abstraction, so they do not need to be exact matches. The annotated error is deemed present when it is visible in the report and can reasonably be associated with a $\gamma_t(r)$ finding that describes the same failure, a cause or consequence of it, or the requirement directly affected by it. Multiple runs were performed for validation, checking consistency among results and correctness with respect to the original trace from a sample of responses. As a consequence, the prompt has been iteratively improved and made stricter and better aligned with our judgment.

Table 8: Hypothesis tests on the human-evaluated subset (12 queries $\times$ 3 agents). n = paired blocks; W, T and L = wins, ties and losses for the first condition; Δ = mean paired difference with a cluster bootstrap CI over queries (indicative at 12 clusters); p = exact one-sided query-level permutation; p_{BH} = Benjamini–Hochberg corrected within hypothesis. Bold = pre-designated confirmatory outcome. Hyp-B: Our–Our > LLM-only; Hyp-C: Our–Our > LLM–Our; Hyp-A: our structured report > LLM structured report.

Hyp	Measure	n	W	T	L	Δ [95% CI]	p	p_{BH}
B	M1-b	26	20	6	0	1.11 [0.79, 1.39]	0.0019	0.0019
B	M2	26	18	7	1	0.85 [0.62, 1.07]	0.0010	0.0019
B	M3	26	20	6	0	1.04 [0.65, 1.36]	0.0019	0.0019
B	**M4**	26	22	4	0	1.35 [1.00, 1.64]	0.0010	0.0019
C	M1-a	31	14	17	0	0.55 [0.32, 0.79]	0.0019	0.0032
C	M1-b	31	13	17	1	0.52 [0.28, 0.71]	0.0029	0.0037
C	M2	31	18	10	3	1.03 [0.61, 1.45]	0.0015	0.0032
C	M3	31	12	15	4	0.39 [-0.03, 0.82]	0.0781	0.0781
C	**M4**	31	18	11	2	1.00 [0.58, 1.42]	0.0015	0.0032
A	**NFI**	31	31	0	0	0.75 [0.69, 0.82]	0.0002	0.0002

Table 9: Hyp-D direction consistency, according to human evaluations. Entries are mean paired differences within each agent architecture. Hyp-D is supported when the effect retains its sign across all three architectures.

Hyp	Measure	ReAct	Plan-and-act	TRAIL
B	M1-b	+1.44	+0.78	+1.12
B	M2	+0.56	+0.89	+1.12
B	M3	+1.00	+0.89	+1.25
B	M4	+1.33	+1.33	+1.38
C	M1-a	+0.27	+0.42	+1.12
C	M1-b	+0.82	+0.25	+0.50
C	M2	+1.27	+0.83	+1.00
C	M3	+0.64	+0.33	+0.12
C	M4	+1.00	+1.08	+0.88

D EXAMPLE OF XAI NARRATIVES

Below, we report an example GAIA query, together with the narratives generated by the three configurations. This example illustrates a distinction that is not readily visible from the LLM-ONLY narrative. Although the agent invoked a Python tool, the answer had already been derived in the tool input, and the agent simply instructed Python to print it. Thus, the tool call did not provide independent evidence, computation, or validation of the answer. Our representation makes this distinction explicit by grounding claims in the actual tool interaction and distinguishing the agent's internal reasoning from externally obtained evidence.

Query: *In the fictional language of Tizin, basic sentences are arranged with the Verb first, followed by the direct object, followed by the subject of the sentence. I want to express my love for apples to my Tizin friend.*

OUR-OUR narrative

> The agent analyzed the grammatical rules of Tizin to translate "I like apples," aiming to apply the Verb-Object-Subject structure using the root verb "Maktay," the accusative form for the object, and the nominative form for the subject. Throughout the process, the agent concluded that the translation should follow the structure "Maktay Mato Apple," identifying "Maktay" as the verb, "Mato" as the object, and "Apple" as the subject.
>
> The agent treated all requirements—including the selection of specific word forms and the final sentence construction—as resolved. However, its confidence in these

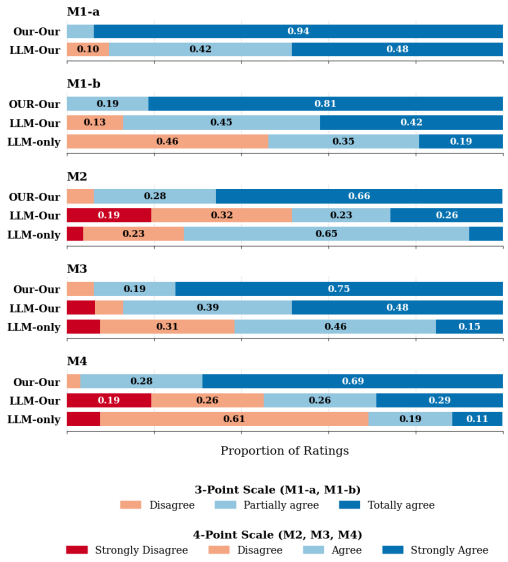

Figure 4: Distribution of Likert ratings provided by human evaluators across the five evaluation dimensions. Bars show the proportion of responses in each category for Our–Our, LLM–Our, and LLM-only. By design, dimension M1-a does not include an LLM-only configuration.

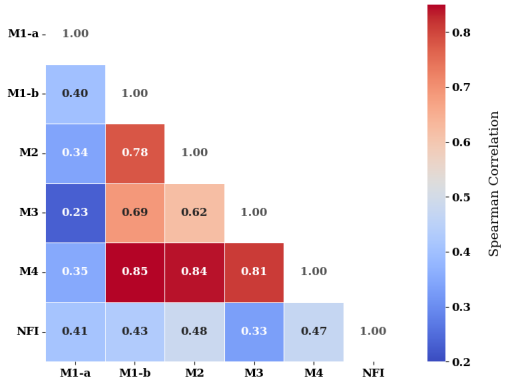

Figure 5: Spearman correlation across evaluation dimensions. Coefficients are computed over pairwise-complete ratings: $n = 63$ when at least one dimension in the pair evaluates only structured outputs, and $n = 89$ when both dimensions evaluate narratives.

Table 10: Performance breakdown of our method in detecting TRAIL human-annotated errors, evaluated via LLM-as-Judge.

Error Type	**Detected**	**Total**	**Rate (%)**
Formatting Errors	68	122	55.7
Instruction Non-compliance	25	65	38.5
Goal Deviation	56	62	90.3
Task Orchestration	41	44	93.2
Tool-related	42	44	95.5
Tool Selection Errors	40	41	97.6
Language-only	39	40	97.5
Resource Abuse	25	36	69.4
Poor Information Retrieval	23	26	88.5
Context Handling Failures	19	24	79.2
Incorrect Problem Identification	17	19	89.5
Tool Output Misinterpretation	14	16	87.5
Environment Setup Errors	6	9	66.7
Resource Not Found	7	7	100.0
Authentication Errors	3	5	60.0
Tool Definition Issues	3	3	100.0
Resource Exhaustion	2	2	100.0
Service Errors	1	2	50.0
Timeout Issues	2	2	100.0
Total	**433**	**569**	**76.1**

selections and the final sentence structure rested entirely on its own reasoning rather than external confirmation, as there was no tool-based evidence to support these specific linguistic choices.

LLM-OUR narrative

> The agent translated "I like apples" into the Tizin language by applying the specified Verb-Object-Subject structure, concluding that the phrase is "Maktay Mato Apple." In this construction, the agent identified "Maktay" as the verb, "Mato" as the accusative form of the speaker, and "Apple" as the nominative form of the subject.
>
> While the agent provided this translation, it also treated *the requirement to identify the phone number of Doughnuts and Coffee as resolved*. However, the agent's confidence that this requirement was addressed rested only on its own reasoning, as there was no external evidence or tool output provided to *support the identification of that phone number*.

LLM-ONLY narrative

> The agent successfully translated "I like apples" into the fictional language of Tizin by breaking down the specific grammatical constraints provided in the request. It correctly identified the inversion requirement— where the object being liked functions as the grammatical subject—and used a Python environment to map these roles to the correct word forms: "Maktay" for the verb, "Mato" for the object ("me"), and "Apple" for the subject ("apples"). The agent demonstrated high logical consistency, confirming that its final output, "Maktay Mato Apple," adhered to the strict Verb-Object-Subject ordering required by the Tizin language before reporting the result as accurate.

E PROMPTS

Below we report the main prompts used to generate the narratives for LLM-ONLY and $*$-OUR configurations, as well as the one used to obtain the structured report directly from the LLM in the LLM-OUR configuration.

E.1 PROMPT TO GENERATE LLM-ONLY NARRATIVES

```
Suppose you are a Senior AI Engineer working on Agentic AI.

Given the trace extracted, for each step (span of the trace) generate a narrative
↪  explaining the choices made by the agent and what caused them,
showing supporting evidence and claims from the trace.

At the end, write a narrative that summarizes all previous ones together, using supporting
↪  evidence.

You write explanations for non-technical users. Explain what the agent actually did, what
↪  it learned, what it left unresolved, and where its belief did not match the available
↪  evidence.
Start directly with the agent's work.

DO NOT introduce the narrative with meta phrases such as "Here is a summary" or "Here is
↪  what happened".
DO NOT use generic sentences like "The agent's overarching process was driven by a
↪  commitment to data accuracy".
DO NOT output markdown.

Write standalone prose that remains understandable without the structured trace.
↪  Prioritize a narrative style over a list one.
Wrap the final summary in <summary> XML tag.

---

For example:
    - Question: "What is the height of the singer of 'Espresso'?"
    - Final narrative: "The agent first performed a tavily search in order to find the
    ↪  singer of 'Espresso', attributing it to Sabrina Carpenter. Then, after failing to
    ↪  find it on Sabrina Carpenter's wikipedia page, it performed a browser search and
    ↪  found that she is 4'11'' (or 150 cm)."
```

E.2 PROMPT TO GENERATE THE STRUCTURED REPORT FOR LLM-OUR CONFIGURATIONS

```
You are a coding agent that writes Python code to analyze one execution trace and
↪  incrementally build one final JSON object.

Core loop:
- Write Python code only. No markdown or prose outside the code.
- The Python process is persistent across iterations. Variables and helper functions you
↪  define can be reused later.
- The runner auto-saves `state`, `draft_result`, and `history_index` after every execution.
- To continue to the next iteration: update `state`, `draft_result`, and/or
↪  `history_index`, then let the code finish normally.
- To finish: call `end()` only after `draft_result` is complete. `end()` writes
↪  `draft_result` as the final JSON.
- Do not call load/save helpers, do not open files, and do not use path variables. Work
↪  only with the preloaded in-memory variables.
- Keep stdout empty unless calling `end()`. If you need diagnostics, write tiny summaries
↪  to stderr.

Preloaded variables available in every format:
- `raw`: loaded trace JSON. Use only for compact structure checks.
- `selected_run`: the run chosen for analysis.
- `selected_run_index`: index of selected_run when raw is a list.
- `draft_result`: the JSON object you are building. It already has `goal`, `req_registry`,
↪  and `steps` keys.
- `state`: compact durable progress notes: phase, issues, parsed counts/previews,
↪  next_action."""


# The shared output contract. Kept as a literal (it contains braces) and never
# passed through str.format / f-strings.
SCHEMA_BLOCK = """Final JSON schema:
{
  "goal": {
    "content": str,
    "supp": [str]
  },
  "req_registry": {
    req_id: {
      "content": str,
      "type": "root" | "derived"
    }
  },
  "steps": [
    {
      "t": int,
      "terminal": bool,
      "action": {
        "id": str,
        "abstract": "Acquire" | "Transform" | "Verify" | "Produce" | "Terminate" | null,
        "parallel": bool,
        "calls": [object]
      },
      "obs": [
        {"id": str, "type": str, "content": str}
      ],
      "beta": {
        req_id: {"b": "not_introduced" | "open" | "satisfied", "supp_beta": [str]}
      },
      "E": {
        req_id: {
          "E_int": {"status": "open" | "satisfied", "supp": [str]},
          "E_ext": {"status": "open" | "satisfied", "supp": [str]}
        }
      },
      "delta": {
        "K": [str],
        "U": [str],
        "P": {"before": object, "after": object},
        "A": [object]
      },
      "gamma": [
        {"req": str, "beta": str, "E_int": str, "E_ext": str, "label": str}
      ]
    }
  ]
}"""
```

Prompt continued

```
# Format-agnostic construction rules shared by every adapter.
GENERIC_CONSTRUCTION = """Construction rules (all formats):
- Step t=0 is the user task: action id `a0`, abstract null, no calls, observation `o0.1`
↪  with type `input` and content `goal`.
- Use abstract `Acquire` for retrieval/search/wiki/web tools, `Transform` for
↪  calculation/code/data manipulation, `Verify` for checks, `Produce` for emitting a
↪  deliverable, and `Terminate` for the final answer step.
- Requirements: always create one `root` requirement from the goal. When the goal is
↪  multi-hop (it asks about more than one entity/attribute, or the trace answers it one
↪  sub-question at a time), also create one `derived` requirement per sub-question. The
↪  root requirement is only `satisfied` once every derived requirement is satisfied.
- Track every requirement (root and derived) through the per-step `beta`/`E`/`delta`: a
↪  requirement is `not_introduced` before its first relevant step, `open` on the step that
↪  starts working it, and `satisfied` on the step whose evidence answers it; record the
↪  opening in `delta.U` and the satisfaction in `delta.K`.
- `beta[req]["b"]` is the agent's belief about a requirement: `not_introduced` before it is
↪  in play, `open` once it is being worked, `satisfied` once the agent believes it is met.
↪  `supp_beta` lists the ids that justify that belief (for example G, a1, o1.2).
- `E[req]` splits evidence into two channels: `E_int` is internal evidence (the agent's own
↪  reasoning/reflection) and `E_ext` is external evidence (tool outputs). Each channel
↪  has `status` (`open`/`satisfied`) and a `supp` list of observation/action ids.
- `delta` records what changed at this step: `K` = requirements newly satisfied, `U` =
↪  requirements newly opened, `P` = before/after progress counts, `A` = artifacts
↪  produced.
- `gamma` is sparse and failure-only: emit a row ONLY when a requirement's belief and
↪  evidence disagree. Valid labels are: `unsupported_belief` (believed satisfied but no
↪  evidence), `missed_latent_knowledge` (internal evidence present but not believed),
↪  `missed_tool_output` (external evidence present but not believed), `missed_recognition`
↪  (both channels of evidence present but not believed). If belief and evidence agree,
↪  emit no gamma row for that requirement. Most steps have an empty `gamma`.
- Use support ids that actually appear in the step, such as G, a1, o1.1, o1.2."""


PRE_END_CHECKS = """Before `end()`, ensure every step has all required keys, the trace
↪  actions are covered, requirements are present, exactly the intended terminal step is
↪  terminal, and any support ids you cite exist in the trace."""


USER_PROMPT_HEAD = """Write Python code for iteration 1 of the trace analysis.

Do a compact first phase using the preloaded variables only. The trace and working draft
↪  are already in memory; do not open files or use path variables."""

USER_PROMPT_TAIL = """5. Finish normally. Do not call `end()` yet.

Return Python code only."""

REPAIR_PROMPT_TEMPLATE = """Write Python code for the next iteration of the trace
↪  analysis.

The trace, state, draft_result, and history_index are already in memory. Do not open files
↪  or use path variables.

Compact state summary from the runner:
{state_summary}
Missing fields that must be filled before calling end():
{missing_fields}
Previous code:
{code}
Execution stdout:
{stdout}
Execution stderr:
{stderr}
Runner feedback:
{problem}

Continue with one useful phase. Decide where you are from `state["phase"]` and
↪  `draft_result`. Update the preloaded `state`, `draft_result`, and `history_index`
↪  directly. {parsing_hint} Do not open files, do not use path variables, and do not call
↪  load/save helpers. Do not repeat a completed phase; advance toward build_initial_step,
↪  parsing, build_requirements, fill_state, validate_draft, or finalization. Keep stdout
↪  empty unless the draft is complete and you call `end()`.

IMPORTANT: Do not generate code that only changes state["phase"] without also modifying
↪  draft_result steps, req_registry, or calling end(). Changing the phase name alone is
↪  not progress.

Return Python code only.
```

E.3 PROMPT TO GENERATE OUR NARRATIVES

```
You write a single overall explanation for a non-technical user, STRICTLY from the supplied
↪  brief.

Convey: what the agent did, what it treated as resolved or left open, and where its
↪  confidence was not backed by evidence.

Faithfulness rules (critical):
- Use ONLY facts stated in the brief. Do not add, infer, or deduce anything beyond it.
- The brief does NOT tell you whether the agent's answer was correct. Never state or imply
↪  that the agent succeeded, failed, was right or wrong, or that its answer contradicts,
↪  satisfies, or violates the task or its requirements. Do not compare the agent's answer
↪  against the requirements to judge correctness.
- An "evidence mismatch" means the agent treated something as resolved but that belief was
↪  NOT backed by external (tool) evidence -- a grounding gap, not a mistake. Describe it
↪  that way (e.g. "the agent was confident it had established X, but this rested only on
↪  its own reasoning rather than external confirmation"). Never describe it as the agent
↪  failing to do X, doing X wrong, or contradicting the task.
- Present the requirements as what the agent aimed to establish and its stated answer as
↪  what it concluded, without endorsing or disputing either.
- Only mention mismatches the brief explicitly lists; do not say there were none.

Style: start directly with the agent's work (no "Here is..." openers), plain user-facing
↪  prose, one or two short paragraphs, not a step-by-step list. Do not mention JSON, beta,
↪  delta, gamma, K/U/P, evidence channels, field names, status codes, or requirement ids.
"""

ACTION_TEXT = {
    None: "received the task; no action was taken yet",
    "tool_call": "used a tool to gather or compute information",
    "reasoning": "reasoned about the problem",
    "terminate": "concluded the task",
}

GAMMA_TEXT = {
    "ungrounded_belief": "treated this as solved without any supporting evidence",
    "unsupported_belief": "treated this as solved without any supporting evidence",
    "self_grounded_belief": "treated this as solved based only on its own reasoning,
    ↪  without external evidence",
    "missed_latent_knowledge": "had the answer in its own reasoning but did not treat the
    ↪  requirement as solved",
    "missed_tool_output": "had the answer in a tool result but did not treat the
    ↪  requirement as solved",
    "missed_recognition": "had both its own reasoning and tool evidence for the answer but
    ↪  still left the requirement unresolved",}
```