AGENT TRACE THEORY: INFORMATION COVERAGE FOR EVALUATING STATEFUL TOOL-USING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using large language model (LLM) agents complete tasks through extended interactions that modify persistent environments and shape later decisions. Evaluations often reduce these executions to call validity, step-level correctness, or terminal success, obscuring evidence carried across the trajectory. We propose **Agent Trace Theory (ATT)**, an information-theoretic framework that distinguishes agent performance from evaluation coverage by relating full stateful traces to their observable projections. ATT characterizes the verdict-relevant information retained by a trace representation and identifies evidence omitted by call-only, step-local, and outcome-only views. It also formalizes how observations, state changes, backend conditions, ordering, and recovery jointly contribute to an auditable execution record. We operationalize ATT with a TraceState-centered framework that records persistent state, tool-induced transitions, backend conditions, and trajectory-level oracle evidence. Across multiple LLM backbones and prompting strategies, execution studies of state persistence, cross-tool dependency, interaction noise, and backend migration reveal dependency violations, stale-state errors, failed recovery, and backend-specific fragility. A same-trace projection audit shows that retaining state and transition evidence makes these failures more observable than reduced trajectory views, thereby separating diagnostic coverage from agent performance.

1 INTRODUCTION

Tool-using LLM agents increasingly express capability through sustained interaction with external environments rather than text generation alone (Yao et al., 2023; Schick et al., 2023; Li et al., 2023; Qin et al., 2024). They invoke APIs, manipulate resources, and coordinate multi-step tasks through tool interfaces and Model Context Protocol (MCP) servers (Xie et al., 2024; Wang et al., 2026b). Actions produce observations and state changes that guide later decisions (Yao et al., 2023; Lu et al., 2025; Yao et al., 2025). Realistic web and enterprise tasks likewise require extended interaction across multiple actions and observations (Yoran et al., 2024; Drouin et al., 2024). Agent behavior must therefore be evaluated as a stateful and interdependent execution process.

Existing benchmarks measure function-call validity, API selection, argument matching, intermediate milestones, or final success (Patil et al., 2025; Li et al., 2023; Lu et al., 2025; Zhou et al., 2024; Yao et al., 2025). These metrics provide useful local or terminal evidence, but they retain different projections of the underlying trajectory. A call-level record can miss whether an action changed the intended state. A terminal score can hide dependency violations, backend-specific failures, or recovery after an intermediate error. Even stateful benchmarks differ in which observations, state transitions, and execution conditions remain available to an evaluator. The resulting gap concerns evaluation coverage and should not be conflated with agent performance.

We distinguish the task specification Q, its oracle constraints Ω_Q, and the resulting trajectory-level verdict Y. The task specification describes the instruction, initial conditions, and tool configuration. The constraints specify state, dependency, permission, consistency, and recovery requirements. They are fixed before execution, whereas the verdict is computed from the recorded trajectory af-

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.

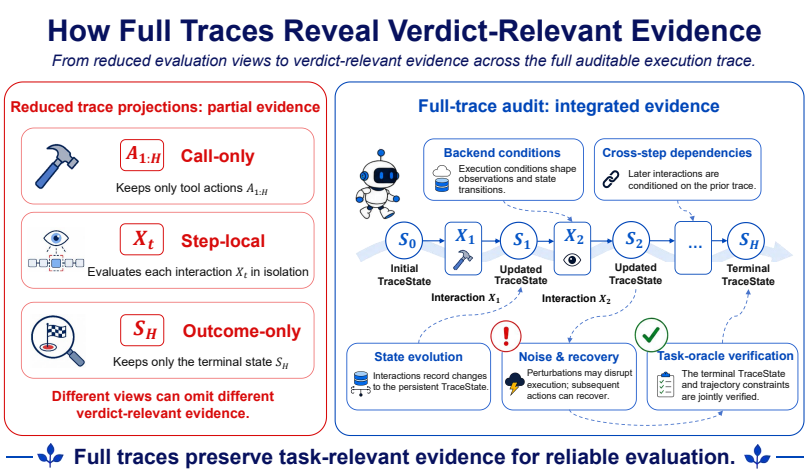

Figure 1: Conceptual motivation for trace-level evaluation. The word "ability" inside the illustration is used only as visual shorthand for verdict-relevant execution evidence made observable by the trace; it does not denote policy performance $\mathcal{P}(\pi)$. Call-only, step-local, and outcome-only records omit different parts of this evidence.

terward. This separation prevents task requirements from being treated as observed execution outcomes.

We propose **Agent Trace Theory (ATT)**, an information-theoretic framework relating complete stateful traces to the representations retained for evaluation. ATT treats each evaluation record as a projection of an auditable trace. It characterizes the verdict-relevant information preserved by that projection rather than redefining policy performance (Tishby et al., 2000; Alemi et al., 2017). This view connects action selection, observations, state propagation, backend conditions, ordering, and recovery within one evaluation object. It also identifies when reduced records cannot distinguish trajectories that support different verdicts. We operationalize ATT through **TraceState**, a shared state carrier read and written by tools during an episode. TraceState organizes entities, relations, resources, permissions, time, and delayed effects into a persistent execution state. The framework records actions, observations, state transitions, backend metadata, and trajectory-level oracle evidence. Four execution studies vary state persistence, cross-tool dependency, interaction noise, and backend migration, while an offline same-trace audit compares deterministic projections of the recorded executions. Our contributions are as follows:

- We distinguish trajectory-level policy performance from evaluation coverage and model each evaluation record as a projection of a complete, externally stateful trace.
- We define verdict-information coverage, projection fidelity, sequential trace information, sufficiency, parsimony, and projection ambiguity. These concepts characterize evidence retained or lost by call-only, independent, pointwise, and outcome-only evaluations.
- We introduce a TraceState-based audit representation with persistent state, append-only interaction recording, backend metadata, and trajectory-level oracle evidence.

2 Related Work

Existing work on tool-using agents spans local tool-use evaluation, stateful and trajectory-aware benchmarks, and sequential information-theoretic perspectives. Toolformer and ReAct establish tool-use and reasoning–action–observation patterns (Schick et al., 2023; Yao et al., 2023), while API-Bank, ToolLLM, Gorilla, and BFCL evaluate API selection, retrieval, argument generation, documentation grounding, and structured function calls (Li et al., 2023; Qin et al., 2024; Patil et al., 2024; 2025). These benchmarks are valuable for assessing local tool-use correctness, but their retained records need not expose state transitions or cross-step constraints. More recent stateful benchmarks, including ToolSandbox, τ-bench, and τ^2-bench, incorporate multi-turn interaction and

persistent environments (Lu et al., 2025; Yao et al., 2025; Barres et al., 2025), while WebArena, OSWorld, CRMArena, and AndroidWorld extend evaluation to web, computer, CRM, and mobile settings (Zhou et al., 2024; Xie et al., 2024; Huang et al., 2025; Rawles et al., 2025). MCP-Bench further studies cross-tool coordination and live backends (Wang et al., 2026b), and AgentDiagnose analyzes failures in complete trajectories (Ou et al., 2025). Complementing these empirical evaluation frameworks, sequential agent formulations condition decisions on interaction histories (Wang et al., 2026a; Yao et al., 2023), while information theory provides tools for characterizing mutual information, data processing, sufficient representations, and relevance-preserving compression (Cover & Thomas, 2006; Tishby et al., 2000; Alemi et al., 2017). ATT builds on these perspectives to study a distinct question: how projecting the same stateful trajectory into different evaluation records changes the verdict-relevant information retained and the resulting decision ambiguity, without making claims of causal identification.

3 Agent Trace Theory

A tool-using agent generates a stateful execution trajectory, whereas an evaluator usually retains only a partial record of that trajectory. Agent Trace Theory (ATT) formalizes the information retained and lost by such trace projections. ATT is an evaluation theory rather than a redefinition of policy performance: it measures how much information an evaluation record preserves about a trajectory-level verdict under a fixed task and environment context. The preliminaries of our information-theoretic framework is provided in **Appendix A.1**.

3.1 Evaluation Context and Chronological Trace

Let π denote an agent policy and $\mathcal{E}$ a stateful environment. Let K collect non-execution context available to every evaluator, including the task family, task specification, oracle constraints, disclosed initial conditions, and evaluation protocol. Conditioning on K prevents information that is already supplied by the task identity or protocol from being credited to a trace representation.

Let $Y \in \mathcal{Y}$ denote the trajectory-level oracle verdict. Throughout, $\mathcal{Y}$ is finite and Y is discrete; a structured verdict is represented as a finite tuple containing final-state, dependency, permission, consistency, and recovery components. This restriction applies to the entropy used to normalize coverage, while trace and context variables may be mixed or continuous whenever the relevant conditional mutual informations are well defined. The evaluation distribution induced by the task mixture, policy, environment, and oracle is

$$P_{\pi,\mathcal{E}}\big(K, Y, S_0, M_{1:H}, X_{1:H}\big), \tag{1}$$

where S_0 is the initial environment state and M_t is an optional recorded controller or runtime state available before the external interaction at step t. M_t does not denote private chain-of-thought. When no controller state is recorded, it is treated as a constant and contributes zero information.

For step t, define the externally auditable interaction

$$X_t = (A_t, O_t, \Delta S_t, B_t), \tag{2}$$

where A_t is the tool/API/MCP action and O_t is the observation delivered to the policy. When observational perturbations are used, the unmodified execution feedback O_t^{raw} is retained only as audit metadata and is not exposed to the policy. ΔS_t is an auditable state-transition record. B_t is the realized execution-condition record, including the configured backend b_t, active backend or schema, pre- and postcondition outcomes, scheduled effects, latency, timeout or fault status, stale-state metadata, and MCP execution conditions. The canonical ATT record therefore remains the four-tuple in Equation (2). The chronological length-H trace is

$$\mathcal{T}_H^{\pi} = (S_0, M_1, X_1, \ldots, M_H, X_H). \tag{3}$$

The externally auditable trace used in our experiments is the special case $M_t \equiv \varnothing$, namely $\mathcal{T}_H^{\pi,\mathrm{ext}} = (S_0, X_{1:H})$. Thus, ATT does not require private reasoning traces.

The sequence $X_{1:H}$ is history-conditioned: earlier actions can create or remove entities, satisfy prerequisites, consume resources, alter permissions, schedule delayed effects, and change the meaning of later observations. The ordering in Equation (3) is therefore part of the trace representation rather than incidental metadata.

Variable episode length. H denotes a common maximum horizon. An episode terminating at $\tau < H$ retains its termination time (or an equivalent termination marker) and is padded thereafter with deterministic null controller states and null interactions that leave the state unchanged. The padded representation is a bijective encoding of the original variable-length trace together with τ, so padding itself neither adds nor removes verdict information.

3.2 Trace Projections, Coverage, and Fidelity

An evaluator normally observes a projected record $\Xi_\phi = \phi(\mathcal{T}_H^\pi)$, where ϕ may retain only calls, calls and observations, terminal state, state changes, backend evidence, or the complete trace.

Definition 1 (Trace-projection coverage). *For $0 < H(Y \mid K) < \infty$, the evaluation coverage of projection ϕ is*

$$\mathcal{C}_\phi(\pi) = \frac{I(Y; \Xi_\phi \mid K)}{H(Y \mid K)}. \tag{4}$$

For the identity projection, write $\mathcal{C}_{\text{full}}(\pi) = \mathcal{C}_{\text{id}}(\pi)$.

Since Y is finite and discrete, $0 \le I(Y; \Xi_\phi \mid K) \le H(Y \mid K)$, and therefore $0 \le \mathcal{C}_\phi(\pi) \le 1$. If $H(Y \mid K) = 0$, coverage is undefined rather than assigned an arbitrary endpoint: under the supplied context, there is no residual verdict uncertainty for a trace representation to resolve.

For a structured verdict $Y = (Y^{(1)}, \ldots, Y^{(R)})$, ATT may additionally report component-wise coverage

$$\mathcal{C}_\phi^{(r)}(\pi) = \frac{I(Y^{(r)}; \Xi_\phi \mid K)}{H(Y^{(r)} \mid K)}, \quad H(Y^{(r)} \mid K) > 0. \tag{5}$$

Joint coverage characterizes the complete verdict vector, whereas component-wise coverage separates state, dependency, permission, consistency, or recovery evidence when those components have different evaluation roles.

Coverage is a population quantity under the fixed evaluation distribution in Equation (1). It is not a score assigned to a single trajectory, and it can change when the task mixture, policy, environment, or oracle changes. In particular, a single deterministic execution does not by itself define or identify coverage; estimation requires a population with residual verdict variation under shared contexts.

Definition 2 (Policy performance). *For a task utility u, policy performance is $\mathcal{P}(\pi) = \mathbb{E}[u(Y)]$. For a binary success verdict, $\mathcal{P}(\pi) = \Pr(Y = 1)$.*

Definitions 1 and 2 separate evaluator informativeness from agent success. A projection can be highly informative about failure while the policy performs poorly, or it can hide failures while reporting a high local score.

Definition 3 (Projection fidelity). *When $I(Y; \mathcal{T}_H^\pi \mid K) > 0$, the fidelity of projection ϕ is*

$$\rho_\pi(\phi) = \frac{I(Y; \Xi_\phi \mid K)}{I(Y; \mathcal{T}_H^\pi \mid K)}. \tag{6}$$

The fidelity is derived from the projection map and evaluation distribution; it is not a free parameter. Directly from the definitions,

$$\mathcal{C}_\phi(\pi) = \rho_\pi(\phi)\, \mathcal{C}_{\text{full}}(\pi). \tag{7}$$

Proposition 1 (Trace projection monotonicity). *If $\Xi_{\phi_1} = g(\Xi_{\phi_2})$ for a deterministic map g, then*

$$I(Y; \Xi_{\phi_1} \mid K) \le I(Y; \Xi_{\phi_2} \mid K), \quad \mathcal{C}_{\phi_1}(\pi) \le \mathcal{C}_{\phi_2}(\pi). \tag{8}$$

The result follows from the conditional data-processing inequality. Proof is provided in **Appendix A.2**. It applies only when one record is a deterministic post-processing of the other under the same target Y and context K; it does not rank benchmarks using different verdicts.

3.3 Chronological Trace Information

The chronological trace permits an exact decomposition that respects the order in which controller states and external consequences become available. For compactness, define the pre-interaction and post-controller histories

$$\mathcal{H}_t^- = (K, S_0, M_{<t}, X_{<t}), \qquad \mathcal{H}_t^+ = (\mathcal{H}_t^-, M_t). \tag{9}$$

Proposition 2 (Chronological trace decomposition)**.** *The verdict information in the complete trace satisfies*

$$I(Y;\mathcal{T}_H^\pi \mid K) = I(Y;S_0 \mid K) + \sum_{t=1}^{H} I(Y;M_t \mid \mathcal{H}_t^-) + \sum_{t=1}^{H} I(Y;X_t \mid \mathcal{H}_t^+). \tag{10}$$

The derivation is provided in **Appendix A.3**. Each term measures additional verdict information given the chronologically available prefix and should not be interpreted as a causal effect. In our externally auditable implementation, $M_t \equiv \varnothing$, so the controller-state sum vanishes and $\mathcal{H}_t^+ = (K, S_0, X_{<t})$. If the full initial state is included in K, then $I(Y;S_0 \mid K) = 0$.

3.4 Call-Only and Independent Views

A call-only record keeps the action sequence but discards observations, state changes, backend conditions, and execution faults:

$$\Xi_{\text{call}} = \phi_{\text{call}}(\mathcal{T}_H^\pi) = A_{1:H}. \tag{11}$$

Proposition 3 (Call-only coverage bound)**.** *Because Ξ_{call} is a deterministic function of the complete trace,*

$$I(Y;\Xi_{\text{call}} \mid K) \le I(Y;\mathcal{T}_H^\pi \mid K). \tag{12}$$

Moreover, because it is a function of $X_{1:H}$,

$$I(Y;\Xi_{\text{call}} \mid K, S_0) \le I(Y;X_{1:H} \mid K, S_0). \tag{13}$$

The detailed conditional-Markov argument is given in **Appendix A.4**. The inequalities may be equalities for a particular task family, but equality must be established rather than assumed. A second simplification independently aggregates individual interactions:

$$\mathcal{I}_{\text{ind}} = \sum_{t=1}^{H} I(Y;X_t \mid K, S_0). \tag{14}$$

The sequential quantity is

$$\mathcal{I}_{\text{seq}} = I(Y;X_{1:H} \mid K, S_0) = \sum_{t=1}^{H} I(Y;X_t \mid K, S_0, X_{<t}). \tag{15}$$

Proposition 4 (No universal ordering for independent-call aggregation)**.** *There is no universal ordering between $\mathcal{I}_{\text{ind}}$ and $\mathcal{I}_{\text{seq}}$.*

Appendix A.5 provides two constructive proofs. Redundant interactions can make $\mathcal{I}_{\text{ind}} > \mathcal{I}_{\text{seq}}$ because the same evidence is counted repeatedly. Synergistic interactions can make $\mathcal{I}_{\text{ind}} < \mathcal{I}_{\text{seq}}$ because no single interaction reveals the verdict although their combination does. Thus, independent-call sums can both over-count and under-count sequential evidence.

3.5 Trace Sufficiency and Parsimony

Let $\mathcal{J}$ index components available in the complete trace and write $Z_{\mathcal{J}} = (Z_j)_{j\in\mathcal{J}}$. Components may include actions, observations, state changes, backend fields, controller states, ordering information, and recovery evidence. For $\mathcal{S} \subseteq \mathcal{J}$, let $Z_{\mathcal{S}}$ be the corresponding subrepresentation. This component formulation avoids declaring a field globally relevant or superfluous from its marginal mutual information alone. A field can be redundant after another field is known, or become informative only in combination with other fields.

Definition 4 (Trace sufficiency)**.** *A deterministic projection Ξ_ϕ is trace-sufficient if*

$$I(Y;\Xi_\phi \mid K) = I(Y;\mathcal{T}_H^\pi \mid K). \tag{16}$$

Proposition 5 (Equivalent forms of trace sufficiency)**.** *For a deterministic projection $\Xi_\phi = \phi(\mathcal{T}_H^\pi)$, the following statements are equivalent:*

$$\begin{aligned} I(Y;\Xi_\phi \mid K) &= I(Y;\mathcal{T}_H^\pi \mid K), \\ H(Y \mid \Xi_\phi, K) &= H(Y \mid \mathcal{T}_H^\pi, K), \\ I(Y;\mathcal{T}_H^\pi \mid \Xi_\phi, K) &= 0. \end{aligned} \tag{17}$$

The equivalence is proved line by line in **Appendix A.6**.

Definition 5 (Conditional redundancy and minimal sufficiency)**.** *For $j \in \mathcal{S}$, Z_j is conditionally redundant relative to the remaining retained components if*

$$I\big(Y; Z_j \mid Z_{\mathcal{S}\setminus\{j\}}, K\big) = 0. \tag{18}$$

A sufficient subset $\mathcal{S}$ is minimally sufficient if removing any $j \in \mathcal{S}$ makes it insufficient.

Minimal sufficiency need not be unique. It is also not equivalent to the shortest serialized log because different fields may have different storage, privacy, instrumentation, and evaluation costs.

Proposition 6 (Incremental coverage of a restored component)**.** *Let $\Xi_{\phi,+j} = (\Xi_\phi, Z_j)$. Then*

$$\mathcal{C}_{\phi,+j}(\pi) - \mathcal{C}_\phi(\pi) = \frac{I(Y; Z_j \mid \Xi_\phi, K)}{H(Y \mid K)} \geq 0. \tag{19}$$

The gain is zero exactly when Z_j is conditionally redundant given the current projection and context.

The proof in **Appendix A.7** follows directly from the conditional chain rule. The proposition makes ATT operational: it identifies the verdict information gained by restoring an omitted observation, state-transition, or backend field. It does not imply that greedy field restoration is globally optimal, because multiple fields can exhibit synergy.

Cost-aware parsimony. For a candidate projection family Φ, trace cost c, and tolerance $\varepsilon \geq 0$, define

$$\begin{aligned} &\inf_{\phi\in\Phi} \quad \mathbb{E}[c(\Xi_\phi)] \\ &\text{s.t.} \quad H(Y \mid \Xi_\phi, K) \leq H(Y \mid \mathcal{T}_H^\pi, K) + \varepsilon. \end{aligned} \tag{20}$$

If the infimum is attained, any minimizer is a cost-aware parsimonious projection. The objective permits controlled reductions in logging or audit cost while explicitly bounding additional verdict uncertainty.

3.6 Chronological Form of Projection Coverage

We now combine projection fidelity with the chronological decomposition.

Corollary 1 (Chronological form of projection coverage)**.** *When $I(Y; \mathcal{T}_H^\pi \mid K) > 0$ and $0 < H(Y \mid K) < \infty$,*

$$\mathcal{C}_\phi(\pi) = \rho_\pi(\phi) \frac{1}{H(Y \mid K)} \left[I(Y; S_0 \mid K) + \sum_{t=1}^{H} I(Y; M_t \mid \mathcal{H}_t^-) + \sum_{t=1}^{H} I(Y; X_t \mid \mathcal{H}_t^+) \right]. \tag{21}$$

A complete derivation is provided in **Appendix A.8**. This corollary combines Definition 3 with Proposition 2; it is a chronological restatement of projection coverage rather than a new information inequality. It does not identify coverage with policy performance.

In our externally auditable implementation, M_t is constant, so the middle sum vanishes. The remaining terms correspond directly to the initial state and the prefix-conditioned external interactions recorded by TraceState.

3.7 Coverage Gap and Projection Ambiguity

Theorem 1 (Coverage gap)**.** *For any deterministic projection $\Xi_\phi = \phi(\mathcal{T}_H^\pi)$ with $0 < H(Y \mid K) < \infty$, we have*

$$\mathcal{C}_{\text{full}}(\pi) - \mathcal{C}_\phi(\pi) = \frac{I(Y; \mathcal{T}_H^\pi \mid \Xi_\phi, K)}{H(Y \mid K)}. \tag{22}$$

The full chain-rule proof is given in **Appendix A.9**. The right-hand side is precisely the verdict information that remains in the full trace after the projection and common context are known. The gap is zero if and only if the projection is trace-sufficient. For the deterministic trace oracle instantiated in our framework,

$$Y = G_{\Omega_Q}(\mathcal{T}_H^{\pi,\text{ext}}, K), \tag{23}$$

where $\mathcal{T}_H^{\pi,\mathrm{ext}}$ is a deterministic projection of $\mathcal{T}_H^{\pi}$. Hence $H(Y \mid \mathcal{T}_H^{\pi}, K) = 0$ and $\mathcal{C}_{\mathrm{full}}(\pi) = 1$. Consequently,

$$1 - \mathcal{C}_{\phi}(\pi) = \frac{H(Y \mid \Xi_{\phi}, K)}{H(Y \mid K)}. \tag{24}$$

This special case is stated only after the general coverage-gap identity, avoiding an implicit assumption that every adjudication process is noiseless.

Definition 6 (Projection ambiguity region)**.** *For a discrete verdict Y, define*

$$\mathcal{A}_{\phi} = \left\{ (k, \xi) : \max_{y} P(Y = y \mid K = k, \Xi_{\phi} = \xi) < 1 \right\}. \tag{25}$$

A projection is verdict-ambiguous if $P((K, \Xi_{\phi}) \in \mathcal{A}_{\phi}) > 0$.

Proposition 7 (Irreducible error under an ambiguous projection)**.** *For 0–1 verdict loss, the minimum achievable error of any evaluator that observes only (K, Ξ_{ϕ}) is*

$$R_{\phi}^{\star} = \mathbb{E}\left[1 - \max_{y} P(Y = y \mid K, \Xi_{\phi})\right]. \tag{26}$$

Hence $R_{\phi}^{\star} > 0$ whenever the ambiguity region has positive probability.

Appendix A.10 provides the proof. This probability-level definition remains valid when trace components such as latency are continuous, unlike a definition requiring individual traces to have positive point mass.

3.8 Beyond ATT: Benchmark Families and Pointwise Under-Coverage

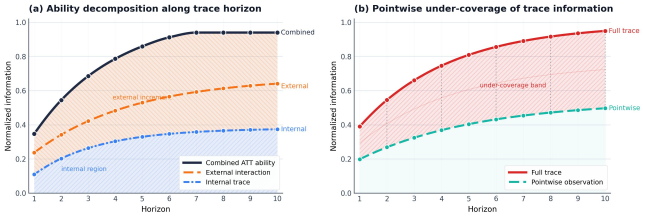

Figure 2: Schematic illustration of ATT. Panel (a) illustrates the chronological accumulation of verdict-relevant trace information, and Panel (b) illustrates possible pointwise under-coverage. For fixed Y and K, nested projections are ordered by Proposition 1; no general monotonicity is claimed for the projection gap across horizons.

A pointwise benchmark retains $Z_{\mathrm{pt}} = \psi(\mathcal{T}_H^{\pi})$. When it evaluates the same verdict Y under the same context K, define $\Delta_{\mathrm{pt}} = \mathcal{C}_{\mathrm{full}}(\pi) - \mathcal{C}_{\mathrm{pt}}(\pi)$. Then, for $0 < H(Y \mid K) < \infty$,

$$\Delta_{\mathrm{pt}} = \frac{I(Y; \mathcal{T}_H^{\pi} \mid Z_{\mathrm{pt}}, K)}{H(Y \mid K)} \geq 0. \tag{27}$$

Thus, a reduced representation cannot retain more verdict information than the full trace. The derivation is given in **Appendix A.11**. This result concerns information coverage, not empirical accuracy. A pointwise benchmark may instead evaluate a local target Y_{pt}, such as call validity, whose accuracy is not directly comparable with coverage of the trajectory verdict Y. Appendix A.12 compares representative benchmark families (Patil et al., 2025; 2024; Li et al., 2023; Qin et al., 2024; Lu et al., 2025; Yao et al., 2025; Barres et al., 2025; Huang et al., 2025; Wang et al., 2026b). ATT characterizes their retained trajectory evidence rather than ranking their overall quality.

3.9 ATT-Guided Evaluation Framework

We instantiate the auditable interaction $X_t = (A_t, O_t, \Delta S_t, B_t)$ with a TraceState-centered execution pipeline. The pipeline maintains persistent state, executes and records tool interactions, and evaluates the completed trajectory under task-specific oracle constraints. The detailed execution pipeline and the mapping between system components and ATT variables are illustrated in Figure 5 and described in **Appendix B.1**.

TraceState execution. TraceState is the shared persistent state read and modified by all tools within an episode. Before each action, the executor advances the environment and applies ready effects, producing the ordered transition $S_{t-1} \xrightarrow{\text{prepare}} S_t^{\text{pre}} \xrightarrow{A_t, O_t, B_t} S_t$. This separates effects inherited from earlier interactions from changes caused by the current action. Paired snapshots make both components of ΔS_t auditable without exposing oracle-only information to the policy. Each tool follows a state-aware contract whose admissibility and postconditions depend on the prepared state. The executor supplies an agent-visible state summary, executes the action, updates TraceState, and appends the resulting interaction to the canonical trace. Validation failures, blocked actions, and failed postconditions remain explicit in B_t, allowing later actions to be evaluated against the evidence and state produced by the trace prefix. TraceState semantics, tool contracts, and recording procedures are detailed in **Appendix B.2–B.4**.

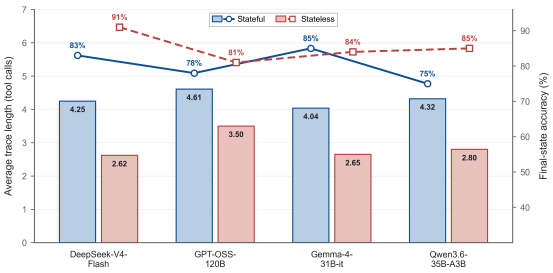

Figure 3: State persistence and observable execution traces across four model backbones. Bars show the average number of executed tool calls per case, while lines report final-state accuracy, defined by whether the terminal TraceState satisfies all task goals.

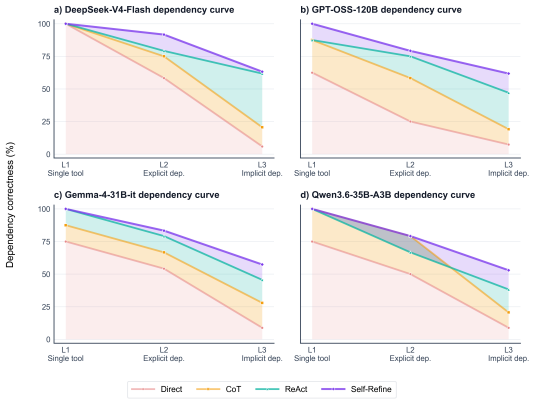

Figure 4: Dependency correctness across increasing cross-tool dependency levels. A trajectory is dependency-correct only if it reaches the target final state while satisfying all task-specific prerequisite and ordering constraints.

Backends and perturbations. ATT distinguishes the configured backend b_t from the realized execution record B_t. Mock, sandbox, and live MCP backends differ in schema enforcement, runtime constraints, feedback, and connectivity while preserving the intended task semantics. Changing b_t alters the execution distribution, whereas changing a projection only removes information from an already recorded trace. Perturbations are recorded through the same channel. Pre-call faults, including timeouts and schema drift, may block execution, whereas post-call faults alter the observation after the underlying state update. The agent-visible feedback is stored as O_t, while the unmodified response and fault metadata remain in B_t for audit. Backend and perturbation semantics are specified in **Appendix B.5–B.6**.

Oracle and evaluation projections. The framework retains one canonical externally auditable trace and derives call-only, Action–Observation, outcome-only, state-aware, and full-audit views offline. Because each view is a deterministic projection of the same trace, their comparison does not rerun the policy. The trajectory-level oracle evaluates terminal-state, dependency, permission, consistency, and recovery predicates. The associated metrics report call status, state correctness, dependency satisfaction, recovery, and execution cost without treating them as direct estimates of ATT's information-theoretic coverage. Oracle construction, projection fields, metrics, and execution protocols are provided in **Appendix B.7**.

4 Experiments

Experimental Setup. All experiments used a closed-loop tool-execution environment with four LLM backbones: DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), Qwen3.6-35B-A3B (Qwen Team, 2026), Gemma-4-31B-it (Gemma Team, 2026), and gpt-oss-120b (OpenAI, 2025). At each step, the model selected an action from the task instruction, tool schemas, prepared state summary, and interaction history. Where applicable, Direct, CoT, ReAct, and Self-Refine were compared under matched tool schemas, execution constraints, and termination criteria. Oracle-triggered termina-

tion was applied uniformly. Tasks were derived from ToolSandbox (Lu et al., 2025) and MCP-Bench (Wang et al., 2026b), covering explicit state transitions and mock, sandbox, and live MCP execution. Our **experimental results** are presented as follows.

Stateful vs. Stateless Execution. We compared stateful and stateless execution on ToolSandbox-derived tasks under the same closed-loop policy. Stateful execution increased trace length by 31.7%–62.2% across backbones (Figure 3), while final-state accuracy changed by -8, -3, $+1$, and -10 points for DeepSeek, GPT-OSS, Gemma, and Qwen. Longer traces exposed additional prerequisite, update, and coordination failures. Because stateful runs also used more actions and tokens, this is a system-level ablation rather than a causal estimate. See **Appendix C.1** for details. We also grouped tasks into L1 single-tool, L2 explicit two-tool, and L3 multi-tool dependency levels. As shown in Figure 4, dependency correctness declined with depth for every strategy, with L1-to-L3 drops of 41.2–71.7 points, showing that locally valid calls do not guarantee prerequisite- and order-consistent trajectories. See **Appendix C.2** for more details.

Table 1: Trajectory robustness under interaction noise, averaged across models and strategies. Success@1 uses the first perturbed trajectory, and Recovery is pooled over eligible episodes.

Noise type	Success@1	State Acc.	Recovery
Timeout	25.3%	54.1%	19.3%
Schema drift	19.7%	35.3%	21.3%
Stale state	8.8%	70.0%	0.0%
Vague observation	19.7%	50.3%	37.1%
Misleading observation	26.2%	75.9%	29.2%

Robustness to Interaction Noise. Under five perturbations—timeout, schema drift, stale state, vague observation, and misleading observation—State Accuracy ranged from 35.3% to 75.9%, whereas Success@1 remained between 8.8% and 26.2% and Recovery between 0% and 37.1% (Table 1). Stale state showed the largest mismatch, with 70.0% State Accuracy but only 8.8% Success@1 and no recovery, indicating that terminal-state correctness alone does not imply valid or recoverable execution. See **Appendix C.3** for details.

Table 2: Backend migration accuracy (%). (a) averages over strategies; (b) averages over models.

(a) Models				*(b) Strategies*			
	Mock	Sand.	Live		Mock	Sand.	Live
DeepSeek	71.7	63.3	60.8	Direct	67.5	52.5	44.2
GPT-OSS	72.5	65.0	58.3	CoT	67.5	51.7	46.7
Gemma	72.5	66.7	63.3	ReAct	77.5	76.7	74.2
Qwen	71.7	63.3	57.5	Self-Refine	75.8	77.5	75.0

Across matched mock, sandbox, and live MCP backends, final-state accuracy decreased by 9.2–14.2 points from mock to live across models (Table 2(a)). Direct and CoT dropped by 23.3 and 20.8 points, compared with 3.3 and 0.8 points for ReAct and Self-Refine (Table 2(b)), revealing strategy-dependent backend fragility. See **Appendix C.4** for details. **For a same-trace comparison,** see **Appendix C.5**. Across offline projections of the same traces, false acceptance decreased from 56.9% (Call-only) to 45.0% (Action–Observation), 14.4% (State-aware), and 0.0% (Full-audit), while Trajectory-Failure Recall increased from 0.0% to 36.9%, 87.6%, and 100.0% (Table 14). Outcome-only still yielded 17.9% false acceptance and 80.0% recall.

5 Conclusion

We proposed **ATT**, an information-theoretic framework that separates agent performance from evaluation coverage by treating evaluation records as projections of complete stateful traces. ATT is operationalized through a TraceState-centered framework that records stateful interactions and trajectory-level oracle evidence. Across state persistence, cross-tool dependency, interaction noise, and backend migration, the evaluation exposed prerequisite violations, stale-state errors, failed recovery, and backend-specific fragility hidden by call-level or outcome-only metrics. The same-trace audit further showed that reduced projections increase verdict disagreement and reduce trajectory-failure recall by omitting state and execution-condition evidence. These findings support stateful traces as auditable evaluation objects while keeping ATT distinct from policy performance. ATT complements task-specific benchmarks by making projection choices and diagnostic limitations explicit. Future work will study coverage at larger scale, noisy or incomplete oracles, and trace-aware diagnosis, training, and recovery.

AI USE STATEMENT

Generative AI tools were used for limited language polishing and presentation assistance, including grammar and wording suggestions for parts of the manuscript. The authors reviewed the assisted edits and take responsibility for the final content. No generative AI tools were used to generate research ideas, methods, algorithms, proofs, datasets, experiments, quantitative results, or conclusions.

REPRODUCIBILITY STATEMENT

We provide the implementation and configuration files required to reproduce the experiments in an anonymous code repository: `https://anonymous.4open.science/r/AgentInferKit-F521/`. The repository contains the implementations of the evaluated inference strategies, experiment and evaluation pipelines, configuration files, and instructions for running the reported experiments. The main text describes the experimental protocol and evaluation methodology, while additional implementation details and experimental settings are documented in the appendices and the accompanying repository.

REFERENCES

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyxQzBceg.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982. Oral at ICML 2026.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, 2006. doi: 10.1002/047174882X.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the capacity of LLM agents to perform professional CRM tasks in realistic environments. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3830–3850, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.194.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of*

the Association for Computational Linguistics: NAACL 2025, pp. 1160–1183, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.65.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Tianyue Ou, Wanyao Guo, Apurva Gandhi, Graham Neubig, and Xiang Yue. AgentDiagnose: An open toolkit for diagnosing LLM agent trajectories. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 207–215. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-demos.15.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565, 2024. doi: 10.52202/079017-4020.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. URL https://arxiv.org/abs/physics/0004057.

Hongru Wang, Cheng Qian, Manling Li, Jiahao Qiu, Boyang Xue, Mengdi Wang, Heng Ji, Amos Storkey, and Kam-Fai Wong. Position: Agent should invoke external tools only when epistemically necessary. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, pp. 1–16, Seoul, South Korea, 2026a. OpenReview. URL https://openreview.net/forum?id=5wTHg2zQRA.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. MCP-Bench: Benchmarking tool-using llm agents with complex real-world tasks via mcp servers. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=fe8mzHwMxN.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. doi: 10.52202/079017-1650.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8938–8968. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.505.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

A Detailed Proofs for Agent Trace Theory

This section gives the complete assumptions, derivations, and proofs for all formal results in the Agent Trace Theory section of the main paper. The main paper states each result together with its operational interpretation; this appendix preserves the full information-theoretic argument.

Correspondence with the main paper. The proof order matches the order of the formal results in the main text:

Main-text result	Appendix location
Trace projection monotonicity	Section A.2
Chronological trace decomposition	Section A.3
Call-only coverage bound	Section A.4
No universal ordering for independent calls	Section A.5
Equivalent forms of trace sufficiency	Section A.6
Restored-component coverage gain	Section A.7
Chronological form of projection coverage	Section A.8
Coverage gap	Section A.9
Projection ambiguity and Bayes risk	Section A.10
Pointwise benchmark under-coverage	Section A.11

Each subsection repeats the exact result title used in the main paper, so the statement–proof correspondence remains explicit throughout the appendix.

A.1 Information-Theoretic Preliminaries

Notation. Let K be common non-execution context and let $Y \in \mathcal{Y}$ be a finite discrete trajectory-level verdict; a structured verdict is a finite tuple of discrete components. Let S_0 be the initial environment state, M_t an optional recorded controller or runtime state available before interaction t, and $X_t = (A_t, O_t, \Delta S_t, B_t)$ the externally auditable interaction. Here, ΔS_t is an auditable state-transition record and B_t is the realized execution-condition record; b_t denotes the configured backend before execution. The chronological trace is

$$\mathcal{T}_H^{\pi} = (S_0, M_1, X_1, \ldots, M_H, X_H). \tag{A.1}$$

The externally auditable special case used in the experiments is

$$\mathcal{T}_H^{\pi,\mathrm{ext}} = (S_0, X_{1:H}), \qquad M_t \equiv \varnothing. \tag{A.2}$$

For variable-length episodes, the termination time (or an equivalent marker) is retained and all later positions are filled by deterministic null values. This padded representation is bijective with the original variable-length trace and its termination time, so padding does not change mutual information. As in the main text, define

$$\mathcal{H}_t^- = (K, S_0, M_{<t}, X_{<t}), \qquad \mathcal{H}_t^+ = (\mathcal{H}_t^-, M_t). \tag{A.3}$$

All entropies and mutual informations are taken under the fixed evaluation distribution induced by the task mixture, policy, environment, and oracle. Coverage is defined only when $0 < H(Y \mid K) < \infty$; when $H(Y \mid K) = 0$, the supplied context leaves no residual verdict uncertainty to be resolved.

For discrete random variables U, V, W, mutual information and conditional mutual information are

$$\begin{aligned} I(U;V) &= H(U) - H(U \mid V), \\ I(U;V \mid W) &= H(U \mid W) - H(U \mid V, W). \end{aligned} \tag{A.4}$$

The verdict variable Y remains finite and discrete throughout ATT. Context and trace variables may be discrete, mixed, or continuous whenever regular conditional distributions exist and the displayed mutual-information quantities are finite. ATT never normalizes by differential entropy. Conditional mutual information is non-negative:

$$I(U;V \mid W) \geq 0. \tag{A.5}$$

The conditional chain rule is

$$I(U; V_1, \ldots, V_n \mid W) = \sum_{i=1}^{n} I(U; V_i \mid W, V_{<i}), \tag{A.6}$$

where $V_{<i} = (V_1, \ldots, V_{i-1})$.

If $Z = g(V)$ is a deterministic function of V, then, conditional on W, $U \to V \to Z$ is a Markov chain because $p(z \mid u, v, w) = p(z \mid v, w)$. The conditional data-processing inequality therefore gives

$$I(U;Z \mid W) \leq I(U;V \mid W). \tag{A.7}$$

In addition, determinism implies

$$I(U;Z \mid V, W) = 0. \tag{A.8}$$

These identities are sufficient for every result below. In particular, for any projected record Ξ_ϕ,

$$0 \leq I(Y;\Xi_\phi \mid K) = H(Y \mid K) - H(Y \mid \Xi_\phi, K) \leq H(Y \mid K). \tag{A.9}$$

Therefore, whenever $0 < H(Y \mid K) < \infty$, $0 \leq \mathcal{C}_\phi(\pi) \leq 1$.

For a structured verdict $Y = (Y^{(1)}, \ldots, Y^{(R)})$, component-wise coverage is

$$\mathcal{C}_\phi^{(r)}(\pi) = \frac{I(Y^{(r)};\Xi_\phi \mid K)}{H(Y^{(r)} \mid K)}, \qquad H(Y^{(r)} \mid K) > 0. \tag{A.10}$$

This quantity supplements, rather than replaces, coverage of the joint verdict vector.

A.2 Proof of Proposition 1: Trace Projection Monotonicity

Proof. Assume $\Xi_{\phi_1} = g(\Xi_{\phi_2})$. Conditional on K, the variables form the Markov chain

$$Y \longrightarrow \Xi_{\phi_2} \longrightarrow \Xi_{\phi_1}. \tag{A.11}$$

Indeed,

$$p(\xi_{\phi_1} \mid y, \xi_{\phi_2}, k) = \mathbf{1}\{\xi_{\phi_1} = g(\xi_{\phi_2})\} = p(\xi_{\phi_1} \mid \xi_{\phi_2}, k). \tag{A.12}$$

Applying conditional data processing gives

$$I(Y;\Xi_{\phi_1} \mid K) \leq I(Y;\Xi_{\phi_2} \mid K). \tag{A.13}$$

Because $H(Y \mid K) > 0$, division by the same positive normalizer yields

$$\frac{I(Y;\Xi_{\phi_1} \mid K)}{H(Y \mid K)} \leq \frac{I(Y;\Xi_{\phi_2} \mid K)}{H(Y \mid K)}, \tag{A.14}$$

which is exactly $\mathcal{C}_{\phi_1}(\pi) \leq \mathcal{C}_{\phi_2}(\pi)$. □

Scope of the result. The proof requires the same Y, the same K, and a deterministic nesting relationship between the two projections. It does not compare evaluations with different target variables or task distributions.

A.3 Proof of Proposition 2: Chronological Trace Decomposition

Proof. Introduce an ordered sequence of trace variables

$$V_0 = S_0, \qquad V_{2t-1} = M_t, \qquad V_{2t} = X_t, \qquad t = 1, \ldots, H. \tag{A.15}$$

Then

$$\mathcal{T}_H^\pi = (V_0, V_1, \ldots, V_{2H}). \tag{A.16}$$

Applying the conditional chain rule in chronological order gives

$$I(Y; \mathcal{T}_H^\pi \mid K) = \sum_{i=0}^{2H} I(Y; V_i \mid K, V_{<i}). \tag{A.17}$$

The $i = 0$ term is

$$I(Y; V_0 \mid K) = I(Y; S_0 \mid K). \tag{A.18}$$

For $V_{2t-1} = M_t$, the variables preceding M_t are $S_0, M_{<t}, X_{<t}$, hence

$$I(Y; V_{2t-1} \mid K, V_{<2t-1}) = I(Y; M_t \mid K, S_0, M_{<t}, X_{<t}). \tag{A.19}$$

For $V_{2t} = X_t$, the variables preceding X_t are $S_0, M_{\le t}, X_{<t}$, hence

$$I(Y; V_{2t} \mid K, V_{<2t}) = I(Y; X_t \mid K, S_0, M_{\le t}, X_{<t}). \tag{A.20}$$

Substituting Equations (A.18)– (A.20) into Equation (A.17) yields

$$\begin{aligned} I(Y; \mathcal{T}_H^\pi \mid K) = I(Y; S_0 \mid K) \\ + \sum_{t=1}^{H} I(Y; M_t \mid \mathcal{H}_t^-) \\ + \sum_{t=1}^{H} I(Y; X_t \mid \mathcal{H}_t^+). \end{aligned} \tag{A.21}$$

This is the claimed decomposition. □

Externally auditable special case. If no controller state is recorded, let $M_t \equiv \varnothing$ be constant. Then

$$I(Y; M_t \mid K, S_0, M_{<t}, X_{<t}) = 0, \tag{A.22}$$

and conditioning on $M_{\le t}$ changes no distribution. Therefore,

$$I(Y; S_0, X_{1:H} \mid K) = I(Y; S_0 \mid K) + \sum_{t=1}^{H} I(Y; X_t \mid K, S_0, X_{<t}). \tag{A.23}$$

Why future controller states are excluded. Conditioning the contribution of X_t on $M_{1:H}$ would include controller states produced after X_t. Although a chain-rule identity could be obtained under another variable ordering, that expression would not represent the chronological information available at step t. The ordering above matches the execution process and avoids future-state leakage in the interpretation.

A.4 Proof of Proposition 3: Call-Only Coverage Bound

Proof. By definition, $\Xi_{\text{call}} = A_{1:H} = \phi_{\text{call}}(\mathcal{T}_H^\pi)$ is a deterministic projection of the complete trace. Conditional on K,

$$Y \longrightarrow \mathcal{T}_H^\pi \longrightarrow \Xi_{\text{call}} \tag{A.24}$$

is a Markov chain. Conditional data processing gives

$$I(Y; \Xi_{\text{call}} \mid K) \le I(Y; \mathcal{T}_H^\pi \mid K). \tag{A.25}$$

The call sequence is also a deterministic function of the external-interaction sequence, because A_t is a component of X_t. Therefore, conditional on (K, S_0),

$$Y \longrightarrow X_{1:H} \longrightarrow \Xi_{\text{call}} \tag{A.26}$$

is a Markov chain, which yields

$$I(Y; \Xi_{\text{call}} \mid K, S_0) \leq I(Y; X_{1:H} \mid K, S_0). \tag{A.27}$$

Both claimed inequalities follow. □

Equality condition. For the first inequality, equality holds exactly when $I(Y; \mathcal{T}_H^\pi \mid \Xi_{\text{call}}, K) = 0$. Thus, the proposition does not claim that every call-only record is strictly lossy; it states the maximum information ordering and gives a condition for no loss.

A.5 Proof of Proposition 4: No Universal Ordering for Independent Calls

Fix $K = k$ and $S_0 = s_0$ throughout the two constructions below. Conditional notation is omitted only to simplify the display.

Redundancy: $\mathcal{I}_{\text{ind}} > \mathcal{I}_{\text{seq}}$. Let $H = 2$, let Y be a fair Bernoulli variable, and set

$$X_1 = Y, \qquad X_2 = Y. \tag{A.28}$$

Then

$$\mathcal{I}_{\text{ind}} = I(Y; X_1) + I(Y; X_2) = H(Y) + H(Y) = 2H(Y). \tag{A.29}$$

By the chain rule,

$$\begin{aligned} \mathcal{I}_{\text{seq}} &= I(Y; X_1, X_2) \\ &= I(Y; X_1) + I(Y; X_2 \mid X_1) \\ &= H(Y) + 0 = H(Y). \end{aligned} \tag{A.30}$$

Hence $\mathcal{I}_{\text{ind}} = 2H(Y) > H(Y) = \mathcal{I}_{\text{seq}}$. The independent sum counts the same evidence twice.

Synergy: $\mathcal{I}_{\text{ind}} < \mathcal{I}_{\text{seq}}$. Let X_1 and X_2 be independent fair Bernoulli variables and define

$$Y = X_1 \oplus X_2. \tag{A.31}$$

For either $t \in \{1, 2\}$, observing X_t leaves Y uniform, so

$$I(Y; X_1) = I(Y; X_2) = 0. \tag{A.32}$$

Therefore,

$$\mathcal{I}_{\text{ind}} = 0. \tag{A.33}$$

But (X_1, X_2) determines Y, and hence

$$\mathcal{I}_{\text{seq}} = I(Y; X_1, X_2) = H(Y) > 0. \tag{A.34}$$

Thus $\mathcal{I}_{\text{ind}} < \mathcal{I}_{\text{seq}}$.

Because valid distributions realize both strict directions, no universal ordering exists between the two quantities.

Correction to a termwise equality claim. Equality of the two sums does not require equality of every corresponding summand; positive and negative differences can cancel. The universally valid statement is the no-ordering result above. A termwise condition $I(Y; X_t \mid K, S_0) = I(Y; X_t \mid K, S_0, X_{<t})$ for every t is sufficient, but not necessary, for equality.

A.6 Proof of Proposition 5: Equivalent Forms of Trace Sufficiency

Proof. Trace sufficiency in the main paper is

$$I(Y;\Xi_\phi \mid K) = I(Y;\mathcal{T}_H^\pi \mid K). \tag{A.35}$$

Using $I(Y;Z \mid K) = H(Y \mid K) - H(Y \mid Z,K)$ on both sides,

$$\begin{aligned} &H(Y \mid K) - H(Y \mid \Xi_\phi, K) \\ &\quad = H(Y \mid K) - H(Y \mid \mathcal{T}_H^\pi, K). \end{aligned} \tag{A.36}$$

Cancelling $H(Y \mid K)$ gives

$$H(Y \mid \Xi_\phi, K) = H(Y \mid \mathcal{T}_H^\pi, K). \tag{A.37}$$

This proves equivalence of the first two statements.

Because Ξ_ϕ is a deterministic function of $\mathcal{T}_H^\pi$,

$$I(Y;\Xi_\phi \mid \mathcal{T}_H^\pi, K) = 0. \tag{A.38}$$

Apply the chain rule to the pair $(\mathcal{T}_H^\pi, \Xi_\phi)$ in two orders:

$$\begin{aligned} I(Y;\mathcal{T}_H^\pi,\Xi_\phi \mid K) &= I(Y;\Xi_\phi \mid K) + I(Y;\mathcal{T}_H^\pi \mid \Xi_\phi, K), \\ I(Y;\mathcal{T}_H^\pi,\Xi_\phi \mid K) &= I(Y;\mathcal{T}_H^\pi \mid K) + I(Y;\Xi_\phi \mid \mathcal{T}_H^\pi, K) \\ &= I(Y;\mathcal{T}_H^\pi \mid K). \end{aligned} \tag{A.39}$$

Equating the two expressions gives

$$I(Y;\mathcal{T}_H^\pi \mid K) - I(Y;\Xi_\phi \mid K) = I(Y;\mathcal{T}_H^\pi \mid \Xi_\phi, K). \tag{A.40}$$

The left side is zero exactly when the right side is zero. Therefore, all three statements are equivalent. □

A.7 Proof of Proposition 6: Incremental Coverage of a Restored Component

Proof. Let $\Xi_{\phi,+j} = (\Xi_\phi, Z_j)$. By the conditional chain rule,

$$I(Y;\Xi_\phi, Z_j \mid K) = I(Y;\Xi_\phi \mid K) + I(Y;Z_j \mid \Xi_\phi, K). \tag{A.41}$$

Subtract $I(Y;\Xi_\phi \mid K)$ and divide by $H(Y \mid K) > 0$:

$$\begin{aligned} \mathcal{C}_{\phi,+j}(\pi) - \mathcal{C}_\phi(\pi) &= \frac{I(Y;\Xi_\phi, Z_j \mid K) - I(Y;\Xi_\phi \mid K)}{H(Y \mid K)} \\ &= \frac{I(Y;Z_j \mid \Xi_\phi, K)}{H(Y \mid K)}. \end{aligned} \tag{A.42}$$

Conditional mutual information is non-negative, so the gain is non-negative. It equals zero exactly when $I(Y;Z_j \mid \Xi_\phi, K) = 0$. □

Why marginal relevance is insufficient. It is possible that $I(Y;Z_j \mid K) = 0$ while $I(Y;Z_j \mid \Xi_\phi, K) > 0$, as in an XOR-type synergy. Conversely, a field with positive marginal mutual information can be redundant after another retained field is observed. Conditional gain is therefore the appropriate quantity for field restoration.

A.8 Derivation of Corollary 1: Chronological Form of Projection Coverage

Proof. By definition of projection fidelity,

$$\rho_\pi(\phi) = \frac{I(Y;\Xi_\phi \mid K)}{I(Y;\mathcal{T}_H^\pi \mid K)}. \tag{A.43}$$

Rearranging,

$$I(Y;\Xi_\phi \mid K) = \rho_\pi(\phi) I(Y;\mathcal{T}_H^\pi \mid K). \tag{A.44}$$

Divide by $H(Y \mid K) > 0$:

$$\mathcal{C}_\phi(\pi) = \rho_\pi(\phi) \frac{I(Y; \mathcal{T}_H^\pi \mid K)}{H(Y \mid K)}. \tag{A.45}$$

From Proposition 2,

$$\begin{aligned} I(Y; \mathcal{T}_H^\pi \mid K) = I(Y; S_0 \mid K) \\ + \sum_{t=1}^{H} I(Y; M_t \mid K, S_0, M_{<t}, X_{<t}) \\ + \sum_{t=1}^{H} I(Y; X_t \mid K, S_0, M_{\le t}, X_{<t}). \end{aligned} \tag{A.46}$$

Substituting Equation (A.46) into Equation (A.45) gives

$$\begin{aligned} \mathcal{C}_\phi(\pi) = \rho_\pi(\phi) \frac{1}{H(Y \mid K)} \Bigg[I(Y; S_0 \mid K) \\ + \sum_{t=1}^{H} I(Y; M_t \mid \mathcal{H}_t^-) \\ + \sum_{t=1}^{H} I(Y; X_t \mid \mathcal{H}_t^+) \Bigg], \end{aligned} \tag{A.47}$$

which is the corollary. □

Externally auditable instantiation. For $M_t \equiv \varnothing$, the controller-state terms are zero and Equation (A.47) reduces to

$$\mathcal{C}_\phi(\pi) = \rho_\pi(\phi) \frac{I(Y; S_0 \mid K) + \sum_{t=1}^{H} I(Y; X_t \mid K, S_0, X_{<t})}{H(Y \mid K)}. \tag{A.48}$$

This is the form operationalized by the reported experiments.

A.9 Proof of Theorem 1: Coverage Gap

Proof. Because $\Xi_\phi = \phi(\mathcal{T}_H^\pi)$ is deterministic,

$$I(Y; \Xi_\phi \mid \mathcal{T}_H^\pi, K) = 0. \tag{A.49}$$

Expand the mutual information of the pair $(\mathcal{T}_H^\pi, \Xi_\phi)$ in two orders:

$$\begin{aligned} I(Y; \mathcal{T}_H^\pi, \Xi_\phi \mid K) &= I(Y; \Xi_\phi \mid K) + I(Y; \mathcal{T}_H^\pi \mid \Xi_\phi, K), \\ I(Y; \mathcal{T}_H^\pi, \Xi_\phi \mid K) &= I(Y; \mathcal{T}_H^\pi \mid K) + I(Y; \Xi_\phi \mid \mathcal{T}_H^\pi, K) \\ &= I(Y; \mathcal{T}_H^\pi \mid K). \end{aligned} \tag{A.50}$$

Therefore,

$$I(Y; \mathcal{T}_H^\pi \mid K) - I(Y; \Xi_\phi \mid K) = I(Y; \mathcal{T}_H^\pi \mid \Xi_\phi, K). \tag{A.51}$$

Divide by $H(Y \mid K) > 0$:

$$\mathcal{C}_{\text{full}}(\pi) - \mathcal{C}_\phi(\pi) = \frac{I(Y; \mathcal{T}_H^\pi \mid \Xi_\phi, K)}{H(Y \mid K)}. \tag{A.52}$$

□

Deterministic-oracle corollary. If $Y = G_{\Omega_Q}(\mathcal{T}_H^{\pi,\mathrm{ext}}, K)$, then $\mathcal{T}_H^{\pi,\mathrm{ext}}$ is a deterministic projection of $\mathcal{T}_H^{\pi}$, and hence

$$H(Y \mid \mathcal{T}_H^{\pi}, K) = 0. \tag{A.53}$$

Using $I(Y; \mathcal{T}_H^{\pi} \mid K) = H(Y \mid K) - H(Y \mid \mathcal{T}_H^{\pi}, K)$, we obtain

$$\mathcal{C}_{\mathrm{full}}(\pi) = 1. \tag{A.54}$$

Also,

$$\mathcal{C}_{\phi}(\pi) = 1 - \frac{H(Y \mid \Xi_{\phi}, K)}{H(Y \mid K)}. \tag{A.55}$$

Therefore,

$$1 - \mathcal{C}_{\phi}(\pi) = \frac{H(Y \mid \Xi_{\phi}, K)}{H(Y \mid K)}. \tag{A.56}$$

A.10 Proof of Proposition 7: Projection Ambiguity and Bayes Risk

Let an evaluator use a decision rule $\delta : \mathcal{K} \times \mathcal{X}_{\phi} \to \mathcal{Y}$ and incur 0–1 loss. Conditional on $(K = k, \Xi_{\phi} = \xi)$, its error probability is

$$\begin{aligned} P(\delta(k,\xi) \neq Y \mid k, \xi) &= 1 - P(Y = \delta(k,\xi) \mid k, \xi) \\ &\geq 1 - \max_{y} P(Y = y \mid k, \xi). \end{aligned} \tag{A.57}$$

Equality is attained by the maximum-a-posteriori rule

$$\delta^{\star}(k,\xi) \in \arg\max_{y} P(Y = y \mid k, \xi). \tag{A.58}$$

Averaging the optimal conditional error over (K, Ξ_{ϕ}) gives

$$R_{\phi}^{\star} = \mathbb{E}\left[1 - \max_{y} P(Y = y \mid K, \Xi_{\phi})\right]. \tag{A.59}$$

On the ambiguity region, $\max_y P(Y = y \mid K, \Xi_{\phi}) < 1$. If that region has positive probability, its strictly positive conditional error integrates to $R_{\phi}^{\star} > 0$.

This argument applies to deterministic or randomized trace variables and does not require individual traces to have positive probability mass.

A.11 Detailed Derivation of Pointwise Benchmark Under-Coverage

Let

$$Z_{\mathrm{pt}} = \psi(\mathcal{T}_H^{\pi}) \tag{A.60}$$

be a pointwise record and suppose it is evaluated against the same verdict Y and context K as the complete trace. Because Z_{pt} is a deterministic function of $\mathcal{T}_H^{\pi}$,

$$Y \longrightarrow \mathcal{T}_H^{\pi} \longrightarrow Z_{\mathrm{pt}} \qquad \text{conditional on } K. \tag{A.61}$$

Conditional data processing yields

$$I(Y; Z_{\mathrm{pt}} \mid K) \leq I(Y; \mathcal{T}_H^{\pi} \mid K). \tag{A.62}$$

After division by $H(Y \mid K) > 0$,

$$\mathcal{C}_{\mathrm{pt}}(\pi) \leq \mathcal{C}_{\mathrm{full}}(\pi). \tag{A.63}$$

For the exact gap, expand the pair $(\mathcal{T}_H^{\pi}, Z_{\mathrm{pt}})$ in two orders:

$$\begin{aligned} I(Y; \mathcal{T}_H^{\pi}, Z_{\mathrm{pt}} \mid K) &= I(Y; Z_{\mathrm{pt}} \mid K) + I(Y; \mathcal{T}_H^{\pi} \mid Z_{\mathrm{pt}}, K), \\ I(Y; \mathcal{T}_H^{\pi}, Z_{\mathrm{pt}} \mid K) &= I(Y; \mathcal{T}_H^{\pi} \mid K) + I(Y; Z_{\mathrm{pt}} \mid \mathcal{T}_H^{\pi}, K). \end{aligned} \tag{A.64}$$

Determinism gives

$$I(Y; Z_{\mathrm{pt}} \mid \mathcal{T}_H^{\pi}, K) = 0. \tag{A.65}$$

Therefore,

$$I(Y; \mathcal{T}_H^{\pi} \mid K) - I(Y; Z_{\mathrm{pt}} \mid K) = I(Y; \mathcal{T}_H^{\pi} \mid Z_{\mathrm{pt}}, K). \tag{A.66}$$

Normalizing yields

$$\mathcal{C}_{\mathrm{full}}(\pi) - \mathcal{C}_{\mathrm{pt}}(\pi) = \frac{I(Y; \mathcal{T}_H^{\pi} \mid Z_{\mathrm{pt}}, K)}{H(Y \mid K)}. \tag{A.67}$$

The gap is positive exactly when the omitted trace retains additional verdict information after the pointwise observation and common context are known.

Table 3: Comparison of ATT with representative benchmark families. Traj. denotes multi-step trajectory representation; Seq. denotes prefix-conditioned sequential dependency; Obs. denotes observations or state changes; and X-tool denotes cross-tool dependency.

Framework type	Stateful	Traj.	Seq.	Obs.	X-tool	Oracle
ATT	✓	✓	✓	✓	✓	✓
Function-call eval.	✗	△	✗	✗	△	△
API/tool-use eval.	✗	△	✗	✗	△	△
Stateful agent eval.	✓	✓	△	✓	✓	✓
MCP/live-backend eval.	△	✓	△	△	✓	△

✓ denotes explicit modeling; ✗ denotes largely absent modeling; and △ denotes partial or task-specific coverage.

Different local verdicts are not ordered by this result. If the pointwise benchmark predicts a different target Y_{pt}—for example, schema validity—then $I(Y_{\mathrm{pt}}; Z_{\mathrm{pt}} \mid K)$ and $I(Y; \mathcal{T}_H^{\pi} \mid K)$ concern different random variables. No data- processing ordering follows between their normalized scores. This is why ATT separates representation coverage from empirical benchmark accuracy.

A.12 Comparison with Existing Benchmark Families

Table 3 summarizes the trace evidence retained by representative benchmark families. The comparison concerns evaluation representations rather than overall benchmark quality or task difficulty.

B Framework Implementation Details

The implementation maps ATT to an executable audit pipeline. It separates the agent policy, state-aware tool execution, trace recording, and trajectory-level evaluation. This section specifies that mapping without altering the reported experimental results. Appendix B.1–B.7 provides the authoritative operational definitions used by the main paper: the ATT mapping, TraceState representation, tool contract, execution procedure, backend semantics, perturbation process, and trace-level evaluation, respectively. Later sections refer to these definitions rather than restating them.

B.1 Implementation Overview and ATT Mapping

At each turn, the policy proposes an action. The executor applies the action to a persistent state under configured backend b_t, and the recorder constructs the realized execution-condition record B_t together with the evidence needed to reconstruct the auditable trace. We use *TraceState* throughout for this shared persistent state representation.

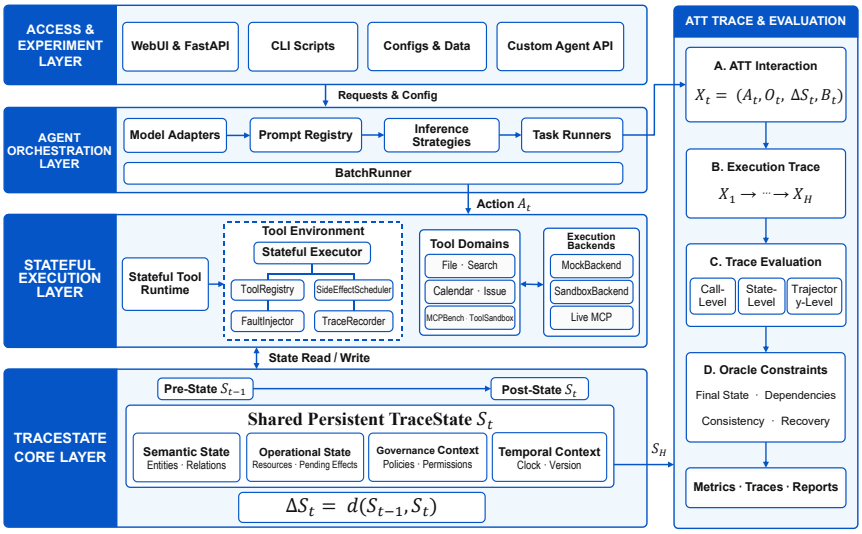

Table 4: Operational mapping of the principal ATT objects.

ATT object	Operational representation	Audit role
K	Task specification, oracle specification, tool schemas, and disclosed context	Conditions shared across evaluators
S_t	TraceState with paired state snapshots and structured transition records	Persistent environment state
A_t	Tool name and structured arguments	Externally observable action
O_t	Returned observation, status, and error	Execution feedback
ΔS_t	Ordered environment-ready and current-action transition records	Auditable state transition with attribution
B_t	Realized backend, schema, latency, timeout, status, and fault metadata	Execution-condition evidence
$\mathcal{T}_H^{\pi,\mathrm{ext}}$	Initial state and append-only interaction records	Full externally auditable trace
Ω_Q, Y	Task-level predicates and the resulting trajectory verdict	Evaluation target
Ξ_ϕ	Deterministic fields selected from the recorded trace	Evaluator-visible projection

Figure 5: ATT-guided evaluation pipeline. Stateful execution records canonical interactions over persistent TraceState, from which multiple evaluation projections are constructed under a shared trajectory-level oracle.

The framework records the variables required to construct projections from the same underlying trace. It does not estimate the mutual-information quantities in ATT.

B.2 TraceState Representation and State Transitions

TraceState is the persistent environment state shared by all tools in an episode. Its structured representation is

$$S_t = \big(E_t^{\mathrm{ent}}, E_t^{\mathrm{rel}}, E_t^{\mathrm{res}}, P_t, \tau_t, V_t, F_t^{\mathrm{pend}}\big). \tag{B.1}$$

Here, E_t^{ent} stores typed entities, including files, calendar events, issues, contacts, messages, settings, search-index entries, and MCP results. The relation and resource stores are E_t^{rel} and E_t^{res}. Policies and permissions are represented by P_t. The environment clock τ_t and pending-effect queue F_t^{pend} represent delayed visibility, and V_t records state revisions.

All tools in a stateful episode receive the same TraceState instance. Let S_t^{pre} be the state after the environment clock advances and ready effects from earlier actions are applied, but before A_t executes. The framework serializes the start, prepared, and post-action states and constructs

$$\begin{aligned} \Delta S_t^{\mathrm{env}} &= d\big(S_{t-1}^{\mathrm{snap}}, S_t^{\mathrm{pre,snap}}\big), \\ \Delta S_t^{\mathrm{act}} &= d(S_t^{\mathrm{pre,snap}}, S_t^{\mathrm{snap}}), \\ \Delta S_t &= \big(\Delta S_t^{\mathrm{env}}, \Delta S_t^{\mathrm{act}}\big). \end{aligned} \tag{B.2}$$

Each component retains paired snapshot references, stable hashes, and a structured diff. This separates effects that became ready at the step boundary from state changes attributable to the current action. Rollback is an available executor operation, not an assumed agent recovery mechanism.

Agent-visible state view. At the beginning of step t, the executor first advances the environment and applies ready pending effects to obtain S_t^{pre}. Before selecting action A_t, the policy receives the task, available tool schemas, the recorded prefix $X_{<t}$, and

$$\Sigma_t = \sigma(S_t^{\mathrm{pre}}), \tag{B.3}$$

where σ returns a compact summary of entity counts, resources, the environment clock, state version, and pending effects. The summary is a policy input rather than a trajectory verdict. Oracle satisfaction and hidden evaluator checks are not components of Σ_t.

B.3 Tool Interface and Execution Contract

Each tool implements a common state-aware contract. `ToolMetadata` specifies its name, version, domain, schemas, permissions, idempotency, and support for partial or asynchronous execution. Optional preconditions and postconditions express state-dependent validity.

At invocation, the executor creates an `ExecutionContext`. It contains the shared state, call identifier, clock, permissions, timeout, attempt counters, environment, and backend. A `ToolExecutionResult` contains the execution status, observation, error, state-change declaration, pending status, scheduled effects, and result metadata. This contract separates call admissibility, execution success, and trajectory-level task success.

B.4 Stateful Execution and Trace Construction

Algorithm 1 converts one proposed action into an append-only trace record. All variables used by failure branches are initialized before validation, and postconditions are checked only after a resolved tool executes.

Algorithm 1 Stateful execution and trace construction

Require: Policy π, task context K, prefix $X_{<t}$, state S_{t-1}, environment $\mathcal{E}$, configured backend b_t
Ensure: Record X_t and updated state S_t

$F_t \leftarrow \varnothing$; $q_t^+ \leftarrow \bot$; $e_t \leftarrow$ false
$(S^{\text{start}}, h^{\text{start}}) \leftarrow$ SnapshotAndHash(S_{t-1})
$S_t^{\text{pre}} \leftarrow$ PrepareEnvironment$(S_{t-1}, \mathcal{E})$ ▷ advance time and apply ready effects
$(S^{\text{pre}}, h^{\text{pre}}) \leftarrow$ SnapshotAndHash(S_t^{pre})
$\Sigma_t \leftarrow \sigma(S_t^{\text{pre}})$
$A_t = (u_t, \mathbf{a}_t) \sim \pi(\cdot \mid K, X_{<t}, \Sigma_t)$
$(\widetilde{b}_t, f_t^-) \leftarrow$ ApplyPreCallPerturbation$(b_t, u_t, \mathbf{a}_t)$
$(T_t, v_t, q_t^-) \leftarrow$ ResolveAndValidate$(u_t, \mathbf{a}_t, S_t^{\text{pre}}, \widetilde{b}_t)$
if $\neg v_t$ **or** Blocks(f_t^-) **then**
    $E_t^{\text{tool}} \leftarrow$ FailureResult(q_t^-, f_t^-)
    $O_t^{\text{raw}} \leftarrow$ NormalizeObservation(E_t^{tool})
    $O_t \leftarrow O_t^{\text{raw}}$; $f_t^+ \leftarrow \varnothing$
else
    $E_t^{\text{tool}} \leftarrow$ Execute$(T_t, \mathbf{a}_t, S_t^{\text{pre}}, \widetilde{b}_t)$
    $F_t \leftarrow$ ScheduleAndApplyEffects$(E_t^{\text{tool}}, \mathcal{E})$
    $O_t^{\text{raw}} \leftarrow$ NormalizeObservation(E_t^{tool})
    $(O_t, f_t^+) \leftarrow$ ApplyPostCallPerturbation$(u_t, O_t^{\text{raw}}, b_t)$
    $e_t \leftarrow$ true
end if
$S_t \leftarrow$ CurrentState$(\mathcal{E})$
$(S^{\text{post}}, h^{\text{post}}) \leftarrow$ SnapshotAndHash(S_t)
$\Delta S_t^{\text{env}} \leftarrow$ TransitionRecord$(S^{\text{start}}, S^{\text{pre}}, h^{\text{start}}, h^{\text{pre}})$
$\Delta S_t^{\text{act}} \leftarrow$ TransitionRecord$(S^{\text{pre}}, S^{\text{post}}, h^{\text{pre}}, h^{\text{post}})$
$\Delta S_t \leftarrow (\Delta S_t^{\text{env}}, \Delta S_t^{\text{act}})$
if e_t **then**
    $q_t^+ \leftarrow$ CheckPostconditions$(T_t, S_t, \Delta S_t^{\text{act}})$
end if
$B_t \leftarrow$ BuildExecutionCondition$(b_t, \widetilde{b}_t, q_t^-, q_t^+, f_t^-, f_t^+, F_t, O_t^{\text{raw}})$
$X_t \leftarrow$ BuildRecord$(A_t, O_t, \Delta S_t, B_t)$
Append$(\mathcal{T}, X_t)$
return X_t, S_t

The resulting record is

$$X_t = (A_t, O_t, \Delta S_t, B_t). \tag{B.4}$$

Here, configured backend b_t is an execution input, whereas B_t is the realized record built after the attempt. It retains the configured and active backend identifiers, schema version, execution duration, precondition outcome q_t^-, postcondition outcome q_t^+, fault records f_t^- and f_t^+, scheduled effects F_t, raw audit-only observation O_t^{raw}, timeout status, and retry metadata. All auxiliary execution metadata passed to the recorder is therefore represented as part of B_t, so the canonical ATT interaction remains $X_t = (A_t, O_t, \Delta S_t, B_t)$. Appending records without overwriting earlier interactions produces

$$\mathcal{T}_t^{\pi,\text{ext}} = (S_0, X_{1:t}). \tag{B.5}$$

Table 5: Backend conditions in the migration study.

Backend	Operational semantics	Experimental role
Mock	Permissive in-memory execution	Controlled baseline
Sandbox	Strict isolated execution with session-scoped artifacts	Stricter backend condition
Live MCP	Live MCP execution with responses normalized into TraceState	Live backend migration

Table 6: Perturbations used in the robustness evaluation. Stale-state feedback replays an earlier observation from the same tool.

Perturbation	Injection stage	Tool executed	Effect on the recorded interaction
Timeout	Before call	No	Returns a timeout failure without applying the tool action
Schema drift	Before call	No	Rejects the supplied arguments as incompatible with the active schema
Stale-state feedback	After call	Yes	Replays a previous observation while retaining the current execution outcome
Vague observation	After call	Yes	Replaces the original observation with low-information feedback
Misleading observation	After call	Yes	Replaces the original observation with configured but incorrect evidence

ATT separates execution termination from verdict computation. A policy-issued finish action or the interaction budget ends an ATT-compatible episode, after which the evaluator computes Y. Any protocol that stops on oracle satisfaction or rejects finish actions supplies oracle-derived control and must be identified as a separate evaluation condition.

B.5 Backend Semantics and Deferred Side Effects

Backend changes preserve the task specification while changing the configured condition b_t and, consequently, the distribution of realized records B_t. Table 5 summarizes the three configured conditions.

Successful MCP responses are normalized as typed state entities. This mapping supports a common verdict interface across backend conditions.

Deferred effects are managed by an environment-level scheduler. A tool may update TraceState immediately while scheduling a related change for a later environment time. For example, a file update can precede the corresponding search-index update:

$$F_t^{\mathrm{pend}} \xrightarrow{\tau_t \geq \tau_{\mathrm{ready}}} F_{t+1}^{\mathrm{pend}}, \qquad S_t \to S_{t+1}. \tag{B.6}$$

Delayed re-indexing is therefore a configured effect, not an automatic consequence of every file update. At termination, the framework applies only effects whose ready time is no later than the current environment time. Future pending effects remain in F_H^{pend} and are available to the oracle; the evaluator does not flush future effects automatically. This terminalization policy is held fixed across backends and prompting strategies.

B.6 Perturbation Injection

Perturbation profiles distinguish conditions that prevent execution from those that preserve the action outcome but alter the observation returned to the agent. The latter can leave the environment changed while exposing unreliable feedback.

Pre-call timeout and schema-drift perturbations do not execute the tool. Because schema drift changes the active schema, it is applied before resolution and argument validation:

$$(\widetilde{b}_t, f_t^-) \sim \kappa_{\mathrm{pre}}(\cdot \mid b_t, A_t, S_t^{\mathrm{pre}}). \tag{B.7}$$

Post-call perturbations preserve the underlying execution result and state update, but replace the observation delivered to the agent:

$$\widetilde{O}_t \sim \kappa_{\text{post}}(\cdot \mid O_t^{\text{raw}}, b_t)\,. \tag{B.8}$$

The recorded O_t denotes the agent-visible observation. When κ_{post} is active, $O_t = \widetilde{O}_t$, while the unmodified feedback O_t^{raw} is retained only in B_t for audit. Each perturbed trajectory receives at most one configured perturbation. The target step is selected from calls for which the perturbation is applicable; stale state additionally requires an earlier observation from the same tool ($\exists j < t : u_j = u_t$). Vague and misleading observations are template-based and schema-compatible. Timeouts consume the attempted action budget, whereas schema drift is scoped to the perturbed call and the nominal schema is restored for subsequent steps. This is a distributional representation rather than a causal claim.

B.7 Trace-Level Oracle and Metric Computation

The task-level constraints Ω_Q are fixed before execution. After termination, the evaluator computes

$$\begin{aligned} Y &= G_{\Omega_Q}\big(\mathcal{T}_H^{\pi,\text{ext}}, K\big) \\ &= g_{\Omega_Q}(S_H, D_{1:H}, \Gamma_{1:H}; K)\,. \end{aligned} \tag{B.9}$$

Here, $D_{1:H}$ contains dependency-satisfaction indicators and $\Gamma_{1:H}$ contains consistency, permission, and recovery predicates. The second line is an implementation factorization of the same trace-level oracle. The verdict may contain final-state, dependency, permission, consistency, and recovery components. Dependency predicates encode semantic task requirements rather than one reference trajectory. Multiple action orders are valid when they satisfy the same preconditions and required state transitions.

The implementation reports the metric family

$$M = \big(M_{\text{call}}, M_{\text{state}}, M_{\text{dep}}, M_{\text{rec}}, M_{\text{cost}}\big). \tag{B.10}$$

These metrics operationalize different verdict components but do not estimate $I(Y; \Xi_\phi \mid K)$ or the component-wise quantities in Equation (A.10). Call metrics describe local execution status. State metrics evaluate terminal predicates, dependency metrics evaluate trajectory constraints, and recovery metrics require a defined recovery opportunity. Cost metrics summarize trace length, token use, latency, or retries. Unless otherwise stated, aggregate rates are computed by pooling the underlying task-condition records within the reported group; recovery is pooled over eligible recovery episodes rather than macro-averaging displayed strategy percentages.

Reported execution metrics. For a reported group $\mathcal{G}$ of $N = |\mathcal{G}|$ trajectories, let $g_{\text{state}}(S_{H_i}; K_i)$ denote satisfaction of the task-specific terminal-state predicates. Final-State Accuracy is

$$\text{FSA} = \frac{1}{N} \sum_{i \in \mathcal{G}} \mathbf{1}[g_{\text{state}}(S_{H_i}; K_i) = 1]\,. \tag{B.11}$$

State Acc. in the robustness experiment is the same metric and uses the same definition; the shorter label distinguishes the table context rather than a different estimator. Success@1 additionally requires the complete trajectory-level verdict on the first sampled execution.

Let D_i indicate satisfaction of all dependency and ordering predicates for trajectory i. Dependency Correctness is

$$\text{DC} = \frac{1}{N} \sum_{i \in \mathcal{G}} \mathbf{1}[g_{\text{state}}(S_{H_i}; K_i) = 1 \wedge D_i = 1]\,. \tag{B.12}$$

For the designated prerequisite-tool requirements $\mathcal{U}_i^{\text{req}}$, let $c_i^{\text{tool}} \in [0, 1]$ be the fraction present in the executed trajectory, with multiplicity retained when a tool is required more than once. Required-Tool Compliance is

$$\text{RTC} = \frac{1}{N} \sum_{i \in \mathcal{G}} c_i^{\text{tool}}. \tag{B.13}$$

It measures prerequisite coverage but does not enforce their order or the terminal-state predicates; those requirements enter Dependency Correctness.

Average Trace Length and Avg. Tokens are

$$\mathrm{ATL} = \frac{1}{N}\sum_{i\in\mathcal{G}} H_i, \qquad \mathrm{ATok} = \frac{1}{N}\sum_{i\in\mathcal{G}}\sum_{\ell=1}^{L_i}\left(p_{i,\ell}+c_{i,\ell}\right), \tag{B.14}$$

where H_i counts every external interaction appended to the trace, including failed or blocked attempts. L_i is the number of model turns, and $p_{i,\ell}$ and $c_{i,\ell}$ are prompt and completion tokens. Model-level rows pool counts across strategies, whereas strategy-level rows pool counts across model backbones. Percentages are computed from the underlying records before rounding.

Reduced evaluators are generated as projections $\Xi_\phi = \phi(\mathcal{T}_H^{\pi,\mathrm{ext}})$ of the same recorded trace:

$$\begin{aligned} \Xi_{\mathrm{call}} &= A_{1:H}, & \Xi_{\mathrm{AO}} &= (A_{1:H}, O_{1:H}),\\ \Xi_{\mathrm{terminal}} &= S_H, & \Xi_{\mathrm{state}} &= (A_{1:H}, O_{1:H}, \Delta S_{1:H}),\\ \Xi_{\mathrm{full}} &= \mathcal{T}_H^{\pi,\mathrm{ext}}. \end{aligned} \tag{B.15}$$

All projections are constructed deterministically from the same trace.

Same-trace projection audit. The audit reuses complete traces from the dependency, robustness, and backend studies without rerunning the policy. The retained records comprise 4,640 traces grouped into 590 non-execution contexts K. Each context identifies the task and oracle scope, disclosed initial-state hash, and evaluation protocol; model outputs and trajectory verdicts are excluded from K. Table 7 locks the fields and deterministic verdict rule for each view. The Action–Observation view treats the success flag, returned error, and response payload shown in the policy history as O_t. Executor-only status, precondition and postcondition results, permissions, timing, fault metadata, and backend conditions remain in B_t.

Table 7: Field-locked projections and deterministic verdict rules in the same-trace audit. Every rule receives the same context K.

Projection	Retained view	Projected verdict rule
Call-only	$A_{1:H}$	At least one tool action is present, and every tool identifier and argument object is parseable.
Action–Observation	$(A_{1:H}, O_{1:H})$	The Call-only rule holds, and every agent-visible response reports success without an error or exception.
Outcome-only	S_H	The terminal-state predicates in Ω_Q are satisfied.
State-aware	$(A_{1:H}, O_{1:H}, \Delta S_{1:H})$	The Action–Observation and terminal-state rules hold, and paired state-transition markers are internally consistent.
Full audit	$(S_0, A_{1:H}, O_{1:H}, \Delta S_{1:H}, B_{1:H})$	The complete trajectory-level oracle G_{Ω_Q} is satisfied.

For trace i, let $Y_i \in \{0,1\}$ be the Full-audit verdict and $\widehat{Y}_{\phi,i}$ the verdict obtained from projection ϕ. False acceptance, false rejection, and total disagreement are unconditional proportions over the audited traces:

$$\begin{aligned} \mathrm{FA}_\phi &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\widehat{Y}_{\phi,i}=1,\, Y_i=0],\\ \mathrm{FR}_\phi &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\widehat{Y}_{\phi,i}=0,\, Y_i=1],\\ \mathrm{Dis}_\phi &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\widehat{Y}_{\phi,i}\neq Y_i] = \mathrm{FA}_\phi + \mathrm{FR}_\phi. \end{aligned} \tag{B.16}$$

The exact collision rate counts traces belonging to a verdict-conflicting equivalence class of (K, Ξ_ϕ):

$$\mathrm{Coll}_\phi = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\exists j:\; K_j = K_i,\; \Xi_{\phi,j} = \Xi_{\phi,i},\; Y_j \neq Y_i]\,. \tag{B.17}$$

Let $\mathcal{R}^+$ contain the available failure classes among state, dependency, recovery, and backend failures, and let $\mathcal{F}_r$ be the traces exhibiting failure class r. Trajectory-Failure Recall is the macro-average

$$\mathrm{TFR}_\phi = \frac{1}{|\mathcal{R}^+|} \sum_{r \in \mathcal{R}^+} \frac{\sum_{i \in \mathcal{F}_r} \mathbf{1}[\widehat{Y}_{\phi,i} = 0]}{|\mathcal{F}_r|}. \tag{B.18}$$

State and dependency failures follow their corresponding oracle components. Recovery failure denotes a triggered perturbation that remains unrecovered in an unsuccessful trajectory, while backend failure denotes an unsuccessful non-mock backend trajectory. Finally, relative cost measures serialized record size rather than model inference cost:

$$\mathrm{RelCost}_\phi = \frac{\mathbb{E}[\mathrm{bytes}(\Xi_\phi)]}{\mathbb{E}[\mathrm{bytes}(\Xi_{\mathrm{full}})]}. \tag{B.19}$$

These empirical diagnostics operationalize projection loss but do not estimate $I(Y; \Xi_\phi \mid K)$ or $\mathcal{C}_\phi(\pi)$ directly.

Execution modes and robustness metrics. In the stateful condition, all calls in an episode read from and write to the same TraceState instance defined in Section B.2 and follow the execution procedure in Algorithm 1. In the stateless condition, each call instead receives a fresh copy of the disclosed initial state, so changes from earlier calls are unavailable to later calls. The textual interaction history is preserved, and the terminal state is the post-state of the final executed call under the same trajectory-level oracle. Both conditions use the same task instructions, tool schemas, step budgets, and oracle predicates.

For robustness experiments, each task-condition is sampled once, so

$$\mathrm{Success@1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[Y_i^{(1)} = \mathrm{success}\right]. \tag{B.20}$$

Recovery is computed on the eligible set

$$\mathcal{E}_{\mathrm{rec}} = \left\{ i : \begin{array}{l} \text{the audit log marks a triggered perturbation,} \\ \text{the perturbation creates a recovery opportunity,} \\ \text{and subsequent actions can restore the terminal verdict} \end{array} \right\}, \tag{B.21}$$

with

$$M_{\mathrm{rec}} = \frac{\sum_{i \in \mathcal{E}_{\mathrm{rec}}} \mathbf{1}[\text{successful restoration}]}{|\mathcal{E}_{\mathrm{rec}}|}. \tag{B.22}$$

Oracle-triggered termination, when enabled, is applied uniformly across compared conditions. It can affect trace length, token consumption, and the opportunity for post-success actions, so cost-related quantities are interpreted under this stated protocol.

C Additional Experimental Results

This section reports the complete experimental results used to support the main findings. The appendix expands the main results along the relevant experimental dimensions while preserving the same metric definitions.

C.1 Experiment 1: Stateful vs. Stateless Execution

The metrics in Table 8 follow Section B.7. Final-State Acc. evaluates the terminal-state predicates, whereas Avg. Trace Length and Avg. Tokens are cost measures computed from the complete recorded trace.

C.2 Experiment 2: Cross-Tool Dependency

The metrics in Table 10 follow the trace-level oracle in Section B.7. Final-State Acc. evaluates terminal-state satisfaction, Dependency Correctness additionally enforces the trajectory constraints in $D_{1:H}$, and Required-Tool Comp. is the mean fraction of designated prerequisite-tool requirements present in each trace.

Table 8: Complete results for Experiment 1 under stateful and stateless execution using the same default closed-loop policy for both modes. Avg. Tokens denotes the average total LLM tokens per case.

Model	Mode	Final-State Acc.	Avg. Trace Length	Avg. Tokens
DeepSeek-V4-Flash	Stateful	83.0%	4.25	13,543
	Stateless	91.0%	2.62	6,246
GPT-OSS-120B	Stateful	78.0%	4.61	8,245
	Stateless	81.0%	3.50	5,553
Gemma-4-31B-it	Stateful	85.0%	4.04	6,808
	Stateless	84.0%	2.65	4,301
Qwen3.6-35B-A3B	Stateful	75.0%	4.32	8,236
	Stateless	85.0%	2.80	4,961

Table 9: Dependency-level construction used in Experiment 2. Examples illustrate the prerequisite structure rather than a unique reference trajectory.

Level	Criterion	Typical instance	Dependency predicate
L1	Single required tool with no cross-call dependency	Create or retrieve one record	Target state is reachable from one admissible operation
L2	Explicit prerequisite chain across tools	Retrieve an identifier before updating its record	Later operation must consume an entity, identifier, or value produced earlier
L3	Implicit state coupling or delayed side effect across tools	Update state and then act on a derived or asynchronously visible condition	The trajectory must satisfy hidden ordering, pending-effect, or cross-domain state constraints

Table 10: Complete marginal results for Experiment 2 across dependency levels. Panel (a) pools each model across prompting strategies, and panel (b) pools each strategy across model backbones. Dependency correctness requires final-state satisfaction and all prerequisite and ordering constraints; Required-Tool Comp. measures mean prerequisite-tool coverage.

(a) Model-level results

Model	Level	Final-State Acc.	Dependency Correctness	Required-Tool Comp.
DeepSeek-V4-Flash	L1	100.0%	100.0%	100.0%
	L2	86.5%	76.0%	86.5%
	L3	53.7%	37.9%	71.4%
GPT-OSS-120B	L1	84.4%	84.4%	100.0%
	L2	68.8%	59.4%	85.4%
	L3	46.0%	33.8%	69.5%
Gemma-4-31B-it	L1	90.6%	90.6%	100.0%
	L2	81.2%	70.8%	91.7%
	L3	53.3%	34.9%	68.1%
Qwen3.6-35B-A3B	L1	93.8%	93.8%	100.0%
	L2	78.1%	68.8%	85.4%
	L3	44.5%	30.1%	66.7%

(b) Strategy-level results

Strategy	Level	Final-State Acc.	Dependency Correctness	Required-Tool Comp.
Direct	L1	78.1%	78.1%	100.0%
	L2	53.1%	46.9%	80.2%
	L3	9.6%	7.7%	44.5%
CoT	L1	93.8%	93.8%	100.0%
	L2	80.2%	69.8%	89.6%
	L3	37.1%	22.1%	65.3%
ReAct	L1	96.9%	96.9%	100.0%
	L2	89.6%	75.0%	87.5%
	L3	69.9%	48.2%	81.1%
Self-Refine	L1	100.0%	100.0%	100.0%
	L2	91.7%	83.3%	91.7%
	L3	80.9%	58.8%	84.9%

C.3 Experiment 3: Robustness to Interaction Noise

Success@1, State Acc., and Recovery in Table 11 use the robustness definitions in Section B.7. Success@1 evaluates the first sampled perturbed trajectory. State Acc. is the same terminal-state metric as Final-State Acc. in Equation (B.11), and Recovery is computed only over eligible episodes with a triggered recovery opportunity.

C.4 Experiment 4: Backend Migration

The metrics in Tables 12 and 13 follow Section B.7. Final-State Acc. evaluates terminal-state satisfaction, and Required-Tool Comp. uses Equation (B.13) to measure the coverage of designated prerequisite-tool requirements in the executed trace.

C.5 Same-Trace Projection Audit

Table 14 reports all aggregate audit metrics. Relative cost uses the field-locked serialized records defined in Section B.7; it is not an API, token, or latency cost. The exact collision rate was zero for every projection after conditioning on K and retaining the complete fields specified for each view. This finite-sample result does not establish trace sufficiency, because false acceptance, disagreement, and Trajectory-Failure Recall continue to differ substantially across projections.

The macro-average in the main paper is expanded by failure class in Table 15. Call-only records did not identify any state, dependency, recovery, or backend failure. Adding agent-visible observations recovered some evidence in every class, whereas State-aware records retained all state failures and nearly all backend failures but still omitted dependency and recovery evidence available to Full audit.

Table 11: Complete marginal results for Experiment 3 across interaction perturbations. Panel (a) pools each model across prompting strategies, and panel (b) pools each strategy across model backbones. Success@1 evaluates the first sampled perturbed trajectory, State Acc. uses the Final-State Accuracy definition, and Recovery pools only fault-triggered eligible episodes.

(a) Model-level results

Model	Perturbation	Success@1	State Acc.	Recovery
DeepSeek-V4-Flash	Timeout	26.2%	53.8%	19.0%
	Schema drift	16.2%	35.0%	11.6%
	Stale state	6.2%	71.2%	0.0%
	Vague observation	16.2%	53.8%	27.1%
	Misleading observation	28.8%	78.8%	31.9%
GPT-OSS-120B	Timeout	30.0%	56.2%	23.6%
	Schema drift	28.8%	42.5%	26.5%
	Stale state	8.8%	70.0%	0.0%
	Vague observation	21.2%	53.8%	41.5%
	Misleading observation	35.0%	75.0%	38.4%
Gemma-4-31B-it	Timeout	25.0%	53.8%	23.5%
	Schema drift	18.8%	31.2%	28.0%
	Stale state	10.0%	68.8%	0.0%
	Vague observation	17.5%	47.5%	37.8%
	Misleading observation	30.0%	76.2%	32.0%
Qwen3.6-35B-A3B	Timeout	20.0%	52.5%	10.4%
	Schema drift	15.0%	32.5%	24.0%
	Stale state	10.0%	70.0%	0.0%
	Vague observation	23.8%	46.2%	43.2%
	Misleading observation	11.2%	73.8%	13.2%

(b) Strategy-level results

Strategy	Perturbation	Success@1	State Acc.	Recovery
Direct	Timeout	20.0%	52.5%	2.1%
	Schema drift	12.5%	27.5%	3.6%
	Stale state	7.5%	65.0%	0.0%
	Vague observation	13.8%	43.8%	27.5%
	Misleading observation	26.2%	72.5%	30.0%
CoT	Timeout	25.0%	52.5%	11.3%
	Schema drift	18.8%	32.5%	20.0%
	Stale state	8.8%	70.0%	0.0%
	Vague observation	17.5%	48.8%	35.0%
	Misleading observation	25.0%	75.0%	28.2%
ReAct	Timeout	28.8%	56.2%	32.1%
	Schema drift	22.5%	40.0%	34.3%
	Stale state	8.8%	72.5%	0.0%
	Vague observation	25.0%	57.5%	44.4%
	Misleading observation	25.0%	78.8%	26.7%
Self-Refine	Timeout	27.5%	55.0%	29.3%
	Schema drift	25.0%	41.2%	23.1%
	Stale state	10.0%	72.5%	0.0%
	Vague observation	22.5%	51.2%	40.0%
	Misleading observation	28.8%	77.5%	31.9%

Table 12: Complete results for Experiment 4 across execution backends. Panel (a) reports model-level results averaged across prompting strategies, and panel (b) reports strategy-level results averaged across model backbones.

(a) Model-level results				*(b) Strategy-level results*			
Model	Backend	Final-State Acc.	Required-Tool Comp.	Strategy	Backend	Final-State Acc.	Required-Tool Comp.
DeepSeek	Mock	71.7%	66.7%	Direct	Mock	67.5%	64.2%
	Sandbox	63.3%	61.7%		Sandbox	52.5%	52.5%
	Live MCP	60.8%	60.8%		Live MCP	44.2%	45.8%
GPT-OSS	Mock	72.5%	69.2%	CoT	Mock	67.5%	64.2%
	Sandbox	65.0%	63.3%		Sandbox	51.7%	51.7%
	Live MCP	58.3%	60.8%		Live MCP	46.7%	46.7%
Gemma	Mock	72.5%	69.2%	ReAct	Mock	77.5%	73.3%
	Sandbox	66.7%	66.7%		Sandbox	76.7%	75.0%
	Live MCP	63.3%	58.3%		Live MCP	74.2%	72.5%
Qwen	Mock	71.7%	68.3%	Self-Refine	Mock	75.8%	71.7%
	Sandbox	63.3%	63.3%		Sandbox	77.5%	75.8%
	Live MCP	57.5%	57.5%		Live MCP	75.0%	72.5%

Table 13: Complete paired backend results for Experiment 4 under Scheme A. Each row reports final-state accuracy for one model–strategy pair across mock, sandbox, and live MCP execution. Marginal summaries in Table 12 are computed from unrounded counts.

Model	Strategy	Mock	Sandbox	Live MCP
DeepSeek-V4-Flash	Direct	66.7%	50.0%	46.7%
	CoT	66.7%	50.0%	46.7%
	ReAct	76.7%	76.7%	73.3%
	Self-Refine	76.7%	76.7%	76.7%
GPT-OSS-120B	Direct	70.0%	56.7%	40.0%
	CoT	66.7%	50.0%	43.3%
	ReAct	76.7%	76.7%	73.3%
	Self-Refine	76.7%	76.7%	76.7%
Gemma-4-31B-it	Direct	70.0%	56.7%	50.0%
	CoT	66.7%	56.7%	53.3%
	ReAct	76.7%	76.7%	76.7%
	Self-Refine	76.7%	76.7%	73.3%
Qwen3.6-35B-A3B	Direct	63.3%	46.7%	40.0%
	CoT	70.0%	50.0%	43.3%
	ReAct	80.0%	76.7%	73.3%
	Self-Refine	73.3%	80.0%	73.3%

Table 14: Complete aggregate results for the same-trace projection audit. False Acc., False Reject., Disagree., Collision, and Trajectory-Failure Recall follow Equations (B.16)–(B.18). Rel. Cost is serialized projection size relative to Full audit.

Projection	False Acc.	False Reject.	Disagree.	Collision	Trajectory-Failure Recall	Rel. Cost
Call-only	56.9%	0.0%	56.9%	0.0%	0.0%	0.010
Action–Observation	45.0%	6.4%	51.4%	0.0%	36.9%	0.056
Outcome-only	17.9%	0.0%	17.9%	0.0%	80.0%	0.114
State-aware	14.4%	6.4%	20.8%	0.0%	87.6%	0.886
Full audit	0.0%	0.0%	0.0%	0.0%	100.0%	1.000

Table 15: Component-wise failure recall used to compute Trajectory-Failure Recall. Each column reports recall over trajectories exhibiting the named failure class.

Projection	State Failure	Dependency Failure	Recovery Failure	Backend Failure
Call-only	0.0%	0.0%	0.0%	0.0%
Action–Observation	21.6%	22.2%	41.6%	62.1%
Outcome-only	100.0%	82.7%	42.9%	94.3%
State-aware	100.0%	84.6%	65.9%	99.7%
Full audit	100.0%	100.0%	100.0%	100.0%

D EXPERIMENTAL CONFIGURATION

This section documents the task sources and shared data composition used across the four experiments, providing the configuration context needed to interpret and reproduce the reported comparisons.

D.1 BENCHMARK TASKS AND DATA SOURCES

Experiments 1–3 use all original ToolSandbox scenarios, comprising 100 scenarios in total, with no augmented variants. This shared task set contains 40 device-settings, 23 contacts, 22 reminders, 9 messaging, and 6 external-search scenarios. Experiment 4 instead uses tasks derived from MCP-Bench to evaluate execution under mock, sandbox, and live MCP backends.

Table 16: Task and episode counts for the four experiments. Episodes/model counts all experimental conditions evaluated for each model, and Total episodes aggregates runs across the four model backbones.

Experiment	Unique tasks	Experimental conditions	Episodes/ model	Models	Total episodes
Exp. 1	100	Stateful, Stateless	200	4	800
Exp. 2	100	4 strategies	400	4	1,600
Exp. 3	100	4 strategies $\times$ 1 perturbed attempt	400	4	1,600
Exp. 4	30	4 strategies $\times$ 3 backends	360	4	1,440

D.2 MODEL ENDPOINTS AND GENERATION PARAMETERS

Generation used the provider-side configuration associated with each model endpoint. Temperature, top-p, and maximum output tokens were not explicitly specified.

Table 17: Model endpoints used in the four experiments. Display names are used throughout the paper, while API model identifiers record the corresponding provider endpoints.

Display name	Provider	API model identifier
DeepSeek-V4-Flash	DeepSeek API	`deepseek-v4-flash`
GPT-OSS-120B	SiliconFlow	`openai/gpt-oss-120b`
Gemma-4-31B-it	SiliconFlow	`google/gemma-4-31B-it`
Qwen3.6-35B-A3B	SiliconFlow	`Qwen/Qwen3.6-35B-A3B`

D.3 EXECUTION PROTOCOL AND STEP BUDGETS

Within each experiment, all compared conditions used the same task specification, initial state, tool schemas, backend condition, and trajectory oracle; only the designated experimental factor and strategy-specific instructions were varied. An episode could terminate before exhausting its budget, and the ranges in Table 18 reflect task-adaptive maximum numbers of LLM decision steps.

Table 18: Maximum LLM decision-step budgets by experiment and inference strategy. Ranges indicate task-adaptive budgets.

Experiment	Direct/default	CoT	ReAct	Self-Refine
Exp. 2	2	4	8	10
Exp. 3	1–9	2–10	4–12	5–13
Exp. 4	1–7	1–7	10	11

API request timeouts were 60 s for Experiments 1–2, 120 s for Experiment 3, and 90 s for Experiment 4. Failed requests were retried up to two times in Experiments 1–2 and up to three times in Experiments 3–4.