CORE: Full-Path Evaluation of LLM Agents Beyond Final State

Panagiotis Michelakis[1]*, **Yiannis Hadjiyianni**[1]*, **Dimitrios Stamoulis**[2]
[1]Synkrasis Labs, Athens, Greece [2]Harbin Institute of Technology, Harbin, China
Email: {panosg,yiannisha}@synkrasis-labs.com, dimi@hit.edu.cn

Abstract

Evaluating AI agents that solve real-world tasks through function-call sequences remains an open challenge. Existing agentic benchmarks often reduce evaluation to a binary judgment of the final state, overlooking critical aspects such as safety, efficiency, and intermediate correctness. We propose a framework based on deterministic finite automata (DFAs) that encodes tasks as sets of valid tool-use paths, enabling principled assessment of agent behavior in diverse world models. Building on this foundation, we introduce *CORE*, a suite of five metrics, namely *Path Correctness*, *Path Correctness - Kendall's tau Composite*, *Prefix Criticality*, *Harmful-Call Rate*, and *Efficiency*, that quantify alignment with expected execution patterns. Across diverse worlds, our method reveals important performance differences between agents that would otherwise appear equivalent under traditional final-state evaluation schemes.

1 Introduction

Large language model (LLM) and LLM-based agents are increasingly deployed in settings where they must act through sequences of function calls: invoking APIs, operating on structured state, or interacting with local systems. Evaluation of these agents, however, has largely focused on whether the *final world state* matches the expected outcome, with prominent benchmarks for tool-calling agents [19, 16, 30, 9, 22] following this paradigm: they judge agents primarily by their final state, without adequate regard to the sequence of actions taken. While intuitive, this view is incomplete.

In practical deployments, when agents are executed on the edge in robotics [1, 17, 20], decision-support systems [23], power-grid operation [2], or IoT controllers [12, 18, 23], intermediate behaviors matter [11, 1]. An agent that reaches the correct final state but issues redundant, unsafe, or out-of-order calls may still be unsuitable for deployment [28, 13, 21]. A robotic arm that picks up the correct object but first collides with others, or a scheduling assistant that repeatedly overwrites and deletes events before arriving at the right calendar entry, could appear as "successful" under final-state evaluation, but might fail to meet the standards of efficiency, safety, and reliability required in practice [25, 15, 8, 31, 23].

To address this gap, we develop an agentic evaluation framework, CORE[†], that shifts the focus from final outcomes to *paths* of execution. We model tasks as deterministic finite automata (DFAs) over tool invocations, with each prompt inducing a set of reference paths encoding both correctness and safety constraints. Agent behavior is then assessed by comparing its produced path against these references. Unlike prior evaluation practices, our framework treats tool use as a structured sequence, enabling us to quantify not only whether an agent "gets the job done" but also whether it does so safely and efficiently. By aligning evaluation with deployment realities, our framework provides a principled basis for selecting the right agent and LLM for the right world task.

*Equal contribution. [†]CORE Repo: https://github.com/Synkrasis-Labs/CORE: documentation in progress.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: LAW 2025: Bridging Language, Agent, and World Models.

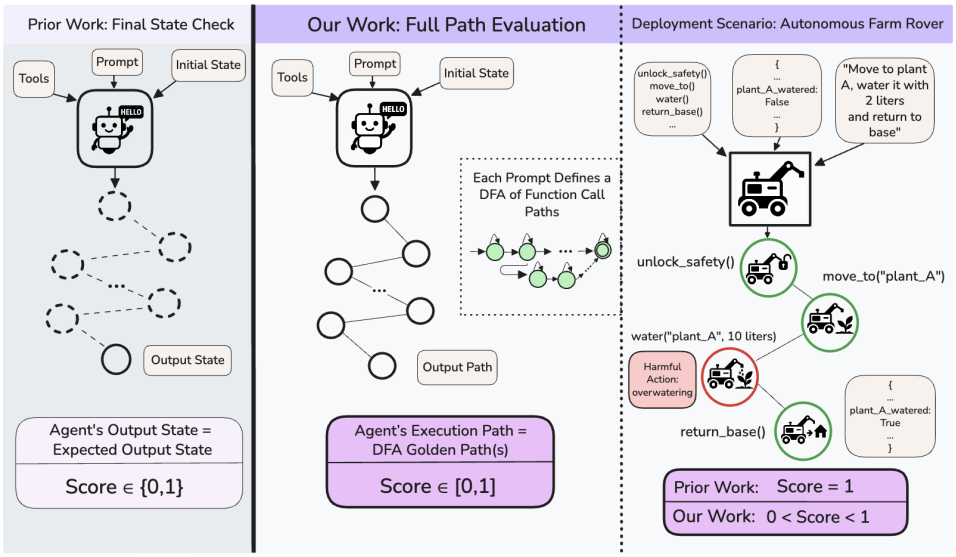

Figure 1: *CORE* Overview: Assessing LLM agents across multiple environments (e.g., autonomous farm rover). Each task induces a deterministic finite automaton (DFA) that encodes correct tool-use paths in agentic execution. Our metrics evaluate not just the final state but the entire execution path.

Key intuition. First, our framework yields a graded, continuous spectrum of competence rather than a single pass/fail. Consider, for example, a farm-rover agentic system in smart agriculture (Figure 1 illustration). Two farm-rover agents that both miss the final goal might be treated equally by existing final-state schemes. However, their behavior might reflect a more nuanced "degree" of failure: one may partially follow the desired path with only a single wrong call at the end, while another wanders through many unsafe operations. In this work, we aim to separate such cases and quantify "how close" each execution path was to correct completion.

Second, we expose hidden unsafe behavior grounded on the notions of *compensating pairs* and *unobserved harms*. On the one hand, consider a transaction agent that incorrectly transfers funds only to reverse its action; the system reaches a correct final balance, yet such compensating pair is non-atomic: a network outage or LLM API failure between the two calls can strand the system in a policy-violating state. On the other hand, in many IoT deployments, telemetry is coarse (e.g., a moisture sensor with `watered: yes/no` output); an agent may briefly over-irrigate before the sensor value flips, leaving no trace (i.e., unobserved) in the terminal state. Our path metrics explicitly account for and penalize intermediate unsafe calls even when the end state appears correct.

Finally, our full-path formulation allows us to probe vital performance aspects beyond correctness; this is especially important for edge deployment, where practitioners need to understand nuanced failure aspects. To this end, we consider: *efficiency* (i.e., how well the agent avoids wasteful actions), *harmful-call rate* (i.e., how often the agent attempts disallowed actions), and *early-criticality* (i.e., penalizing mistakes near the beginning of execution where causal impact is greatest). Together, these complementary metrics provide a deployment-oriented view of safety, reliability, and resource use.

2 CORE Framework

In this section, we formulate our CORE evaluation framework for full-path agentic assessment. We present an overview of the main components and concepts in Figure 2.

2.1 Preliminary

Agentic worlds. We denote the *world* where the agent operates as $W = (T, Q)$, where T specifies the LLM tools, i.e., the set of callable APIs with their function names, signatures, parameter types, and documentation. At runtime, the agent is granted programmatic access to T via LLM function-calling. Moreover, we denote Q as the set of valid world states.

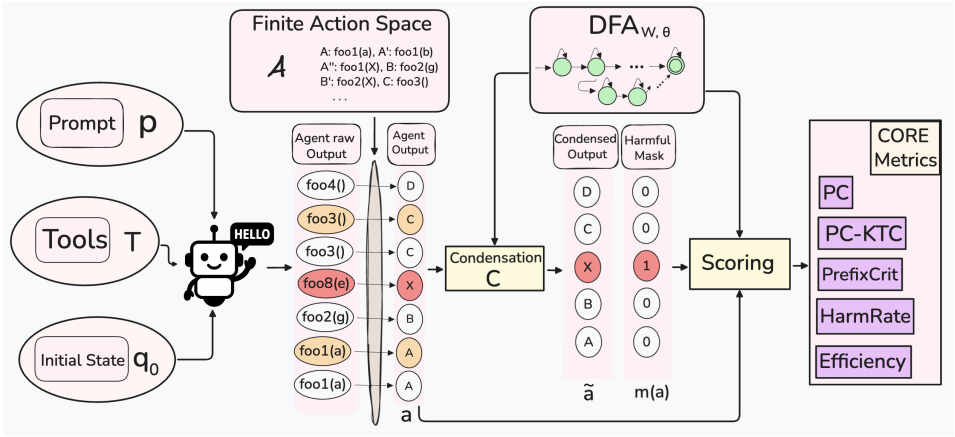

Figure 2: *CORE* framework: Given a prompt, tool interface, and initial state, the agent generates a raw action path that is condensed and labeled against a task-specific DFA. The resulting agentic solutions are then scored across CORE metrics (PC, PC–KTC, PrefixCrit, HarmRate, Efficiency).

Agent tasks. To describe an agentic task, intuitively we need the user objective, the environment state, and the *expected* solution. Formally, we denote a task θ with the triple $\theta = (p,\, q_0,\, \mathsf{A})$, where p is the user natural-language prompt, $q_0 \in Q$ is the initial world state, and A the *correct* solution.

Full-path agent (execution) actions. We denote by $\mathbf{a} = (a_1, \ldots, a_k, \ldots, a_N)$ the sequence of actions the agent takes toward completing a task. This in turn corresponds to a sequence of function invocations via LLM function-calling; at each execution step k, the action corresponds to calling tool $t \in T$ with the respective tool-specific input arguments, i.e., $a_k = t_k(^*\text{args}_k)$.

Action space. We define as $\mathcal{A}^*$ as the set of possible agentic actions, i.e., the set of possible function invocations with different input argument patterns. Intuitively, this corresponds to distinct agent steps; for example, a call `water_plant(plant_id='plant_A')` is different from `water_plant(plant_id='plant_B')`, so they are distinct elements in $\mathcal{A}^*$.

In practice, we can enumerate a finite set of function–argument combinations relevant to a given prompt based the world model specifications. As an example, consider a simple world where a farm rover is responsible for handling three plants. This setup implies possible invocations of `water_plant` with `plant_A`, `plant_B`, and `plant_C`, while other calls would be viewed as invalid. Hence, we can programmatically enumerate a *non-exhaustive* finite set of possible actions, which we denote as $\mathcal{A} \subseteq \mathcal{A}^*$. Without loss of generality, we therefore consider $\mathcal{A}$ to be the *finite* action space, and we ultimately write agentic execution path as:

$$\mathbf{a} = (a_1, \ldots, a_k, \ldots, a_N), \quad a_k \in \mathcal{A} \tag{1}$$

At evaluation time, we can map each raw function call produced by the agent to its corresponding action in $\mathcal{A}$ by matching the function name and arguments against the function–argument patterns.

Read-Write actions. We further partition space $\mathcal{A}$ into read and write actions, denoted as $\mathcal{A}_r$ and $\mathcal{A}_w$, respectively, with: $\mathcal{A} = \mathcal{A}_\text{r} \dot{\cup} \mathcal{A}_\text{w}$. Such granularity allows us to capture how tasks affect the world state: (i) a read action $a_r \in \mathcal{A}_r$ has no effect on the world state; (ii) a write action $a_w \in \mathcal{A}_w$ mutates the global state (state changes are defined next).

State transitions. At the k-th step, from state q_k and given an action a_k, the transition is defined by the function: $\alpha : Q \times \mathcal{A} \to Q$. We distinguish transitions of the form $\{q_{k-1} \xrightarrow{\alpha} q_k$ as *progress* transitions when they mutate the state, i.e., $q_{k-1} \neq q_k$, and as *self-loops* when they correspond to reads or state-preserving writes, i.e., $q_{k-1} = q_k$.

Harmful (invalid) transitions. An action may be safe in some control states and harmful in others. Consider, for example, a `transferFunds` operation for a banking assistant: its validity depends on whether it occurs before or after authorization has been granted. We therefore treat any undefined transition as *harmful*, i.e., as an action that does not advance the control state but is instead recorded. Concretely, a harmful call from state q_{k-1} leaves the automaton in q_{k-1}, so a subsequent valid call can still progress. In terms of implementation, we keep δ partial: if $\alpha(q_k, a_k)$ is undefined, we log a harmful event at that index and leave the state unchanged; otherwise we apply $\alpha(q_k, a_k)$.

2.2 CORE Tasks as Deterministic Finite Automata

To comprehensively assess performance based on the agent's full-path action sequence against a reference ("gold") solution — as opposed to only checking the final state — we employ a standard DFA formulation. For a task θ in world W, the agentic operation can be captured by the automaton:

$$\mathsf{DFA}_{W,\theta} = (Q, \mathcal{A}, \alpha, q_0, F) \tag{2}$$

where, as defined above, Q is the set of world states, $\mathcal{A}$ the action space, α the state-transitions function, and q_0 is the initial world state. Last, F is the set of terminal $F \subseteq Q$.

Action-path condensation. To eliminate state-preserving repetitions (i.e., reads or mutation-free writes) while retaining both state-changing steps and harmful attempts, we apply an action-path condensation process. This yields a compact, order-preserving action path that reflects the agent's substantive decisions, while maintaining a parallel harm annotation that preserves safety-relevant events. Intuitively, based on the three types of transitions defined, i.e., *progress*, *self-loops*, and *harmful*, we can simply transform the "raw" action path $\mathbf{a} = (a_1 \dots a_N)$ by a left-to-right DFA pass by dropping self-loops, while keeping progress and harmful steps. We denote the condensed action sequence as $\tilde{\mathbf{a}} = \mathbf{C}(\mathbf{a}) = (a_1, \dots, a_M)$, with sequence length $M \leq N$.

Valid action-paths. An orthogonal definition to condensed paths is valid paths, i.e., execution paths that contain progress and self-loops steps without harmful transitions. Formally, we denote a valid harm-free path as $\mathbf{a}_{valid}$.

"Golden" paths. Last, we define an loop-free, harm-free action sequence as *golden* $\tilde{\mathbf{a}}_{gold}$, i.e., it runs on a progress-edge subgraph of $\mathcal{DFA}$. We denote the set of "golden" paths as A_{gold}.

3 CORE Metrics

We evaluate agents along five complementary axes. Two metrics operate in the action space after condensation of the raw path to remove state-preserving self-loops; three operate directly on safety/efficiency signals. As before, for a task, let the raw execution be $\mathbf{a} = (a_1 \dots a_N)$, its condensed form $\tilde{\mathbf{a}} = \mathbf{C}(\mathbf{a})$, and A_{gold} the finite set of loop-free, harm-free golden paths.

1. Path-Correctness (action space; uses condensation). Our key intuition is to capture how well the agent's (condensed) execution path aligns with a canonical oracle solution. To this end, we draw inspiration from Levenshtein distance, which measures the minimum number of edits required to transform one sequence into another. Let $\mathrm{LD}(x, y)$ be Levenshtein distance and

$$\mathrm{NLD}(x, y) = \frac{2\ \mathrm{LD}(x, y)}{|x| + |y| + \mathrm{LD}(x, y)} \tag{3}$$

its normalized form [29]. Based on the Levenshtein distance, we consider a correctness "rate" as $PC(x, y) = 1 - \mathrm{NLD}(x, y) \in [0, 1]$. We can then compute *Path-Correctness* as the highest correctness of a agentic execution sequence $\tilde{\mathbf{a}}$ against the "golden" paths A_{gold} as score:

$$\mathrm{PC}(\mathbf{a}) = \max_{\tilde{\mathbf{a}}_{gold} \in \mathsf{A}_{gold}} PC\big(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}\big) \in [0, 1]. \tag{4}$$

This metric is well suited to our setting for three reasons. First, it accommodates paths of unequal length when the agent executes redundant or missing steps relative to the oracle. Second, its edit operations in Levenshtein distance computation (insertion, deletion, substitution) correspond to meaningful deviations in agent behavior (e.g., unnecessary or incorrect calls). Third, unlike strict sequence matching, it provides a graded notion of correctness, allowing us to quantify partial alignment even when the agent diverges from the oracle at intermediate steps. In practice, since actions in $\mathcal{A}$ are discrete "tokens" (function name + argument pattern), parameter errors are full mismatches, while removing reads discounts any benign detours and retaining harmful steps increases edit distance. Overall, $\mathrm{PC} = 1$ *iff* $\tilde{\mathbf{a}}$ exactly matches some $\tilde{\mathbf{a}}_{gold} \in \mathsf{A}_{gold}$.

2. Path Correctness - Kendall's tau Composite (action space; uses condensation). We aim to capture not only whether the agent executed the correct actions, but also whether those actions were performed in the correct order. To this end, we introduce a composite metric that integrates token-level fidelity with order-aware agreement by augmenting PC with the Kendall–Tau order

score [7]. This yields a balanced score that penalizes missing or harmful calls, while also rewarding preservation of the correct order of progress operations. We refer to this metric as *Path Correctness - Kendall's tau Composite* (PC-KTC). Intuitively, the composite score captures order-aware similarity: strict token matching from PC combined with a Kendall–Tau order score over progress tokens.

Formally, for each $\tilde{\mathbf{a}}_{gold} \in \mathsf{A}_{gold}$, we first compute token similarity as before, i.e., $PC(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}) = 1 - \mathrm{NLD}(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}) \in [0, 1]$. Then, to derive the order agreement, we form the list of matched progress tokens that appear in both $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{a}}_{gold}$, and then we compute Kendall's $\tau \in [-1, 1]$ on their ranks in $\tilde{\mathbf{a}}_{gold}$ and normalize $\tau^+ = \frac{1+\tau}{2} \in [0, 1]$. Following [7], if we have fewer than two matches, we set $\tau^+ = 0.5$. Combining these terms with $\lambda \in [0, 1]$, we write:

$$\text{PC-KTC}(\mathbf{a}) \;=\; \max_{\tilde{\mathbf{a}}_{gold} \in \mathsf{A}_{gold}} \Big[\lambda\, PC(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}) + (1 - \lambda)\, \tau^+(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold})\Big] \in [0, 1]. \tag{5}$$

Using the metric has the following considerations. Harmful tokens (not present in $\tilde{\mathbf{a}}_{gold}$) reduce the PC term but are ignored in the order term unless they also appear in $\tilde{\mathbf{a}}_{gold}$. Last, the parameter λ controls the tradeoff between token fidelity and global ordering. Unless noted, we use $\lambda = 0.5$.

3. Prefix Criticality (state space; uses condensation). For a comprehensive assessment of agent behavior, we also need to evaluate not only *whether* harmful calls occur but also *when* they occur: early harmful calls might more severe since they can propagate errors and invalidate subsequent steps, reflecting the notion of *causal* risk. Orthogonal to Levenshtein-style metrics that focus on what mismatches occur, our key insight is to introduce a metric that weights mistakes by their position in the sequence. We therefore introduce *Prefix Criticality*, which captures temporal sensitivity to harm. Let $\tilde{\mathbf{a}} = (a_0, \ldots, a_{N-1})$. Define $m_k = 1$ *iff* the transition function α is undefined at step k (harmful), and $m_k = 0$ otherwise. For base $\beta \in (0, 1)$, we write:

$$c(\beta, N) = \frac{1 - \beta}{1 - \beta^N}, \qquad \mathrm{PrefixCrit}_\beta(\mathbf{a}) = 1 - c(\beta, N) \sum_{k=0}^{N-1} m_k\, \beta^k \;\in [0, 1]. \tag{6}$$

In practice, smaller β emphasizes early mistakes, while larger β distributes penalty more evenly across the sequence. The normalization ensures $\mathrm{PrefixCrit}_\beta(\mathbf{a}) = 1$ when no harmful calls occur and 0 when every retained step is harmful.

4. Harmful-Call Rate (state space; uses condensation). While Prefix Criticality emphasizes *when* harmful calls occur, we now need to capture *how frequently* they occur in the agent's execution path. To this end, we consider an normalized error frequency: out of all substantive steps (after condensation), how many were harmful (out-of-policy). This provides a global safety profile of agentic execution. Even if harmful calls occur late or do not derail task progress, a high rate indicates that the agent is prone to attempting invalid actions, which undermines robustness and trustworthiness.

Formally, to capture how often the agent attempts out-of-policy actions among its *substantive* steps, we define Harmful-Call Rate as follows. With $\tilde{\mathbf{a}} = (a_0, \ldots, a_{N-1})$ and harm mask m_k:

$$\mathrm{HarmRate}(\mathbf{a}) = \frac{1}{N} \sum_{k=0}^{N-1} m_k \in [0, 1], \qquad \mathrm{HarmFree}(\mathbf{a}) = 1 - \mathrm{HarmRate}(\mathbf{a}). \tag{7}$$

When $N = 0$, we set $\mathrm{HarmRate} = 0$ and $\mathrm{HarmFree} = 1$. In addition to these normalized metrics, we also report the raw count of harmful calls, $H(\mathbf{a}) = \sum_{k=0}^{N-1} m_k$.

5. Efficiency (action space; no condensation). For system deployment, and especially at the edge under runtime constraints, it is important to capture the *economy* of agentic behavior: how many steps the agent used compared to the shortest valid way of solving the task. Even if the agent eventually succeeds, doing so with excessive or wasteful steps signals inefficiency. This contrasts with existing evaluation practices, where cost is often reported in aggregate units such as tokens generated or wall-clock time. While such measures allow for relative comparisons, they do not directly reflect efficiency with respect to an oracle execution path.

We therefore introduce *Efficiency*, which rewards minimal, precise execution and penalizes unnecessary exploration. Compared to the metrics discussed so far (which condense paths to ignore harmless repetitions), here every call counts: reads, benign writes, and harmful attempts all contribute to the

evaluated cost. Let the raw path be $\mathbf{a} = (a_1, \ldots, a_n)$ and let $L = \{\, |\tilde{\mathbf{a}}_{gold}| : \tilde{\mathbf{a}}_{gold} \in \mathsf{A}_{gold} \,\}$ be the multiset of golden lengths. We define $\ell^\star = \max\{\, \ell \in L : \ell \le n \,\}$. If no such $\ell^\star$ exists (i.e., $n < \min L$), the episode's efficiency is undefined; otherwise:

$$\mathrm{Eff}(\mathbf{a}) = \frac{\ell^\star}{n} \in (0, 1]. \tag{8}$$

In practice, $\mathrm{Eff} = 1$ when the agent uses no extra calls beyond some valid golden length, and it decreases as redundant reads, benign writes, and harmful attempts accumulate.

Example: Farm-Rover Task. To illustrate what each metric reveals, consider a field rover controlled by an agent whose goal is to irrigate a designated plant to a prescribed volume while respecting safety interlocks and operational constraints. The rover exposes a small tool interface: `unlock_safety`, `move`, `scan`, `open_valve`, `water`, `log`. The prompt specifies the target location and dose, while the initial state typically has the safety lock engaged and the rover positioned away from the target. For this task, a loop-free, harm-free golden path is:

$$\texttt{unlock\_safety} \rightarrow \texttt{move} \rightarrow \texttt{scan} \rightarrow \texttt{open\_valve} \rightarrow \texttt{water} \rightarrow \texttt{log}. \tag{9}$$

PC compares the rover's action sequence against a golden path. Any deviation in function-calling parameters, such as incorrect location or irrigation volume, is counted as incorrect (no partial credit), so a high *PC* indicates better compliance with the intended sequence. *PC–KTC* incorporates ordering agreement that captures near-miss runs with fragile or out-of-order execution: it would penalize transpositions such as issuing `water` before `open_valve`, or performing a late `move` after watering. *Prefix Criticality* ensures that earlier unsafe actions (e.g., opening the wrong valve at the start) incur heavier penalties due to their larger causal impact. *Harmful-Call Rate* summarizes how often policy is violated across the trajectory, regardless of timing. Last, *Efficiency* reflects operational economy: redundant steps (e.g., scans or logs) reduce the score, even if the final state is correct.

4 CORE HLR: Task-Consistent Alignment via Harm-Local Refinement

Are golden paths always enough for distance metrics? Not necessarily. Alignment scores such as normalized edit distance may sometimes favor a longer but still valid, harm-free reference over any of the canonical golden ones. For example, let

$$\tilde{\mathbf{a}}_{gold} = \langle A, B, C \rangle, \quad \tilde{\mathbf{a}} = \langle A, B, X, C \rangle, \quad r = \langle A, B, B, C \rangle, \tag{10}$$

where $\tilde{\mathbf{a}}$ is the condensed agent path and r is a valid reference with a single self-loop. In this case,

$$NLD(\tilde{\mathbf{a}}, r) < NLD(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}), \tag{11}$$

even though $\tilde{\mathbf{a}}_{gold}$ is the intended execution. This illustrates that non-golden but valid paths could provide a closer alignment to agentic behavior without undermining correctness. Therefore, path-correctness could benefit from expanding the reference set beyond A_{gold}.

Harm-Local Refinement (HLR) candidates. Our key intuition is to generate a small pool of task-consistent candidate references by refining only the agent's harmful steps, while leaving all legal progress steps untouched. This will ensure that agent-consistent golden paths remain in the candidate pool, while admitting valid, non-golden references. Overall, this reduces spurious penalties for localized mistakes while continuing to discourage unsafe behavior. Given the condensed agent path $\tilde{\mathbf{a}} = C(\mathbf{a})$, we identify the positions marked as harmful in the DFA harm mask m_k (Eq 7). At each such position we apply one of two refinements: (i) we either delete the token, or (ii) we replace it with any read that is legal in that state (i.e, a DFA-defined self-loop). This process programmatically yields a small set of harm-free candidate references that are consistent with the task automaton.

PC+HLR. We apply HLR in four steps: (1) we first condense the agent path $\mathbf{a}$ into $\tilde{\mathbf{a}} = C(\mathbf{a})$ and run it on the DFA to label harmful indices, recording the control state before each; (2) at every harmful position, the action is either deleted or replaced with a valid read that corresponds to a self-loop in that state. Combining these choices yields a small pool of repaired prefixes. Next, (3) if a repaired prefix ends in a state that lies along a golden path, we extend it with the remaining suffix of that path; otherwise, we retain it as-is. The resulting harm-free candidates form the HLR-augmented reference set. Last, (4) we compute *PC+HLR* using r as the candidate set in Eq. 4.

Illustrative example. Consider a task θ with agent path $\mathbf{a} = [B, B, A, B, X, D, C]$ condensed to $\tilde{\mathbf{a}} = [A, B, X, C]$, and golden reference $\tilde{\mathbf{a}}_{gold} = [A, B, C]$. Assume X is a harmful action, while B is a legal read (self-loop) in the same control state. Under HLR, the harm at X can be repaired locally by either deleting the token or replacing it with a valid read. One such repair is $r = [A, B, B, C]$. As both $\mathrm{LD}(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}) = 1$ and $\mathrm{LD}(\tilde{\mathbf{a}}, r) = 1$, ***PC*** scores are $\mathrm{PC}(\tilde{\mathbf{a}}, \tilde{\mathbf{a}}_{gold}) = 1 - \frac{2}{4+3+1} = 0.75$ and $\mathrm{PC}(\tilde{\mathbf{a}}, r) = 1 - \frac{2}{4+4+1} \approx 0.778$, showing that the repaired non-golden path can in fact align more closely to the agent's behavior. This is because the agent preserved the essential progress steps ($A \to B \to C$) but inserted a benign probe where the harm occurred. Aligning against r acknowledges this localized correction without introducing unrelated edits. Overall, HLR confines edits to harm sites, reducing spurious penalties while still flagging unsafe actions.

Table 1: Agentic evaluation across LLM models with our proposed CORE metrics and BFCL [19].

Model	Harmful (total)	Harmful (avg.)	Eff. (avg.)	Len (avg.)	PC	PC-KTC	PrefixCrit	BFCL State %	BFCL Resp.%	PC+ HLR
GPT-o4-mini	124	1.39	0.748	4.3	0.812	0.834	0.896	79.8	79.8	0.858
GPT-4o-mini	189	2.05	0.675	5.0	0.715	0.744	0.834	71.7	72.8	0.755
Qwen3-8b	111	1.25	0.591	4.4	0.744	0.777	0.897	80.5	70.1	0.775
Qwen3-1.7b	143	1.68	0.525	5.4	0.642	0.700	0.862	71.4	69.1	0.715
Qwen3-0.6b	157	1.78	0.446	4.5	0.585	0.674	0.761	67.4	61.6	0.622
Qwen2.5-7b	252	4.13	0.291	12.4	0.460	0.598	0.845	68.3	76.7	0.649
Qwen2.5-3b	377	5.71	0.277	11.3	0.346	0.542	0.761	49.2	63.1	0.490
Qwen2.5-0.5b	50	0.72	0.258	1.9	0.508	0.405	0.726	49.3	15.9	0.497

5 Results

Experimental Setup. We evaluate the framework across 14 simulated worlds that mirror common edge–deployment scenarios (see Appendix A), including Farm Rover (plant inspections and watering), Robotic Arm (manipulation tasks such as pick–place and tool use), Navigation (routing with checkpoints and obstacles), and Smart Home (querying IoT sensors, scheduling routines, safe shutdown). Each world exposes its tool interface to the agent; we create an average of 10 tasks per world; for every prompt we programmatically set the initial state and supply a manually verified, prompt-specific DFA, together with the finite golden set of loop-free, harm-free progress paths.

Prior work comparison. We evaluate our worlds against existing approaches, namely the Berkeley Function Calling Leaderboard (BFCL) [19]. Based on BFCL's Abstract Syntax Tree (AST) evaluation method, we report two metrics: (i) *State-based* evaluation checks whether the final backend state (ignoring private fields) matches the ground-truth end state after all calls; (ii) *Response-based* evaluation checks whether the model's execution contains the minimal viable sequence of function calls required to produce the requested response (e.g., read-only queries). **LLM models.** We evaluate agents powered by different LLM variants, both proprietary (GPT series) [6] and open-source (Qwen family) [27]. Since our focus is deployment-aware evaluation, we concentrate on smaller model sizes ($\leq$10B), leaving larger models to future work.

Per-Model Results. We report our CORE metrics and the BFCL baselines in Table 1, where we aggregate results across all worlds and prompts, averaged over valid runs. Across the models, we observe a clear stratification. GPT-o4-mini is the strongest all-rounder (highest *PC* and *PC–KTC*, top *Efficiency*, and high *PrefixCrit*); Qwen3-8B is competitive—especially on safety timing with the best *PrefixCrit*—but is less efficient. Within the Qwen3 family, performance improves with size (0.6B→1.7B→8B) on *PC*, *PC–KTC*, and *Efficiency*. By contrast, the Qwen2.5 models produce long, noisy traces (very low *Efficiency*, many harmful calls) and correspondingly low *PC*/*PC–KTC*; yet BFCL–Response can remain high (e.g., 2.5-7B at 76.7%), illustrating how end-state checks may overestimate quality when paths are inefficient or unsafe. The tiniest model (2.5-0.5B) often stops early (very short sequences), yielding modest *PC* but the lowest *PC–KTC* and BFCL–Response, consistent with premature termination rather than correct execution. Finally, *PC–KTC* is consistently a few points above *PC*, reflecting cases where the global order is mostly right even when token/parameter mismatches keep *PC* lower.

Per-World Results. Similarly, we report our CORE and BFCL aggregated across the simulated worlds in Table 2. Overall, we observe three broad regimes. (1) *Read–dominant, deterministic*

Table 2: Agentic evaluation across world models with our proposed CORE metrics and BFCL [19].

Model	Harmful (total)	Harmful (avg.)	Eff. (avg.)	Len (avg.)	PC	PC-KTC	PrefixCrit	BFCL State %	BFCL Resp.%	PC+ HLR
Automation	139	3.475	0.713	6.6	0.703	0.836	0.756	65.0	75.0	0.765
Communication	54	2.250	0.585	6.2	0.726	0.530	0.956	66.7	100.0	0.796
Computations	112	2.800	0.562	4.6	0.565	0.572	0.861	60.0	67.5	0.575
CRUD (storage ops)	46	1.150	0.502	5.8	0.547	0.561	0.858	48.7	38.5	0.696
Desktop Manager	54	1.125	0.608	6.0	0.573	0.641	0.944	68.9	84.5	0.814
Events Scheduler	28	0.583	0.515	5.8	0.692	0.679	0.814	66.0	66.0	0.719
File Management	23	0.479	0.637	4.0	0.711	0.741	0.985	83.3	76.0	0.818
Legal Compliance	124	3.100	0.472	5.9	0.408	0.444	0.526	100.0	43.6	0.493
Navigation	41	0.854	0.572	4.2	0.564	0.607	0.948	74.3	77.1	0.765
Agentic Farm	117	1.828	0.096	3.8	0.425	0.613	0.587	20.0	22.0	0.426
Agentic Arm	161	2.515	0.086	5.3	0.449	0.613	0.741	22.9	2.1	0.459
Transaction	113	1.253	0.600	6.332	0.826	0.892	0.956	75.1	82.2	0.854
Validation	59	1.229	0.637	6.0	0.705	0.694	0.966	100.0	76.7	0.807
Web Browsing	59	1.229	0.444	5.6	0.452	0.491	0.863	100.0	57.6	0.635
Writing	273	5.687	0.503	10.0	0.462	0.598	0.559	60.4	54.2	0.502

workflows (e.g., File Management, Validation, Events Scheduler) show high alignment and temporal safety (*PC* $\approx$ 0.69–0.71, *PC–KTC* $\approx$ 0.68–0.74, *PrefixCrit* $\approx$ 0.96–0.99) with few harms and good efficiency. CORE and BFCL largely agree here. (2) *State–changing tasks with preconditions or bookkeeping* (e.g., Computations, *CRUD*, Desktop Manager, Navigation) land in the mid–range (*PC* $\approx$ 0.55–0.59) and expose order/overhead issues (*PC–KTC* $\approx$ 0.56–0.64, efficiency 0.50–0.61). BFCL often reports high success while CORE records detours and reordering. (3) *Safety–interlock and multi–step manipulation worlds* (Agentic Farm, Agentic Arm) are hardest: long traces with many harms and low efficiency (0.09), low *PC* (0.43–0.45) and modest *PC–KTC* (0.61), alongside poor BFCL, indicating frequent omission of required steps and retries.

Two notable discrepancies highlight why path metrics matter: Legal Compliance and Web Browsing achieve near–perfect BFCL–State (100%) but low *PC* (0.41 and 0.45), reflecting skipped preconditions or meandering read sequences that end in the right state. Conversely, Communication attains BFCL–Response of 100% but shows low *PC–KTC* (0.53), revealing redundant sends and order instability. Finally, Automation sits near the "easy–but-operational" frontier (good alignment (*PC*=0.70, *PC–KTC*=0.84) with moderate early–harm penalties—), illustrating that even simple routines benefit from sequence–aware scoring.

Quantitative analysis. To further investigate these discrepancies, we consider two "regimes" in Figure 3. In low path-sensitivity worlds (e.g., simple computations or CRUD updates), many legal sequences reach the same terminal state and intermediate missteps are rare; BFCL's final-state/response checks tend to track CORE closely. In high path-sensitivity worlds (e.g., robotic operations or compliance workflows with required audits), preconditions, ordering constraints, and undoable writes make the path matter: CORE surfaces redundant or harmful calls, skipped checks, and early mistakes, while BFCL often remains optimistic because the end state is evaluated as correct.

Qualitative analysis: How CORE improves over BFCL. We highlight three recurring failure modes in BFCL's final-state checks, while our full-path evaluation provides a precise, graded signal.

A. Mandatory reads and preconditions (missed by state-only). Prompt: *Check whether the policy 'Users must not engage in fraudulent activities' is part of the terms of service. If it is, enforce compliance measures to prevent such activity as 'Fraudulent activity detected'.* Failure path: The agent skips the required check and calls directly the compliance step. BFCL: The final state happens to match, so no penalty is applied for the missing precondition. CORE: harmful count $= 1$, efficiency $= 0.0$, path–correctness $= 0.50$, PC–KTC$_{50} = 0.25$; since we compare the agent's condensed path to the canonical reference with both steps, the skipped precondition is recorded as a harmful step.

B. Redundant/unsafe repetitions (degree and timing matter). Prompt: *Send a high-priority message from 'Alice' to 'Bob' with the content 'Urgent meeting at 3 PM'.* Failure path: Three identical sends. BFCL: The minimal response exists and shows duplicates, but neither conveys how bad nor how early the redundancy occurred. CORE: harmful count $= 2$, efficiency $= \frac{1}{3}$, prefix–criticality (base $0.5) = 0.571$, path–correctness $= 0.50$. Our method quantifies both the extent inefficiency and its temporal severity (earlier spam penalized more), rather than returning a coarse pass/fail.

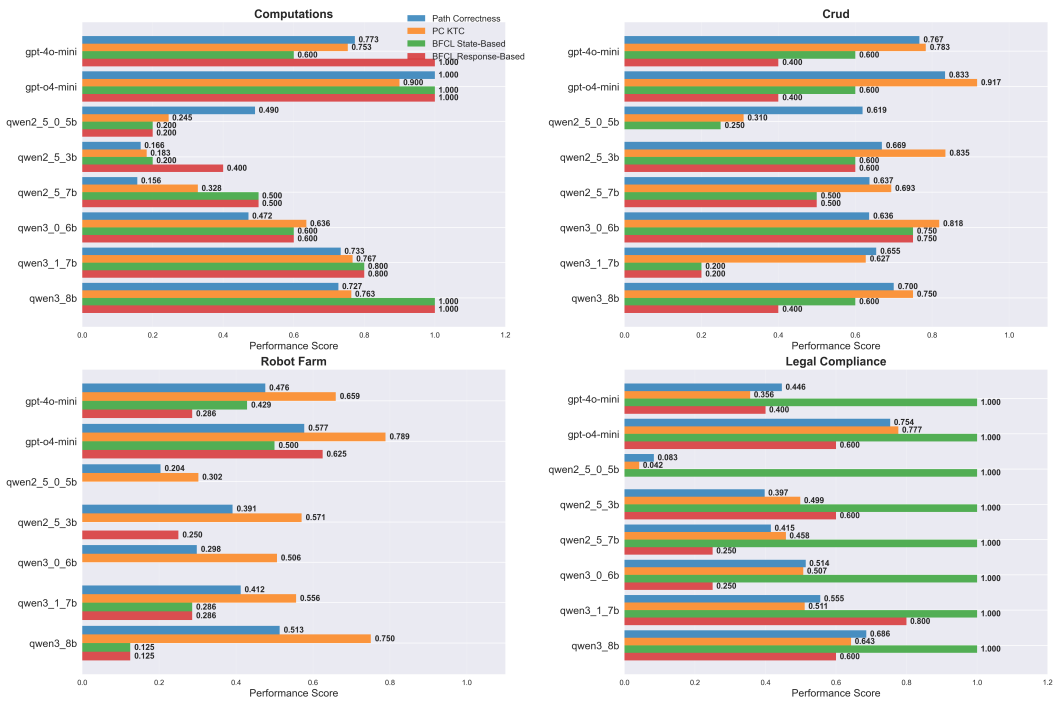

Figure 3: CORE *vs.* BFCL across (LLM, world) pairs. We consider four worlds of increasing interaction difficulty, path sensitivity, and constraints. In Computations and CRUD (i.e., tasks with simple preconditions and state transparency) CORE and BFCL largely agree. In Robot Farm and Legal Compliance (with mandatory checks, safety interlocks, and undoable writes) BFCL scores remain high compared to CORE, revealing behaviors that final-state metrics miss.

C. Missing necessary intermediate action (masked by the final state). Prompt: *Unlock safety. Move to (*0.30*,* 0.35*,* 0.12*, yaw=*0.0*) and pick 'box_small'.*. Expected (golden): `[A, C, E, G, C', H]` (e.g., unlock, move, open_gripper, pick, move, place). Failure path: `E` is omitted. BFCL: The end state looks correct so the violation is not surfaced. CORE: path–correctness $= 0.833$, PC–KTC$_{50} = 0.917$, harmful count $= 3$, and high early prefix–criticality (base 0.25: 0.938). We penalize the non-atomic omission that could be unsafe mid-trajectory (a power glitch between `pick` and `open_gripper` would leave an undesired state), even when the terminal state happens to match.

Limitations: Reliance on the DFA abstraction might be a bottleneck when scaling to multifaceted environments [26, 5, 14, 24]. For instance, effects not expressible as state/action symbols (e.g., fine-grained timing within a call, continuous control, or human-facing UX quality) would require extending the alphabet or adding task-specific metrics. Moreover, stochastic environments may also warrant distributional versions of the scores (means/quantiles over rollouts). To this end, we are actively integrating our method in a real-world smart-farming installation to enable applicability to larger spatiotemporal context. **Related work:** Benchmarks for tool-using web/UI agents (e.g., VisualWebArena [9], WebArena [30]), enterprise/browser suites (e.g., BrowserGym[3], WorkArena [4], GAIA [16]), remote-sensing [10, 1, 22], primarily score goal completion and response quality, not the safety of intermediate actions. For comprehensive comparison with prior work beyond BFCL, we are currently developing DFA-based implementations tailored to domain-specific benchmarks [1, 17, 20].

6 Conclusion

We introduced *CORE*, a deployment-oriented, *path-based* evaluation framework for tool-using LLM agents. Unlike final-state checks, our method exposes skipped preconditions, compensating action pairs, and redundant or reordered calls, yielding a graded picture of agentic capability rather than pass/fail results. Across diverse worlds, stronger models achieve higher *PC*/*PC–KTC*, lower harmful rates, and better efficiency, while existing evaluation schemes often miss critical mid-trajectory errors.

References

[1] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025.

[2] Emmanuel O Badmus, Peng Sang, Dimitrios Stamoulis, and Amritanshu Pandey. Powerchain: Automating distribution grid analysis with agentic ai workflows. *arXiv preprint arXiv:2508.17094*, 2025.

[3] Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The BrowserGym Ecosystem for Web Agent Research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

[4] 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*, pages 11642–11662. PMLR, 21–27 Jul 2024.

[5] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[6] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[7] M. G. Kendall. A new measure of rang correlation. *Biometrika*, 30(1-2):81–93, 06 1938. ISSN 0006-3444. doi: 10.1093/biomet/30.1-2.81.

[8] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[9] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

[10] Chaehong Lee, Varatheepan Paramanayakam, Andreas Karatzas, Yanan Jian, Michael Fore, Heming Liao, Fuxun Yu, Ruopu Li, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. Multi-agent geospatial copilots for remote sensing workflows. *preprint arXiv:2501.16254*, 2025.

[11] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[12] Ruonan Li, Yang Qin, Jie Liu, Lu Zang, and Jinlong Li. Path planning strategy based on principal component federation for multi-agent in connected vehicles. *IEEE Transactions on Automation Science and Engineering*, 2024.

[13] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[14] Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. *arXiv preprint arXiv:2308.01399*, 2023.

[15] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[16] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

[17] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[18] Varatheepan Paramanayakam, Andreas Karatzas, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. Less is more: Optimizing function calling for llm execution on edge devices. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pages 1–7. IEEE, 2025.

[19] Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, 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 *Forty-second International Conference on Machine Learning*, 2025.

[20] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

[21] Julian Quevedo, Percy Liang, and Sherry Yang. Evaluating robot policies in a world model. *arXiv preprint arXiv:2506.00613*, 2025.

[22] Simranjit Singh, Michael Fore, and Dimitrios Stamoulis. GeoLLM-Engine: A Realistic Environment for Building Geospatial Copilots. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 585–594, 2024.

[23] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*, pages 477–490. PMLR, 2022.

[24] Keyon Vafa, Justin Y Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model. *Advances in Neural Information Processing Systems*, 37:26941–26975, 2024.

[25] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[26] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[27] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[28] Sriram Yenamandra, Arun Ramachandran, Mukul Khanna, Karmesh Yadav, Jay Vakil, Andrew Melnik, Michael Büttner, Leon Harz, Lyon Brown, Gora Chand Nandi, et al. Towards open-world mobile manipulation in homes: Lessons from the neurips 2023 homerobot open vocabulary mobile manipulation challenge. *arXiv preprint arXiv:2407.06939*, 2024.

[29] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095, 2007. doi: 10.1109/TPAMI.2007.1078.

[30] 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 *The Twelfth International Conference on Learning Representations*, 2024.

[31] Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. *arXiv preprint arXiv:2503.24278*, 2025.

A CORE Worlds Overview

Table 3: Simulated agentic worlds: Example tools and tasks.

World	***Tools***	**Tasks (function calls)**
Agentic Farm	*move_to, water_plant, harvest_fruit, ...*	Water plant C with 4.5 liters while keeping moisture within safe limits. Then harvest plant A and deliver the load to the collection bin. Empty the hopper and return to base.
Agentic Arm	*open_gripper, pick, place, ...*	Move to (0.90, -0.10, 0.14, yaw=0.0) and pick 'panel_X'. Rotate yaw to 1.57 rad while above (0.95, 0.20, 0.10) and place it there.
Transactions	*create_account, transfer, deposit, ...*	Create account 'A123', deposit 100, charge a fee of 50, and check the balance.
Web Browsing	*move_to_url, get_page_source, view_browsing_history, ...*	Navigate to 'page2.html', then search for the text: 'Matt then discusses his former job,'.
Automation	*lock_door, turn_on_lights, activate_alarm, ...*	Lock the door and activate the alarm in that order.
Legal Compliance	*check_compliance, flag_violation, approve_policy, ...*	Verify if the statement 'Users must be informed before data collection' adheres to our privacy policy. If it does, approve it as a valid policy statement.
Communication	*forward_message, delete_message, schedule_message, ...*	Send a high-priority message from 'Alice' to 'Bob' with the content 'Urgent meeting at 3 PM'.
CRUD	*add_user, generate_timestamp, update_user_email, ...*	Show all users and update the email of the user with name 'Alice' to 'alice@example.com'. Finally verify the changes were applied.
Desktop Manager	*open_application, perform_action, print_application_actions, ...*	Open 'Terminal', run an $'execute\_command'$ action, and then list all currently open applications.
Event Scheduler	*schedule_event, get_event_time, schedule_recurring_event, ...*	Reschedule 'Team Sync' to '2025-02-10T10:00:00' and check the remaining time until the event.
File Management	*create_file, copy_file, get_file_size, ...*	Search for the word 'agenda' in 'meeting_notes.txt'. If it's not found, append it.
Navigation	*move_up, move_right, get_player_position, ...*	Move the player to the bottom-right corner of the grid (4,4) as fast as possible.
Validation	*validate_email, hash_password, validate_username, ...*	Validate if 'John_Doe' is a proper username, then hash the password 'MyStrongPass!'.
Computations	*add_numbers, multiply_numbers, calculate_average, ...*	Calculate the average of the numbers 10, 20, and 30.
Writing	*add_article, add_verb, add_noun, ...*	Create a sentence consisting of the words: 'runs', 'the', 'dog', 'happy'. Put them in the correct order first.

B Deployment Evaluation with CORE Metrics

We discuss how the proposed five metrics (*Path Correctness* (PC), *PC-KTC*, *Prefix Criticality*, *Harmful–Call Rate*, and *Efficiency*) could better cover the principal axes that matter for deploying tool-using agents: task attainment via an allowed procedure, order and parameter fidelity, safety (incidence and timing of violations), and economy of action.

Standing assumptions. (A1) Each task prompt θ is encoded as a DFA with harmful transitions given by undefined (q, σ); reads are side-effect free. (A2) The golden set $\mathcal{P}_\theta$ is non-empty; progress subgraph is acyclic. (A3) Execution cost/latency is roughly proportional to the number of calls.

Desiderata for deployment.

1. Goal via valid procedure: the action path should align to a harm-free accepting path.
2. Order/parameter fidelity: even when the bag of operations matches, wrong order or near-miss parameters can be unacceptable.
3. Safety — incidence: minimize the number of harmful (out-of-policy) invocations.
4. Safety — causality: earlier harmful invocations are more severe (cascading effects).
5. Economy: avoid redundant reads/benign writes and unnecessary steps.

Coverage claim (informal). Under (A1)–(A3), the tuple

$$\Big(\underbrace{\mathrm{PC}}_{\mathrm{D1}}, \underbrace{\mathrm{PC-KTC}}_{\mathrm{D2}}, \underbrace{\mathrm{HarmRate}}_{\mathrm{D3}}, \underbrace{\mathrm{PrefixCrit}}_{\mathrm{D4}}, \underbrace{\mathrm{Eff}}_{\mathrm{D5}}\Big)$$

is sufficient to detect and quantify every failure mode that can arise from an agent's sequence of calls relative to the DFA.

Table 4: Failure-mode coverage table.

Failure mode	PC	PC - KTC	HarmRate	PrefixCrit	Eff.
Wrong op/parameter	✓	✓			
Right ops, wrong order	partial	✓			
Any harmful invocation			✓	✓	
Early harmful invocation				✓	
Redundant reads/benign writes					✓
Exploration bloat (too many steps)					✓

Aggregation and use. We recommend reporting the vector of scores and using a Pareto view rather than a single scalar. If a single number is required, a task-owner can choose weights that reflect deployment risk (e.g., high weight on PrefixCrit/HarmRate for safety-critical devices, high weight on Efficiency for battery-constrained systems).

Limitations. As discussed previously, we note that completeness holds relative to the DFA abstraction: effects not expressible as state/action symbols (e.g., fine-grained timing within a call, continuous control, or human-facing UX quality) require extending the alphabet. Stochastic environments may also warrant distributional versions of the scores (means/quantiles over rollouts).

C Path-Correctness: Properties and Proofs

Definitions. Let $\mathrm{LD}(x, y)$ be Levenshtein distance and $\mathrm{NLD}(x, y) = \frac{2\ \mathrm{LD}(x,y)}{|x|+|y|+\mathrm{LD}(x,y)}$ its normalized form [29]. Define the *pairwise* similarity

$$s_{\mathrm{PC}}(x, y) \ = \ 1 - \mathrm{NLD}(x, y) \in [0, 1].$$

For a prompt θ, the *aggregated* Path-Correctness score uses the HLR candidate set (§4) and the condensed agent path:

$$\mathrm{PC}_\theta(\mathbf{a}) \;=\; \max_{r \in \mathcal{R}_\theta^{\mathrm{HLR}}(C_\theta(\mathbf{a}))} s_{\mathrm{PC}}\big(C_\theta(\mathbf{a}),\, r\big) \in [0,1].$$

Symbols encode *function name + parameter pattern*; parameter errors are full token mismatches.

Basic properties.

- Range. $0 \le s_{\mathrm{PC}}(x,y) \le 1$ and $0 \le \mathrm{PC}_\theta(\mathbf{a}) \le 1$.
- Perfect match (pairwise). $s_{\mathrm{PC}}(x,y) = 1 \iff x = y$.
- Perfect match (aggregated). $\mathrm{PC}_\theta(\mathbf{a}) = 1$ iff $C_\theta(\mathbf{a})$ exactly equals some $r \in \mathcal{R}_\theta^{\mathrm{HLR}}(C_\theta(\mathbf{a}))$ (in our DAG prompts, this includes the agent-consistent golden path when no harms occur).
- Maximal mismatch (pairwise). $s_{\mathrm{PC}}(x,y) = 0$ iff $\mathrm{LD}(x,y) = |x| + |y|$ (e.g., one of x, y is empty and the other is not).
- Monotonicity under edits. For fixed y, inserting, deleting, or substituting a symbol in x cannot increase $s_{\mathrm{PC}}(x,y)$.

Illustrative examples (pairwise). Let $\mathbf{a}$ denote the condensed agent path and $\mathbf{p}$ a reference.

1. Perfect match: $\mathbf{a} = \texttt{ABC},\ \mathbf{p} = \texttt{ABC} \Rightarrow \mathrm{LD} = 0$, so $s_{\mathrm{PC}} = 1$.
2. Harmless detours vanish under condensation: raw `A R R B` with `R` a read self-loop $\Rightarrow \mathbf{a} =$ `AB`. With $\mathbf{p} =$ `AB`, $s_{\mathrm{PC}} = 1$.
3. Single substitution: $\mathbf{a} = \texttt{ABD},\ \mathbf{p} = \texttt{ABC}$. $\mathrm{LD} = 1 \Rightarrow \mathrm{NLD} = \frac{2}{7} \approx 0.286$, so $s_{\mathrm{PC}} \approx 0.714$.
4. Maximal mismatch: $\mathbf{a} = \varepsilon,\ \mathbf{p} = \texttt{XYZ} \Rightarrow \mathrm{LD} = 3, \mathrm{NLD} = 1$, so $s_{\mathrm{PC}} = 0$.

Metric foundation (pairwise). NLD is a metric.

Theorem. NLD is a metric on Σ^* (non-negativity, identity, symmetry, triangle inequality).

Proof sketch. Let $d(x,y) = \mathrm{LD}(x,y)$ (a metric), and define $f(u,v,t) = \frac{2t}{u+v+t}$ for $u,v,t \ge 0$. Then $\mathrm{NLD}(x,y) = f(|x|,|y|,d(x,y))$. For fixed (u,v), $t \mapsto f(u,v,t)$ is increasing and subadditive: $f(u,v,t_1{+}t_2) \le f(u,v,t_1) + f(u,v,t_2)$. Using $d(x,z) \le d(x,y) + d(y,z)$ and subadditivity gives $\mathrm{NLD}(x,z) \le \mathrm{NLD}(x,y) + \mathrm{NLD}(y,z)$. Other axioms are immediate. □

Similarity triangle (pairwise). Since $s_{\mathrm{PC}}(x,y) = 1 - \mathrm{NLD}(x,y)$ and NLD is a metric,

$$s_{\mathrm{PC}}(x,z) \;\ge\; s_{\mathrm{PC}}(x,y) \;+\; s_{\mathrm{PC}}(y,z) \;-\; 1.$$

Thus the complement dissimilarity $1 - s_{\mathrm{PC}} = \mathrm{NLD}$ is a true metric.

Aggregated score PC_θ basic facts.

- Range. $0 \le \mathrm{PC}_\theta(\mathbf{a}) \le 1$ by construction (max over $[0,1]$).
- Attaining 1. $\mathrm{PC}_\theta(\mathbf{a}) = 1$ iff $C_\theta(\mathbf{a})$ equals an HLR candidate.
- Attaining 0. $\mathrm{PC}_\theta(\mathbf{a}) = 0$ occurs when every HLR candidate is maximally distant from $C_\theta(\mathbf{a})$ (e.g., one is empty and the other non-empty).
- Non-metric. $\mathrm{PC}_\theta(\cdot)$ is a *unary* prompt-level score (a max over references), not a distance between two sequences; metric axioms do not apply to PC_θ itself.