

Beyond Final Success: Semantic Progress Diagnostics for Agent Trajectory Evaluation

Nan Xue^{1*}, Tinghao Wang², Li Tang^{2†}

¹Shanghai Jiao Tong University ²Ant Group

nan.xue@sjtu.edu.cn

wangtinghao.wth@antgroup.com tangli.tl@antgroup.com

Abstract

Large language model agents are increasingly expected to handle complex, long-horizon tasks that require sustained tool use and stateful interaction. Final outcome labels in existing agent benchmarks provide only a coarse view of long-horizon agent execution. They obscure differences in the execution process among successful trajectories and collapse failed trajectories that may have achieved different amounts of task-relevant progress. We introduce the Auditable Semantic Reference Graph (ASRG), a post-hoc framework that represents task-relevant progress with task-specific Semantic Reference Units (SRUs) and matches trajectories against these references. Each SRU is a checkable task milestone with an explicit verifier. Deterministic matching quantifies process coverage and coverage-weighted first-hit timing, while an action-level diagnostic records actions unexplained by the labeling rules. We apply ASRG to 4,740 Toolathlon evaluation trajectories from 22 models across 108 tasks. ASRG reveals process-coverage differences between models with identical final-success rates and first-support timing differences between same-task trajectories with matched coverage. By relating measured progress to resource use, ASRG also supports process-efficiency analysis. ASRG complements final outcome evaluation with auditable, reference-conditioned process diagnostics.

1 Introduction

Large language model (LLM) agents are increasingly evaluated in interactive environments where completing a task requires extended sequences of tool calls, observations, state changes, and artifact creation. Unlike single-turn generation, their execution unfolds over long trajectories whose intermediate decisions affect both what is accomplished

and the resources consumed. As agent benchmarks expand toward long-horizon and stateful settings, evaluating the execution trajectory becomes important alongside evaluating the final outcome.

For a given task in a long-horizon agent benchmark, identical outcome labels can hide substantially different trajectories. Some agents stall early, while others retrieve relevant evidence, invoke appropriate tools, and produce required intermediate artifacts, but still miss a final-state or submission-contract requirement. Outcome-level evaluation captures whether a task was completed, but it does not reveal what task-relevant work was accomplished along the way, when it was accomplished, or where the remaining gap arose. This limitation becomes increasingly consequential as agent benchmarks cover multi-step tool use, web interaction, software applications, mobile or desktop control, and general assistant tasks (Shi et al., 2017; Yao et al., 2022; Zhou et al., 2024; Deng et al., 2023; Drouin et al., 2024; Li et al., 2020; Xie et al., 2024; Rawles et al., 2025; Yoran et al., 2024; Li et al., 2026).

Existing evaluation protocols often provide limited visibility into how task-relevant progress unfolds over an execution trajectory. Consequently, outcome labels can obscure efficiency differences among successful trajectories and collapse early failures, inefficient partial progress, failures after substantial intermediate progress, and ambiguous evidence gaps into the same negative outcome. Prior work has begun to analyze agents through progress measures, trajectory diagnostics, tool-use correctness, process verification, and diagnostic visualization (Lightman et al., 2024; Ma et al., 2024; Ou et al., 2025; He et al., 2026; Deshpande et al., 2025). A remaining gap is a reusable post-hoc diagnostic framework for measuring task-relevant progress across heterogeneous tasks and models. Without such a framework, systematic comparisons of process efficiency among successful trajectories

*Work done during an internship at Ant Group.

†Corresponding author.

and measured progress within failed trajectories remain limited. Resource measures such as token count, turn count, latency, and tool-call count quantify execution cost, but they do not indicate whether that resource use produced task-relevant progress.

A reusable process diagnostic must accommodate multiple valid solution paths without relying on a single reference trace or fixed tool-call sequence. It should describe task-relevant milestones at a level that is both verifiable from recorded evidence and flexible enough to recognize different valid behaviors. It should also support post-hoc analysis of existing trajectories, and make measurements traceable to specific trajectory evidence (Zheng et al., 2023; Lù et al., 2025).

We introduce the Auditable Semantic Reference Graph (ASRG), a framework for diagnosing task-relevant semantic progress in agent benchmark trajectories. ASRG represents each task with Semantic Reference Units (SRUs). An SRU is a task-specific, checkable milestone with an explicit verifier. Together, the SRUs define the task-level process reference for measuring task-relevant progress across trajectories. The SRUs are synthesized from benchmark artifacts and development trajectories reserved for reference construction. The core diagnostics are Progress Coverage (PC), First-Hit Score (FHS), and Unknown Action Rate (UAR), which quantify process coverage, coverage-weighted first-hit timing, and the unexplained-action proportion. A separate judge-assisted layer provides descriptive failure categories for failed trajectories. We instantiate ASRG on 4,740 released Toolathon evaluation trajectories from 22 models across 108 tasks. Aggregate and matched-trajectory analyses show that the diagnostics preserve process variation among trajectories with the same evaluator outcome and support analyses of process efficiency and failed-trajectory progress.

The paper makes three contributions. First, it introduces SRUs as task-specific, checkable references for measuring task-relevant progress in long-horizon agent trajectories. These references connect raw trajectory evidence to final outcome labels and are constructed from benchmark artifacts and designated development trajectories before evaluation trajectories are scored. Second, it develops ASRG as a deterministic trajectory-to-SRU alignment framework that computes process coverage, coverage-weighted first-hit timing, and the unexplained-action proportion. A separate judge-assisted layer supports evidence-bounded

failure analysis. Third, it evaluates ASRG on 4,740 Toolathon trajectories from 22 models across 108 tasks. Claude 4 Sonnet and GPT-5 High have the same final-success rate of 0.3102 but differ by 0.2000 in task-paired process coverage, showing that final-success rates can conceal substantial process differences.

2 Progress-Aware Trajectory Diagnostics

We study a post-hoc diagnostic setting for existing agent benchmarks. Let $\mathcal{T}$ be a set of tasks, $\mathcal{M}$ a set of agent systems, and $\tau_{m,t,i}$ the i -th trajectory produced by agent m on task t . Each trajectory contains agent messages, tool calls, observations, intermediate artifacts, and final responses. The benchmark provides an outcome label $y_{m,t,i} \in \{0, 1\}$ indicating whether the task-level success condition is satisfied.

Outcome labels provide the task-completion anchor. A progress-aware diagnostic layer complements this outcome view by characterizing the execution process through measurements of semantic progress, coverage-weighted first-hit timing, residual action-label coverage, and failure evidence. Together, the outcome label and process diagnostics describe complementary aspects of the same trajectory. The label $y_{m,t,i}$ records task completion, while the process diagnostics characterize how the trajectory progresses relative to a task-specific reference.

For each task t , let $\mathcal{R}_t = \{r_1, \dots, r_k\}$ denote its set of SRUs. Each SRU is a task-specific, checkable milestone with an explicit verifier. Some tasks admit mutually exclusive valid paths. For such tasks, ASRG evaluates global process SRUs together with the branch-conditioned process SRUs of a uniquely activated path. If path activation is ambiguous, it evaluates only global process SRUs. SRUs are defined at the level of task-relevant states or operations, allowing different trajectories to satisfy the same milestone through different valid behaviors. Together, the SRUs form an auditable reference layer for measuring task-relevant progress at the milestone level. Satisfying an SRU indicates that the trajectory reached the corresponding task-relevant state or operation. Progress coverage is defined with respect to the evaluated SRUs in $\mathcal{R}_t$, while task completion remains anchored by $y_{m,t,i}$. Accordingly, full process-SRU coverage means that every process SRU evaluated for that trajectory receives a strict hit.

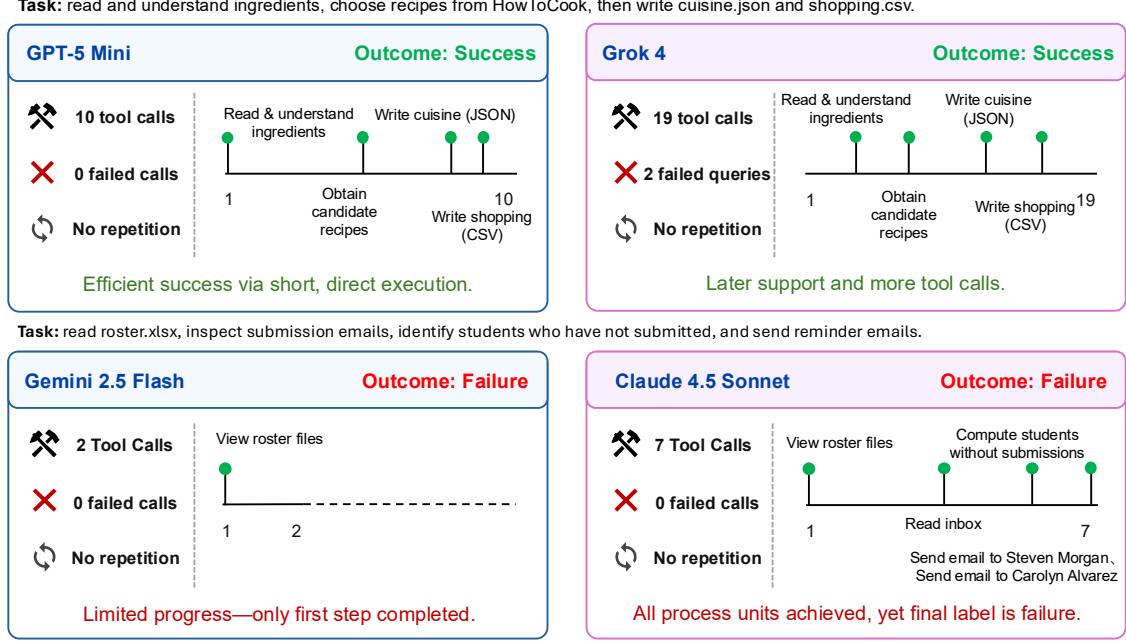


Figure 1: Matched same-task trajectories. Top: both COOKING-GUIDANCE runs succeed and fully cover the evaluated process SRUs, but differ in first-support timing and tool use. Bottom: both COURSE-ASSISTANT runs fail, while covering one versus all four process SRUs. Markers denote first strict support, and line endpoints denote total tool calls. Positions are counted in the ordered tool-call subsequence.

Given $\tau_{m,t,i}$ and $\mathcal{R}_t$, ASRG examines the ordered trajectory evidence to determine which evaluated SRUs receive support. For each SRU, it records the resulting match status and, for a strict hit, the earliest supporting tool-call position. Strict hits support the main progress metrics, while weaker or unresolved statuses preserve uncertainty for sensitivity analysis and audit. The resulting diagnostics summarize strict SRU coverage, coverage-weighted first-hit timing, and the proportion of labelable actions that the deterministic action labeler cannot assign to a known category. For failed trajectories, a separate judge-assisted layer additionally records evidence-supported failure categories.

3 ASRG for Semantic Progress Diagnostics

ASRG operationalizes the diagnostic view in Section 2 as trajectory measurement conditioned on a task-specific semantic reference. Its core diagnostics are PC, FHS, and UAR. This section defines the semantic reference used by these diagnostics and describes the pipeline that constructs the reference, normalizes trajectories, performs strict matching, and computes the metrics. Failure attribution follows metric computation as a separate diagnostic

layer. The metric-masked analysis withholds PC, FHS, and UAR from the attribution judge.

3.1 Formal Setup, Audit Graph, and Diagnostics

For task t , let $B_t = (x_t, \mathcal{A}_t, \mathcal{E}_t^{\text{spec}}, D_t^{\text{ref}})$ collect the task instruction, tool and environment schemas, evaluator specifications, and development evidence reserved for reference construction. The constructor $\mathcal{C}$ maps these inputs to a typed task reference

$$\mathcal{R}_t = \mathcal{C}(B_t) = \mathcal{P}_t \cup \mathcal{T}_t \cup \mathcal{I}_t. \quad (1)$$

Here $\mathcal{P}_t$ contains process SRUs, $\mathcal{T}_t$ contains terminal requirements, and $\mathcal{I}_t$ contains invariants. The process set consists of the global subset $\mathcal{P}_t^{\text{global}}$ and a branch-conditioned subset $\mathcal{P}_{t,b}^{\text{branch}}$ for each $b \in \mathcal{B}_t$. Each reference unit has a unique type, and each branch-conditioned process unit has a unique branch identifier, so the sets in both decompositions are mutually disjoint.

Let $\mathcal{S}_t = \{s_{\text{global}}\} \cup \mathcal{B}_t$ be the scope nodes. Scope nodes and SRUs use distinct node types. ASRG represents the task reference as the typed attributed graph

$$G_t^{\text{ref}} = (\mathcal{S}_t \cup \mathcal{R}_t, \mathcal{E}_t^{\text{scope}}, \phi_t), \quad (2)$$

where $\mathcal{E}_t^{\text{scope}}$ records global or branch membership and ϕ_t stores each SRU's type, predicate, verifier, timing policy, weight, and provenance. For a trajectory τ , let $\mathcal{V}_\tau$ be its normalized events, which use a third node type. Strict matching adds strict-support and weak-candidate relations. The audit edge set is

$$\mathcal{E}_{\tau,t}^{\text{audit}} = \mathcal{E}_t^{\text{scope}} \cup \mathcal{E}_{\tau,t}^{\text{strict}} \cup \mathcal{E}_{\tau,t}^{\text{cand}}.$$

The resulting audit graph is

$$G_{\tau,t}^{\text{audit}} = (\mathcal{S}_t \cup \mathcal{R}_t \cup \mathcal{V}_\tau, \mathcal{E}_{\tau,t}^{\text{audit}}, \phi_t, \psi_\tau). \quad (3)$$

The three relations are derived separately from the audit records. Scope edges record global or branch membership, while strict and candidate edges link SRUs to supporting and weak-candidate events, respectively. The map ψ_τ stores event attributes and order. Construction provenance remains an attributed record rather than a separate dependency-edge type. Thus, the graph is an audit structure over scopes, SRUs, and trajectory evidence, not a task-dependency or reference-plan graph.

Let $\beta_t(\tau) \in \mathcal{B}_t \cup \{\emptyset, \perp\}$ denote a unique activated branch, no activated branch, or ambiguous activation. Writing $b = \beta_t(\tau)$, the scored process set is

$$\mathcal{P}_t^{\text{eval}}(\tau) = \begin{cases} \mathcal{P}_t^{\text{global}} \cup \mathcal{P}_{t,b}^{\text{branch}}, & b \in \mathcal{B}_t, \\ \mathcal{P}_t^{\text{global}}, & b \in \{\emptyset, \perp\}. \end{cases} \quad (4)$$

When branch activation is ambiguous, PC is computed from global process SRUs only. No branch-conditioned coverage value is reported. For $r \in \mathcal{P}_t^{\text{eval}}(\tau)$, let $\mathbf{1}_{\text{hit}}(r, \tau)$ equal one when at least one event or ordered event chain in $\mathcal{V}_\tau$ satisfies every applicable verifier requirement and produces a strict match for r . Otherwise, it equals zero. Only strict matches contribute to the main metrics, while candidate matches are retained for audit and sensitivity analysis. Let $w_r > 0$ be the stored within-task salience weight, normalized inside each metric denominator. Progress Coverage is

$$\text{PC}(\tau, t) = \frac{\sum_{r \in \mathcal{P}_t^{\text{eval}}(\tau)} w_r \mathbf{1}_{\text{hit}}(r, \tau)}{\sum_{r \in \mathcal{P}_t^{\text{eval}}(\tau)} w_r}. \quad (5)$$

PC is scored only over the process SRUs in the task reference. Terminal success from the original evaluator is reported separately.

Let $\pi_{\text{tool}}(\tau) = (c_1, \dots, c_K)$ be the ordered tool-call subsequence of trajectory τ , and let $\mathcal{P}_t^{\text{time}}(\tau) \subseteq \mathcal{P}_t^{\text{eval}}(\tau)$ be the timing-enabled process SRUs. For a strictly hit unit r , $h_r(\tau)$ is the

one-indexed position in $\pi_{\text{tool}}(\tau)$ of the earliest supporting call recorded by the verifier. We refer to this coordinate throughout as the tool-call position. For a multi-event verifier, this is the earliest supporting call in a chain that satisfies all configured requirements. Let $H_t > 0$ be the task reference horizon. It is the median tool-call count in successful development trajectories, or the development median with a minimum fallback of one when no development success is available. The normalized first-hit earliness score $\tilde{s}(r, \tau)$ equals one at the first tool-call position, decreases linearly relative to H_t , and is truncated at zero after the reference horizon. An unhit unit contributes through $s(r, \tau) = 0$.

$$s(r, \tau) = \begin{cases} \tilde{s}(r, \tau), & \text{if strict hit,} \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

$$\tilde{s}(r, \tau) = \max \left(0, 1 - \frac{h_r(\tau) - 1}{H_t} \right).$$

With $Z_t(\tau) = \sum_{r \in \mathcal{P}_t^{\text{time}}(\tau)} w_r$, the trajectory-level First-Hit Score is

$$\text{FHS}(\tau, t) = \frac{\sum_{r \in \mathcal{P}_t^{\text{time}}(\tau)} w_r s(r, \tau)}{Z_t(\tau)}. \quad (7)$$

FHS combines which timing-enabled process SRUs are strictly hit with where their first supporting tool calls occur under the verifier. For a strictly hit SRU, $\tilde{s}(r, \tau)$ decreases monotonically as its first-hit position moves later within H_t , so a larger unit score denotes earlier verifier support. Across trajectories on the same task, a higher FHS indicates earlier weighted first support only when timed-SRU coverage and the strict-hit set are held fixed. This condition is clearest when both trajectories have full process-SRU coverage.

Finally, let $A_\tau = (a_1, \dots, a_{N_\tau})$ be the sequence of labelable normalized action occurrences, where $N_\tau = |A_\tau| > 0$. Let $\mathcal{L}$ be the finite action-label set listed in Table 4, with UNKNOWN $\in \mathcal{L}$. For each $j \in \{1, \dots, N_\tau\}$, let $\ell_\tau(j) \in \mathcal{L}$ denote the deterministic label assigned to action occurrence a_j . Unknown Action Rate is

$$\text{UAR}(\tau) = \frac{1}{N_\tau} \sum_{j=1}^{N_\tau} \mathbf{1}[\ell_\tau(j) = \text{UNKNOWN}]. \quad (8)$$

Higher UAR indicates a larger proportion of actions that the deterministic labeler cannot assign to a known explanatory category. We report it as the unexplained-action proportion.

3.2 Reference Construction

Reference construction begins with LLM-assisted SRU proposal from B_t and normalization into a common schema. Deterministic schema checks and rubric-based semantic review then assess task grounding, observability, verifier support, abstraction over valid action sequences, and held-out isolation. Rule-based repair normalizes flagged fields, after which quality gating either accepts the reference or rejects records with blocking violations. SRUs describe checkable semantic milestones rather than fixed traces, so different trajectories may satisfy a unit through different valid events or intermediate states.

Construction uses only B_t , which contains task instructions, tool schemas, environment context, evaluator summaries or code excerpts, and designated development trajectories. Held-out evaluation trajectories are excluded. Each SRU records its unit type, checkable statement or predicate, scope, weight, and a structured verifier specifying the required observable evidence, among other audit fields. Reference construction is completed before held-out scoring. Appendix A.1 and Table 2 give the complete record and matching schema.

3.3 Trajectory Evidence and Strict Matching

An evaluation trajectory is represented as $\tau = (x_t, z_1, \dots, z_L, o_\tau)$, where x_t is the task input, o_τ is the final response, and each z_j is an observable record such as an assistant message, a tool call with its name and arguments, a tool result or observation, or an artifact update. Its ordered tool-call subsequence is $\pi_{\text{tool}}(\tau) = (c_1, \dots, c_K)$, which supplies the coordinate used by FHS. ASRG normalizes the records into the order-preserving evidence view $\mathcal{V}_\tau$. Fixed branch-activation rules determine $\beta_t(\tau)$.

Within the activated scope, the strict matcher evaluates a candidate event or an ordered event chain. It first applies the configured confidence threshold and the successful-call requirement when that requirement is enabled. It then checks every configured app, tool, argument, entity, output, output-availability, output-grounding, minimum-evidence, and evidence-chain requirement. A configured anti-pattern prevents a hit. A process SRU receives a strict edge only when the candidate supplies task-specific observable grounding and every applicable requirement is satisfied. Weak-candidate, unknown, and unverifiable statuses remain available for audit. Terminal requirements

stay anchored to the benchmark evaluator, while invariants use separate preserved and unknown logic. Strict-hit assignment and the computation of PC, FHS, and UAR are deterministic and do not use an LLM judge.

A deterministic action labeler assigns labels including direct or chain-support progress, terminal support, failure or recovery, verification, redundancy, harmful mutation, and UNKNOWN. These labels define ℓ_τ and are not used as success labels. Appendix A.1 gives the complete matching status rules and action-label priority order.

3.4 Judge-Assisted Failure Attribution

Failure modes may co-occur even though aggregate comparisons require mutually exclusive counts. The judge therefore assigns one primary category for aggregate analysis while retaining secondary categories, supporting evidence, and evidence against the assigned category for qualitative analysis of mixed cases. The metric-masked condition used in the main comparison receives the de-identified task, evaluator outcome, trajectory excerpt, and fixed-length unit and action evidence selected independently of status. It does not receive PC, FHS, UAR, progress bins, model or run identifiers, evidence positions, aggregate event counts, the rule proxy, or metric-derived summaries.

The judge distinguishes an incorrect final artifact or environment state from a finalization-contract error in which required content, schema, delivery, or submission requirements remain incomplete. Blocked or thrashing execution describes a stalled or looping trajectory. An evidence gap records insufficient trajectory evidence for verifying task-relevant completion. Task misframing indicates that the trajectory pursued an incorrect interpretation or goal. The original metric-exposed condition is retained as a sensitivity comparison. Both conditions provide descriptive, evidence-bounded attributions rather than rule-derived labels, human-validated gold-standard labels, or causal failure labels.

Worked example. For COOKING-GUIDANCE, the agent must select three recipes using the available stock, write the exact dish names to `cuisine.json`, and record missing or insufficient ingredients in `shopping.csv`. The evaluated process reference contains four SRUs covering reading `ingredients.csv`, retrieving a recipe, writing `cuisine.json`, and writing

shopping.csv. The strict matcher records support when the ingredient file returns the expected table, the recipe tool returns a named dish and its ingredients, and the filesystem confirms the two structured outputs. These SRUs first receive strict support at tool-call positions (1, 4, 10, 11), yielding $PC = 1$, $FHS = 0.6912$, and $UAR = 0$. The original evaluator nevertheless returns failure. This example shows that full strict coverage of the evaluated process SRUs does not imply benchmark success because the terminal outcome remains evaluator-anchored.

4 Instantiating ASRG on Toolathlon

We instantiate ASRG on Toolathlon, a long-horizon tool-use benchmark with task-specific evaluators and released trajectories (Li et al., 2026). We use the original evaluator labels as the completion anchor and add post-hoc diagnostics for semantic progress, timing, action coverage, and failure attribution.

4.1 Trajectories and Models

We normalize 7,116 released Toolathlon trajectories: 2,376 development trajectories used only for reference construction and 4,740 held-out evaluation trajectories used for all reported results (Li et al., 2026). The evaluation set spans 22 models, 108 tasks, 1,089 successes, and 3,651 failures, with 2,370 available model-task pairs and two runs per pair. Nineteen models cover all tasks, while three miss two tasks each. Trajectory-level analyses keep runs separate, whereas paired model analyses first average runs within each model-task pair. The evaluation trajectories average 36.03 steps, 25.12 tool calls, 9.89 assistant turns, and 1.02 user turns.

4.2 Toolathlon Instantiation Details

Final references contain 6–15 SRUs per task (mean 9.05) and exclude evaluation trajectories. Held-out scoring and action labeling are deterministic. Appendices A and K report implementation and audit details.

5 Semantic Progress and Process Efficiency

We analyze aggregate alignment with final success, process differences between models with identical final-success rates, and matched trajectories using PC, FHS, and UAR.

5.1 Overall Diagnostic Distributions

Across 4,740 trajectories, final success averages 0.2297 and PC averages 0.6365 (median 0.7000). They measure evaluator-defined completion and strict process-SRU coverage, respectively, and are not comparable on a common completion scale. FHS averages 0.4001 (median 0.4122), with model means ranging from 0.2048 to 0.4795. UAR averages 0.1810 over trajectories with at least one labeled action.

5.2 Aggregate Alignment and Trajectory-Level Non-Redundancy

Across the 22 model aggregates, the Pearson correlations of final success with PC and FHS are 0.7424 and 0.6966, while its correlation with UAR is -0.6084 . Across all 4,740 trajectories, the trajectory-level correlations of final success with PC and FHS are 0.2160 and 0.1081. UAR is defined only for the 4,595 trajectories containing at least one labelable action, over which its correlation with final success is -0.2294 . Model-level Spearman correlations show the same pattern, with 0.7520 for success versus PC and 0.6915 for success versus FHS, indicating substantial but imperfect alignment.

Cheap Proxies Are Weak Semantic Substitutes.

At the trajectory level, tool-call count, assistant turns, and total tokens correlate weakly or negatively with final success (-0.0224 , -0.0867 , and -0.0615) and only weakly with PC and FHS (maxima 0.1825 and 0.1231). Model-level patterns are mixed, as detailed in Table 6. Resource use therefore reflects execution scale but is not a substitute for semantic progress.

Cost-Normalized Progress Separates Efficiency from Completion. Table 1 reports absolute performance and median trajectory-level resource-to-progress ratios for the eight models in Figure 2. Gemini 3 Pro Preview and GPT-5.1 use fewer tokens per unit of measured progress than several models with stronger absolute outcomes. Lower token-per-PC and token-per-FHS ratios partially reflect execution efficiency on agent tasks, indicating that more measured semantic progress is obtained with fewer tokens.

5.3 Model-Level Process Differences Beyond Success

Figure 2 preserves the broad outcome ordering but exposes process differences. Claude 4.5 Opus has

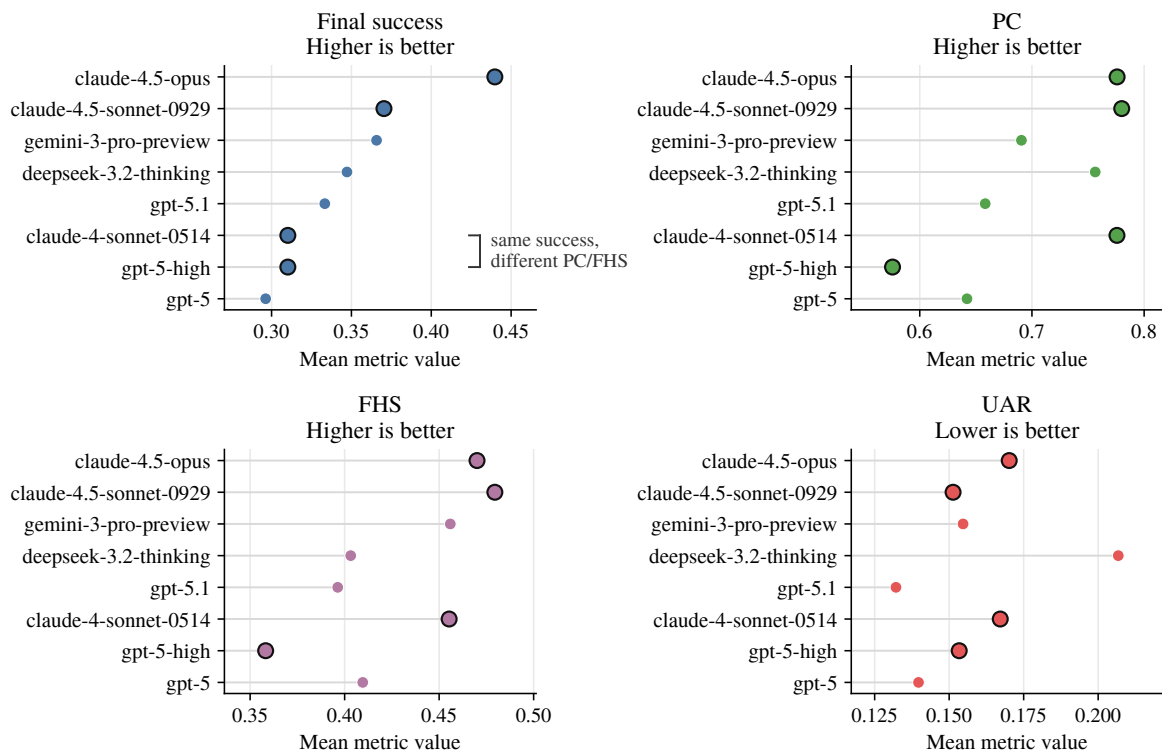


Figure 2: Model-level diagnostic rankings for the top 8 models by final success. All panels share the final-success order. Appendix G reports the full 22-model view.

the highest success, whereas Claude 4.5 Sonnet has the strongest PC/FHS profile. Most clearly, Claude 4 Sonnet and GPT-5 High share a 0.3102 final-success rate but differ by 0.2000 in paired PC. This comparison shows different coverage of the constructed reference without establishing which model is closer to benchmark success. We therefore treat PC, FHS, and UAR as complementary process profiles. Together with resource-to-progress analysis, these diagnostics broaden the evidence available for model selection on agent tasks by allowing outcome quality, semantic progress, and execution efficiency to be considered jointly.

5.4 Trajectory-Level Differences under Matched Task and Outcome

We compare two successful trajectories on COOKING-GUIDANCE to hold both task and evaluator outcome fixed. Both trajectories attain $PC = 1$, but their first-support timing and tool use differ substantially. The top row of Figure 1 plots the tool-call positions at which reading the ingredients, querying recipes, writing the cuisine file, and writing the shopping list first receive strict support.

In reference-SRU order, the GPT-5 Mini trajectory receives first strict support at tool-call posi-

tions (1, 4, 8, 9), uses 10 tool calls, and attains $FHS = 0.7474$. The corresponding positions for the Grok 4 trajectory are (1, 4, 17, 18). It uses 19 tool calls and attains $FHS = 0.4947$. Both trajectories succeed on the same task and fully cover the evaluated process SRUs, but differ in their total tool calls and first-support timing.

5.5 Cross-Benchmark Interface Portability

Beyond Toolathlon, we apply the same reference and alignment interface to the non-visual subset of AgentRewardBench (Lù et al., 2025). A benchmark-specific adapter allows ASRG to construct task-level references and score held-out trajectories from AssistantBench, WebArena, and WorkArena. Appendix C reports PC, FHS, UAR, and repeated-failure diagnostics. This experiment tests interface portability, not cross-benchmark calibration or direct comparability of metric values.

6 Measured Progress in Failed Trajectories

We compare failed trajectories on the same task, summarize the full-coverage endpoint across failures, and examine descriptive failure patterns under metric masking.

Model	Success	Median PC	Median FHS	Median tokens/PC	Median tokens/FHS	Median calls/FHS
Claude 4.5 Opus	0.440	0.714	0.384	867k	1.554M	69.2
Claude 4.5 Sonnet	0.370	0.800	0.395	1.148M	1.938M	67.2
Gemini 3 Pro Preview	0.366	0.588	0.361	510k	0.983M	56.4
DeepSeek 3.2 Thinking	0.347	0.722	0.306	1.616M	3.734M	132.0
GPT-5.1	0.333	0.588	0.300	428k	0.974M	67.2
Claude 4 Sonnet	0.310	0.778	0.377	1.450M	2.866M	61.7
GPT-5 High	0.310	0.500	0.292	601k	1.059M	66.8
GPT-5	0.296	0.588	0.334	640k	1.033M	59.7

Table 1: Absolute outcomes and median trajectory-level cost-per-progress ratios for the top eight models by final success. Ratios are computed only where the corresponding denominator is positive. Eligible counts range from 192 to 215 for tokens per PC and from 189 to 212 for the two FHS-based ratios.

6.1 Trajectory-Level Differences for the Same Task and Failed Outcome

The bottom row of Figure 1 compares two failed COURSE-ASSISTANT trajectories that differ on four process SRUs: retrieving inbox submissions, reading course records, identifying students missing a final presentation, and sending reminders. Gemini 2.5 Flash receives strict support only for reading the course records at tool-call position 2, uses two calls, and obtains $PC = 0.25$ and $FHS = 0.2250$. Claude 4.5 Sonnet receives strict support for all four SRUs at tool-call positions (2, 3, 5, 6), uses seven calls, and obtains $PC = 1$ and $FHS = 0.7000$. Both trajectories receive a failed outcome from the original evaluator.

6.2 Progress Levels among Failed Trajectories

Among 3,651 failures, 747 (20.46% of failures and 15.76% of all trajectories) attain $PC = 1$, indicating full strict coverage of the evaluated process SRUs. From $PC=0$ to $PC=1$ failures, mean FHS rises from 0.0000 to 0.6059 and mean UAR falls from 0.6639 to 0.1101. These endpoint comparisons describe process profiles and do not isolate timing from coverage. Evaluator success remains separate.

6.3 Descriptive Failure Patterns

Under metric masking, wrong-result-or-state and finalization-contract errors together account for 4.98% of $PC=0$ and 90.23% of $PC=1$ failures. The metric-exposed rates are 10.90% and 90.50%. The broad association remains after masking even though fine-grained categories shift. Post-hoc UAR is higher for evidence-gap than finalization-contract cases (0.319, $n = 556$, versus 0.159, $n = 267$, with a task-clustered difference of 0.160). The association is descriptive and does not explain

individual unknown actions. The attributions remain neither human-validated gold-standard nor causal labels.

7 Related Work

Agent benchmarks increasingly supplement final success with intermediate progress signals. Agent-Board derives progress from benchmark-specific state-goal similarity or manually annotated necessary subgoals (Ma et al., 2024). AgentQuest provides a milestone and scoring interface, while each benchmark integration supplies its concrete milestone set and scoring function (Gioacchini et al., 2024). Aghzal et al. derive high-level reference plans from expert annotations and use an LLM postcondition checker to diagnose planning, low-level execution, and replanning (Aghzal et al., 2026). WebCanvas’s Mind2Web-Live verifies path-independent key nodes through URL, element, or value matching, while ToolSandbox dynamically evaluates intermediate and final milestones in stateful tool interactions (Pan et al., 2024; Lu et al., 2025). Other benchmarks operationalize progress with expert landmarks or programmatic checkpoints (Bogin et al., 2024; Xu et al., 2025; Yuan et al., 2026; Al-Kaswan et al., 2026). ASRG does not claim milestone-based progress itself as novel. It addresses a complementary post-hoc setting by constructing frozen, typed, verifier-backed references from benchmark artifacts and designated reference-construction data, then deterministically aligning completed trajectories without requiring a human-authored reference plan, a benchmark-provided intermediate-state progress function, or access to the evaluated agent’s internal planner.

Trajectory-level diagnostics examine execution quality, failure location, and evaluator reliability. TRAJECT-Bench scores tool selection, arguments, dependencies, and execution order, whereas Agent-

Diagnose assesses competencies such as task decomposition, observation reading, self-verification, and exploration (He et al., 2026; Ou et al., 2025). AgentRewardBench compares automatic trajectory judgments with expert labels for success, side effects, and repetition (Lù et al., 2025). Related work analyzes trace reasoning, critical failure steps, hierarchical diagnoses, and trajectory-level preferences (Deshpande et al., 2025; Barke et al., 2026; Zhai et al., 2026; Hong et al., 2026; Wang et al., 2026). ASRG aggregates verifier-supported task-relevant evidence across complete successful and failed trajectories, and adds judge-assisted descriptive categories only after deterministic progress measurement.

Process-supervision and online-guidance methods use intermediate signals to improve training or action selection. Watch Every Step estimates Monte Carlo step rewards against expert-trajectory steps, AgentPRM models future success and local progress for search and reinforcement learning, and Web-Shepherd learns a process reward model from checklist-based step preferences (Xiong et al., 2024; Xi et al., 2026; Chae et al., 2025). Other approaches guide trajectory search or intervene online through principle-based preferences and execution monitoring (Zhang et al., 2026; Wan et al., 2026). Many task or state trackers operate during execution by maintaining or updating the agent’s internal plan or task state. ASRG does not modify the evaluated agent. It analyzes completed trajectories post hoc and can be used to test whether an online intervention changes final success, process coverage, or verifier timing. This separates the reference object used for evaluation from interventions that alter execution.

8 Conclusion

ASRG provides auditable, reference-conditioned measurements of process coverage, coverage-weighted verifier timing, and unexplained actions while keeping evaluator success and judge-assisted attribution separate. On Toolathlon, it reveals process differences between models with identical final-success rates and identifies 747 failed trajectories with full evaluated process-SRU coverage but unmet terminal requirements. Together, these diagnostics support process-aware and resource-aware agent evaluation.

Limitations

Synthesized semantic references. ASRG constructs SRUs through an LLM-assisted reference-construction process over benchmark artifacts and development trajectories. Held-out evaluation trajectories are not used during SRU construction, and the generated references are passed through schema validation and quality checks before evaluation. Nevertheless, the resulting SRUs are not gold-standard human annotations. They may omit valid but less common progress paths, encode imperfect abstractions of task requirements, or include references for which evidence is difficult to verify in some trajectories. The reported PC and FHS values should therefore be interpreted as measurements relative to the constructed reference layer, not as exhaustive coverage of every possible solution path.

Conservative strict matching. The main progress metrics use strict trajectory-to-SRU matching. This design improves reproducibility and reduces over-crediting, but it can undercount implicit progress when a trajectory contains weak, indirect, or incomplete evidence. ASRG retains weak, unknown, and unverifiable statuses for audit and sensitivity analysis, but the main PC and FHS values are conservative estimates of measured semantic progress.

Judge-assisted attribution. Failure attribution is produced by a structured judge over evidence-bounded trajectory packets. The main endpoint comparison uses metric masking, while the metric-exposed condition is retained for sensitivity. The resulting categories are diagnostic rather than human-validated gold-standard or causal labels. Stronger causal claims require larger, multi-annotator, adjudicated validation.

Coarse failure taxonomy. The failure taxonomy is intentionally coarse to support aggregate analysis across many tasks and models. In particular, wrong final result or state aggregates several content-level error types, such as incorrect entities, incomplete artifacts, wrong values, or invalid state mutations. This level of granularity is sufficient for the main analysis of $PC = 1$ failures, but future work could use the retained audit evidence and trajectory evidence to subdivide broad categories into finer-grained failure modes. More generally, applying ASRG to additional agent benchmarks would test how well the same reference-construction and attribution pipeline transfers beyond Toolathlon.

Ethical Considerations

This work analyzes released benchmark trajectories and does not deploy an agent system or interact with end users. The primary ethical risk is interpretive rather than user-facing. Progress scores and failure categories may be mistaken for definitive judgments of model capability or causal responsibility. We therefore frame ASRG as a diagnostic layer rather than a replacement for benchmark evaluators, report PC/FHS/UAR as measurements relative to a constructed reference layer, and treat failure attribution as judge-assisted diagnostic evidence rather than gold-standard human causal labeling.

The analysis uses LLM-assisted reference construction and judge-assisted failure attribution, which may inherit biases or inconsistencies from the underlying models. To mitigate overclaiming, progress metrics are computed through strict matching rather than free-form judgment, and attribution records are backed by inspectable evidence bundles. If diagnostic materials are released, they should preserve the anonymity and licensing constraints of the underlying benchmark trajectories and should not be used for high-stakes deployment decisions without human review.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. OpenAI Codex was used during preparation of the author response and camera-ready manuscript for Chinese–English translation, coding assistance, writing refinement, and LaTeX formatting. Language-model use within ASRG is reported in Section 3 and Appendix A.

References

- Mohamed Aghzal, Gregory J. Stein, and Ziyu Yao. 2026. [Why do LLM-based web agents fail? a hierarchical planning perspective](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32157–32180. Association for Computational Linguistics.
- Ali Al-Kaswan, Maksim Plotnikov, Maxim Hájek, Roland Vízner, Arie van Deursen, and Maliheh Izadi. 2026. [Do agents dream of root shells? partial-credit evaluation of llm agents in capture the flag challenges](#). In *Proceedings of the 3rd ACM International Conference on AI-Powered Software*, pages 349–357. ACM.
- Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. 2026. [AgentRx: Diagnosing AI agent failures from execution trajectories](#). *Preprint*, arXiv:2602.02475.
- Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. 2024. [SUPER: Evaluating agents on setting up and executing tasks from research repositories](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12622–12645. Association for Computational Linguistics.
- Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Han, Taeyoon Kwon, Minju Kim, Beong-woo Kwak, and 2 others. 2025. [Web-Shepherd: Advancing PRMs for reinforcing web agents](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. [Mind2Web: Towards a generalist agent for the web](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114.
- Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. 2025. [TRAIL: Trace reasoning and agentic issue localization](#). *Preprint*, arXiv:2505.08638.
- Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. [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.
- Luca Gioacchini, Giuseppe Siracusano, Davide Sanvito, Kiril Gashteovski, David Friede, Roberto Bifulco, and Carolin Lawrence. 2024. [AgentQuest: A modular benchmark framework to measure progress and improve LLM agents](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 185–193, Mexico City, Mexico. Association for Computational Linguistics.
- Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. 2026. [TRAJECT-bench: a trajectory-aware benchmark for evaluating agentic tool use](#). In *International Conference on Learning Representations*.

- Sirui Hong, Zhijie Liu, Tengfei Li, Wei Tao, Yifan Wu, and Chenglin Wu. 2026. [DiagEval: Trajectory-conditioned diagnosis for reliable software evaluation with GUI agents](#). *Preprint*, arXiv:2605.17439.
- Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng LIU, Zhaochen Su, Yiyang Guo, FAN ZHOU, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, and 2 others. 2026. [The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution](#). In *International Conference on Learning Representations*.
- Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. [Mapping natural language instructions to mobile UI action sequences](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*, pages 39578–39601.
- 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. 2025. [ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183. Association for Computational Linguistics.
- Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stanczak, Peter Shaw, Christopher Pal, and Siva Reddy. 2025. [AgentRewardBench: Evaluating automatic evaluations of web agent trajectories](#). In *Second Conference on Language Modeling*.
- Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. [AgentBoard: An analytical evaluation board of multi-turn LLM agents](#). In *Advances in Neural Information Processing Systems*, volume 37.
- Tianyue Ou, Wanyao Guo, Apurva Gandhi, Graham Neubig, and Xiang Yue. 2025. [AgentDiagnose: An open toolkit for diagnosing LLM agent trajectories](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 207–215, Suzhou, China. Association for Computational Linguistics.
- Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. 2024. [WebCanvas: Benchmarking web agents in online environments](#). *Preprint*, arXiv:2406.12373.
- Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyi Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. 2025. [AndroidWorld: A dynamic benchmarking environment for autonomous agents](#). In *International Conference on Learning Representations*, pages 406–441.
- Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. [World of bits: An open-domain platform for web-based agents](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3135–3144. PMLR.
- Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. 2026. [COM-PASS: Enhancing agent long-horizon reasoning with evolving context](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3360–3380. Association for Computational Linguistics.
- Jiaxuan Wang, Yulan Hu, Wenjin Yang, Zheng Pan, Xin Li, and Lan-Zhe Guo. 2026. [Aligning agents via planning: A benchmark for trajectory-level reward modeling](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23174–23200. Association for Computational Linguistics.
- Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. 2026. [AgentPRM: Process reward models for LLM agents via step-wise promise and progress](#). In *Proceedings of the ACM Web Conference 2026*, pages 4184–4195. Association for Computing Machinery.
- 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. 2024. [OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 52040–52094.
- Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. [Watch every step! LLM agent learning via iterative step-level process refinement](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1556–1572. Association for Computational Linguistics.
- Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang,

Yiqing Xie, and 2 others. 2025. [TheAgentCompany: Benchmarking LLM agents on consequential real world tasks](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. [WebShop: Towards scalable real-world web interaction with grounded language agents](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. [AssistantBench: Can web agents solve realistic and time-consuming tasks?](#) In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8938–8968, Miami, Florida, USA. Association for Computational Linguistics.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, and 17 others. 2026. [OSWorld 2.0: Benchmarking computer use agents on long-horizon real-world tasks](#). *Preprint*, arXiv:2606.29537.

Yuwen Zhai, Runze Li, Liang Wang, Nian Shi, Liwu Xu, Wei Zhang, Ran Lin, Bo Xu, and Benlei Cui. 2026. [GUIDE: Interpretable GUI agent evaluation via hierarchical diagnosis](#). *Preprint*, arXiv:2604.04399.

Yao Zhang, Shijie Tang, Zeyu Li, Zhen Han, and Volker Tresp. 2026. [WebArbiter: A generative reasoning process reward model for web agents](#). In *International Conference on Learning Representations*, pages 123661–123688.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhonghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-bench and Chatbot Arena](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

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. 2024. [WebArena: A realistic web environment for building autonomous agents](#). In *International Conference on Learning Representations*, pages 15585–15606.

A Implementation and Reproducibility Details

This appendix documents how the reported diagnostics are produced and how their claims should be interpreted. It focuses on the implementation details, robustness checks, and audit records needed

to interpret the main PC, FHS, UAR, and failure-attribution results.

Result provenance. All reported TOOLATHLON trajectory-level metrics, aggregate statistics, figures, and failure-attribution labels are computed from a single frozen TOOLATHLON evaluation snapshot. The snapshot covers all 4,740 expected evaluation trajectories, with no excluded trajectories. The diagnostic snapshot includes unit-hit evidence, action labels, event links, attribution evidence, and 108 finalized task references. The targeted manual audit in Appendix K is a separate validation layer over selected trajectories; it assesses metric reasonableness and attribution boundaries but does not modify the reported ASRG scores or labels. The reference-construction boundary is also fixed: SRUs are built in a within-task reference-construction setting, not as a task-held-out generalization benchmark.

LLM-assisted reference construction. The fixed run uses `mimo-v2.5-pro` for LLM-assisted SRU proposal and review. The proposal prompt requires observable task-level milestones, verifier-grounded evidence, and explicit separation from human-certified gold annotation. The review prompt checks whether candidate SRUs are observable, verifier-supported, and sufficient to cover the task objective. Failure attribution uses a separate evidence-grounded prompt over failed-trajectory packets. These LLM components construct references and attribution records, but they do not directly score PC, FHS, or UAR.

A.1 SRU Record and Matching Details

Table 2 gives a representative process-SRU record used by the strict matcher. The example is a process unit because it measures an intermediate milestone rather than final task success. Terminal-goal units use the same record schema, but in this Toolathlon instantiation their hit status is anchored to the original evaluator outcome rather than independently re-verified by ASRG.

The strict matcher assigns one status per SRU using the rules in Table 3. A strict `hit` requires both a selector match and task-semantic grounding. Matching only a generic app, tool, or keyword is insufficient for process scoring. Weak and unresolved statuses are retained in the diagnostic records, but the main PC and FHS metrics count only `hit` process SRUs.

Field	Example value
unit_id	PU_1
unit_type	progress; the unit enters process coverage and first-hit timing.
statement	The agent has read the ingredient table and parsed ingredient types, names, and quantities.
scope	/ global / false / process_timed.
required_for_success	/
timing_policy	
weight	0.15. Weights are positive within-task salience weights assigned during reference construction and normalized inside each metric denominator.
source / confidence	task_instruction, spec_dev_trace / 0.95.
Development support	Development evidence stores the normalized-event positions and one-indexed tool-call positions supporting the unit during reference construction.
verifier.type	tool_event.
verifier selectors	Required app filesystem; required tools include file-reading tools; required argument key path; required argument pattern matches the ingredient-table filename.
verifier grounding	Requires available tool output, an output pattern matching the ingredient-table header, successful tool call, output grounding, minimum evidence 1, confidence threshold 0.85, and anti-patterns for error or missing-file outputs.
scoring evidence fields	Unit-hit records store the one-indexed tool-call position in <code>first_hit_tool_call_index</code> and the corresponding normalized-event position in <code>first_hit_step</code> . FHS uses the former. The latter is retained only for audit linkage. Records also store status, matched event ids, matched tool-call positions, matched evidence types, failed evidence requirements, weak-candidate event ids, and a verifier summary.

Table 2: Representative process-SRU record. The table shows the fields consumed by strict scoring; optional descriptive fields such as construction notes and observable predicates are retained with the record when available.

Status	Rule
hit	At least one candidate event satisfies the verifier confidence threshold, no anti-pattern fires, required tool/app/argument/entity/output conditions hold, required output or state evidence is available when requested, and the matched evidence includes task-semantic grounding. Multi-event chain hits require a grounded chain over usable events.
weak_candidate	Candidate events match shallow selectors, but semantic grounding, output pattern, entity match, evidence-chain requirement, or minimum evidence is insufficient.
unverifiable	Matching requires output or state evidence that is unavailable in the normalized trajectory record.
unknown	No usable evidence is available, event confidence is below threshold, branch activation is ambiguous, or an invariant lacks state or evaluator evidence.
not_hit	Strict matching fails without weak or unverifiable evidence; inactive non-ambiguous branch units are also treated as not hit.
preserved	Invariant status used when evaluator or state evidence supports preservation; otherwise invariants remain unknown in this instantiation.

Table 3: Operational SRU status rules used by strict matching. Terminal-goal units are mapped from evaluator success in this Toolathlon instantiation; process units are matched against structured trajectory evidence.

The task horizon H_t used by FHS is not the maximum, mean, or a percentile over all trajectories. ASRG sets H_t to the median tool-call count among successful development trajectories for the task. If no successful development trajectory is available, it uses the median over development trajectories, with a minimum fallback of one. In the current Toolathlon instantiation, 77 task references use the successful-development median and 31 use the fallback development median or task-budget source.

Action labels are deterministic and single-label. Table 4 gives the priority order. When more than one rule could apply, ASRG assigns the first applicable label in this order and stores the related units or evidence links that triggered the label. This conflict-resolution rule keeps UAR reproducible. UNKNOWN means that none of the deterministic evidence rules explained the action.

Across 119,076 labeled actions in the frozen evaluation snapshot, 20,138 receive UNKNOWN. The mutually exclusive fallback reasons are no rule fired (11,114), no

evidence reuse chain to later SRU hit (4,881), and mutation has no verified invariant violation (4,143). These codes localize why the deterministic labeler abstains. They do not determine whether the underlying action is beneficial, harmful, or omitted from the reference.

B Repeated Reference-Validity Checks

Model-auditor convergence. We conducted a blinded repeated audit using three related model auditors that were not used for reference construction, with three stochastic runs per auditor. For SRU content, each run saw only the task artifacts and frozen SRUs, without evaluation trajectories, automatic matches, PC/FHS/UAR, or prior audits. Across all 635 process and branch-process SRUs, all nine runs judged 627 units meaningful or minor progress and 613 observable or partially observable. In the blinded audit of 197 SRU-trajectory pairs, 99/116 automatic strict hits received at least partial semantic support in all nine runs and 111/116 in at least

Priority	Label family	Condition
1	Direct or terminal support	The action is the first hit, or supporting evidence, for one or more SRUs; successful finalization events tied only to terminal units are labeled terminal support.
2	Failure and repeated failure	Error events are labeled failed; repeated error signatures are labeled repeated failed.
3	Recovery	A recovery event after an observed failure is labeled successful recovery.
4	Verification	New verification facts are labeled valid verification; duplicate verification facts are redundant.
5	Evidence reuse	Search, read, extract, or non-terminal compute events are chain-support progress if their evidence is reused by a later direct SRU hit; duplicate evidence is redundant.
6	Mutation	Mutations linked to violated invariants are harmful mutations; evaluator-confirmed successful finalization mutations are terminal support; otherwise they remain unknown.
7	Other terminal support	Other successful finalization-like events are terminal support when tied to evaluator-confirmed terminal success.
8	Unknown	If no rule applies, the action is labeled UNKNOWN.

Table 4: Action-label priority and conflict resolution. Labels support action-coverage diagnostics only and are not task-success labels.

seven; the five lower-support cases were retained for matcher error analysis. One of these strict-hit pairs was rejected by all nine runs. Two human reviewers independently checked the evidence and conclusions of the GPT-5.6 Sol audit stream and confirmed that they were evidence-supported. Among 65 automatic weak candidates, 31 received full-hit support in all nine runs and 12 received no semantic support in all nine. Because the model auditors are related variants, we treat the broader audit as convergence and error discovery rather than gold strict-hit accuracy. The heterogeneous weak candidates are not promoted into headline PC.

Artifact-bounded omission screen. A material omission is a major, task-grounded, observable process obligation not represented by any process SRU. A majority of the nine audit runs identified no material omission for 103 of 108 tasks. The five remaining tasks and their recurrent candidates are disclosed in Table 5. Auditor calibration differed across model families, so this result is an omission-support profile rather than a completeness rate and does not establish that the other references are exhaustive.

C Cross-Benchmark Portability on AgentRewardBench

Implemented Interface Portability. We test whether the reference and alignment interface of ASRG can run on external agent trajectories by using the non-visual subset of AgentRewardBench (Lù et al., 2025). With a benchmark-specific adapter, the implementation normalizes 1,001 trajectories, freezes 223 task-level references, and scores 446 held-out trajectories from AssistantBench, WebArena, and WorkArena. Another 56 evaluation trajectories remain unscored. This run

has no semantic audit of the transferred references or automatic hits. It therefore supports implemented interface portability, not cross-benchmark calibration or validation of the transferred metric values.

Descriptive Domain Outputs. For transparency, WebArena yields success 0.473, PC 0.533, FHS 0.445, and UAR 0.319. WorkArena yields 0.279, 0.205, 0.183, and 0.704, respectively, while AssistantBench yields success 0.125, PC 0.702, and FHS 0.600. These are descriptive outputs of separately constructed task references. They are not placed on a common calibrated progress scale and are not used to compare semantic quality across domains.

Exploratory Associations with Supplied Annotations. Non-looping trajectories have success 0.526, PC 0.494, UAR 0.387, and a repeated-failure rate of 0.011, whereas looping trajectories have success 0.093, PC 0.280, UAR 0.596, and a repeated-failure rate of 0.042. Under AgentRewardBench optimality labels, complete failures have PC 0.288 and UAR 0.522, while completely optimal trajectories have PC 0.488 and UAR 0.279. Across scored trajectories, UAR is positively correlated with looping ($r = 0.284$) and negatively correlated with optimality ($r = -0.257$); PC and FHS have weaker positive correlations with optimality ($r = 0.175$ and $r = 0.153$). These modest associations are exploratory. Without a semantic audit, they do not validate UAR as a measure of looping, redundancy, inefficient exploration, or behavioral quality.

Reliability-Flagged Subsets. The ambiguous-branch rate is 0.018. UAR is 0.477 overall and 0.121 on the action-reliable subset, where SRU coverage is 0.517, FHS is 0.539, and success is 0.449. The intersection of all reliability flags has $n = 62$, success 1.000, SRU coverage 0.840, FHS 0.500, and UAR 0.088. These conditioned subsets

Task	Recurrent material-omission candidate
CANVAS-SUBMIT-LATE-WORK	Read the candidate submission and substantively compare it with the corresponding Canvas assignment requirements before submission.
DATASET-LICENSE-ISSUE	Trace synthetic data to direct data-source or model licenses and compare derivative or secondary-use permissions.
K8S-PR-PREVIEW-TESTING	Execute the target-branch test script and inspect its results rather than only writing a report.
PRICE-COMPARISON	Read the task-specified <code>internal_pricing_sheet.xlsx</code> rather than substituting a CSV path.
TRAVEL-EXPENSE-REIMBURSEMENT	Inspect invoices item by item and verify both their existence and amount consistency.

Table 5: Five tasks receiving majority material-omission concern in the artifact-bounded screen. The candidates disclose recurrent auditor concerns and are not retrospectively added to the frozen references or headline metrics.

show that the implementation carries its reliability flags across adapters. Since the flags select on evidence availability and the transferred references lack semantic audit, the subsets are not independent estimates of process quality.

D Strictness and Reliability Checks

Numerical-threshold sensitivity. Perturbing every process verifier’s explicit or default event-confidence threshold by -0.05 leaves the strict $PC = 1$ failure count unchanged at 747 of 3,651. Perturbing it by $+0.05$ reduces the count to 722. This check changes only the numerical confidence threshold while retaining the other strict grounding and evidence requirements.

Weak-evidence sensitivity. Weak candidates are related events for which at least one applicable strict grounding or evidence condition is not established. Granting every weak candidate full credit deliberately relaxes those mandatory requirements. Under this permissive sensitivity, mean PC rises from 0.6365 to 0.9527 and 3,421 of 3,651 failed trajectories receive full coverage. This is a weak-inclusive evidence profile, not an alternative estimate of strict PC.

Reliability gates bound stronger readings. Reliability gates are useful as claim boundaries rather than as additional model scores. Action reliability covers 4,214 of 4,740 trajectories (88.90%), branch reliability covers 4,608 trajectories (97.22%), and invariant reliability is much narrower at 1,056 trajectories (22.28%). Their intersection contains 971 trajectories (20.49%). The all-reliable subset has lower UAR (0.0740) and invariant unknown rate (0.0248) than the full set, indicating lower residual action and invariant uncertainty. Because these gates condition on evidence availability, the sub-

set should not be treated as an independent performance estimate; it identifies where action, branch, and invariant diagnostics are most interpretable.

E Proxy Correlation Details

Table 6 reports Pearson correlations between inexpensive trajectory proxies and outcome or progress metrics. The trajectory-level rows use individual held-out trajectories, while the model-level rows use model means over the evaluation split. Model-level correlations are higher for PC than trajectory-level correlations, but the trajectory-level results remain weak, and raw resource proxies do not replace semantic progress diagnostics.

F Uncertainty Intervals

Table 7 reports 95% intervals for the main reported quantities on their original metric scales. UAR has a smaller denominator because action-reuse accounting is available only for trajectories with labeled actions. The $PC=1$ failure and closure rows serve as boundary checks for the analysis of failures with full strict coverage of the evaluated process SRUs. In particular, the broad any-diagnostic closure quantity includes weak proxy closures, whereas the strong closure rate is the conservative quantity used for strict closure claims.

G Task-Paired Rank Checks and a Comparison of Models with Identical Final-Success Rates

We report task-paired uncertainty checks for model-level progress claims. For each model and metric, we first average the two available evaluation runs within each task, then resample tasks with replacement to obtain 95% bootstrap intervals. Rank deltas are computed within each bootstrap sample as final-success rank minus process-metric rank,

Level	Proxy	Success	PC	FHS
Trajectory	Tool-call count	-0.0224	0.1825	0.1231
Trajectory	Assistant turns	-0.0867	0.1648	0.0807
Trajectory	Total tokens	-0.0615	0.1209	0.0300
Model	Mean tool-call count	0.5998	0.7317	0.5464
Model	Mean assistant turns	0.1548	0.5423	0.3591
Model	Mean total tokens	0.4453	0.7796	0.5889

Table 6: Pearson correlations between inexpensive resource proxies and final success, PC, and FHS.

Metric	Estimate	N	95% interval
Success	0.2297	4,740	[0.2177, 0.2422]
PC	0.6365	4,740	[0.6271, 0.6461]
FHS	0.4001	4,740	[0.3935, 0.4063]
UAR	0.1810	4,595	[0.1752, 0.1873]
PC=1 failure rate	0.1576	4,740	[0.1477, 0.1673]
PC=1 strong-closed rate	0.0032	4,740	[0.0017, 0.0049]

Table 7: Uncertainty intervals for reported result quantities.

so positive values indicate that the process metric ranks the model higher than final success. Table 8 reports PC, FHS, and rank-delta intervals for every model.

The comparison of models with identical final-success rates uses paired task-level differences and a sign-flip permutation test. Table 9 shows that Claude 4 Sonnet and GPT-5 High are tied under final success and differ in paired PC. We report the paired FHS difference for completeness, but do not use it to claim earlier model-level timing because process coverage differs between the models.

Three models, GROK-4, GROK-4-FAST, and GROK-CODE-FAST-1, are observed on 106 of 108 tasks. We therefore repeat the model-level analysis on the 106-task complete-case subset. The three incomplete models keep the same success, PC, and FHS ranks under complete-case evaluation. Other changes are adjacent swaps with maximum absolute movement of one rank position. This sensitivity check indicates that the reported rank-change claims are not driven by the two missing tasks.

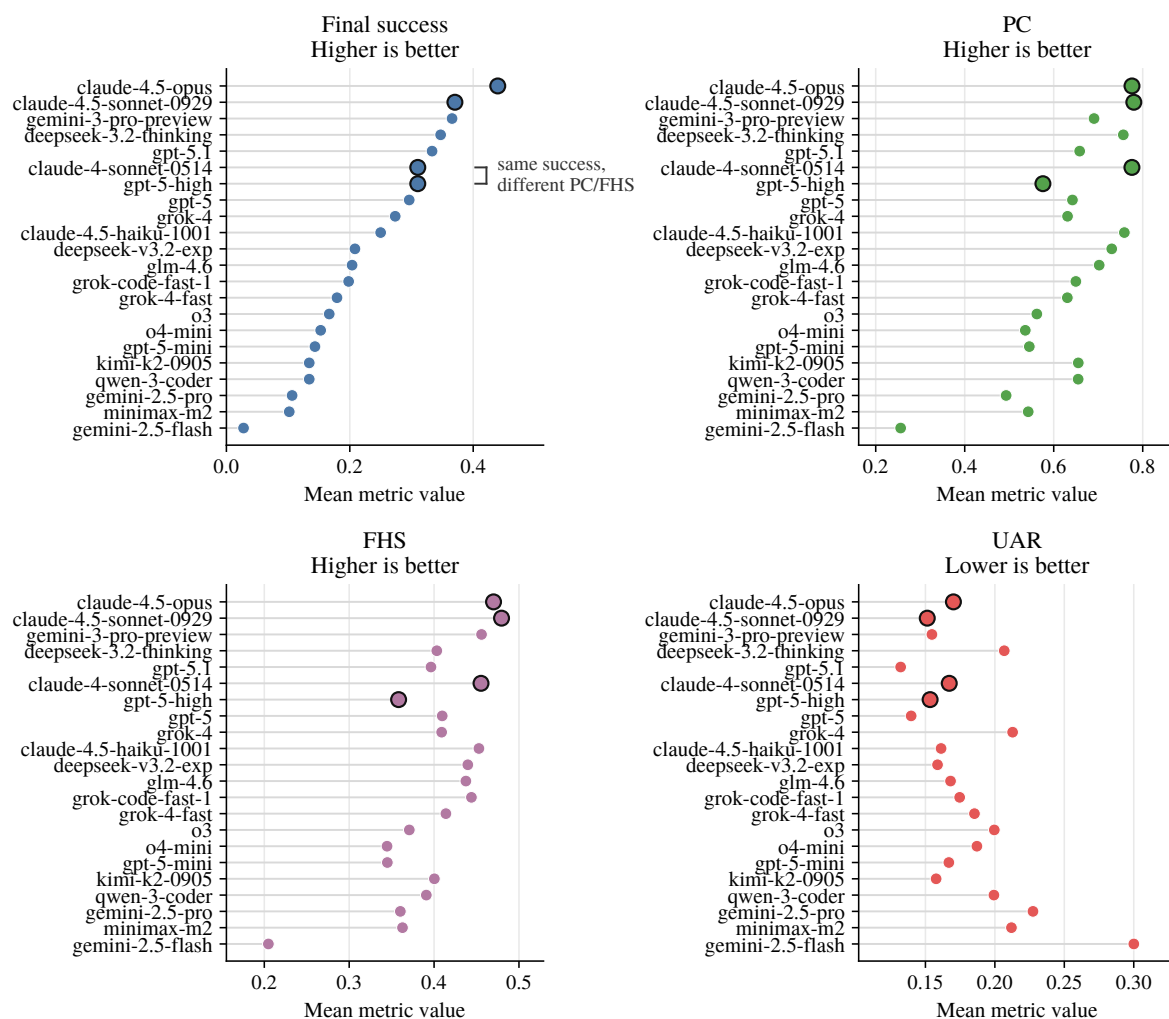


Figure 3: Full 22-model diagnostic ranking view corresponding to Figure 2. Models are ordered by final success rate in all panels. The full view preserves complete model coverage, while the main-text figure shows the top 8 models to improve readability.

Model	PC [95% CI]	FHS [95% CI]	Δ rank PC [95% CI]	Δ rank FHS [95% CI]
CLAUDE-4.5-OPUS	0.7759 [0.7314, 0.8187]	0.4701 [0.4314, 0.5090]	-1.0 [-4.0, 0.0]	-1.0 [-4.0, 0.0]
CLAUDE-4.5-SONNET-0929	0.7800 [0.7372, 0.8213]	0.4795 [0.4478, 0.5123]	1.0 [-2.0, 5.0]	1.0 [-1.0, 5.0]
GEMINI-3-PRO-PREVIEW	0.6906 [0.6464, 0.7348]	0.4559 [0.4228, 0.4900]	-5.0 [-7.0, -1.0]	0.0 [-4.5, 3.5]
DEEPSEEK-3.2-THINKING	0.7564 [0.7113, 0.7995]	0.4032 [0.3670, 0.4395]	-1.0 [-3.0, 3.5]	-8.0 [-12.0, -3.0]
GPT-5.1	0.6583 [0.6069, 0.7059]	0.3963 [0.3630, 0.4310]	-4.0 [-10.0, -1.0]	-9.0 [-13.0, -3.0]
CLAUDE-4-SONNET-0514	0.7758 [0.7343, 0.8168]	0.4553 [0.4205, 0.4912]	3.5 [0.5, 8.0]	2.5 [-2.0, 6.0]
GPT-5-HIGH	0.5757 [0.5220, 0.6269]	0.3582 [0.3222, 0.3956]	-9.5 [-14.0, -7.0]	-12.5 [-16.0, -8.0]
GPT-5	0.6421 [0.5918, 0.6922]	0.4096 [0.3746, 0.4456]	-5.0 [-9.0, -1.0]	-2.0 [-8.5, 1.0]
GROK-4	0.6311 [0.5747, 0.6848]	0.4089 [0.3687, 0.4483]	-5.0 [-9.0, 0.0]	-2.0 [-7.0, 2.0]
CLAUDE-4.5-HAIKU-1001	0.7587 [0.7127, 0.8015]	0.4528 [0.4200, 0.4862]	6.0 [2.0, 9.0]	5.0 [0.0, 9.0]
DEEPSEEK-V3.2-EXP	0.7304 [0.6777, 0.7782]	0.4397 [0.4022, 0.4777]	5.0 [3.0, 10.0]	4.0 [1.0, 10.0]
GLM-4.6	0.7022 [0.6532, 0.7487]	0.4375 [0.3993, 0.4739]	5.0 [2.0, 8.5]	4.0 [2.0, 9.5]
GROK-CODE-FAST-1	0.6498 [0.5979, 0.6971]	0.4440 [0.4073, 0.4799]	1.0 [-3.0, 6.0]	7.0 [2.5, 12.0]
GROK-4-FAST	0.6306 [0.5768, 0.6795]	0.4139 [0.3766, 0.4506]	-1.0 [-3.0, 5.0]	5.0 [-1.0, 8.0]
O3	0.5621 [0.5014, 0.6236]	0.3708 [0.3275, 0.4149]	-2.0 [-7.0, 2.0]	-1.0 [-7.0, 3.5]
O4-MINI	0.5361 [0.4762, 0.5956]	0.3446 [0.3048, 0.3857]	-4.0 [-6.0, 1.0]	-5.0 [-7.0, 1.0]
GPT-5-MINI	0.5452 [0.4813, 0.6086]	0.3449 [0.3058, 0.3839]	-1.0 [-6.0, 2.0]	-3.0 [-7.0, 1.5]
KIMI-K2-0905	0.6552 [0.5971, 0.7117]	0.4003 [0.3607, 0.4418]	8.5 [1.0, 12.0]	5.5 [0.0, 11.0]
QWEN-3-CODER	0.6549 [0.6027, 0.7075]	0.3909 [0.3535, 0.4297]	7.5 [2.0, 11.0]	3.5 [-1.0, 8.0]
GEMINI-2.5-PRO	0.4931 [0.4418, 0.5439]	0.3603 [0.3213, 0.3983]	-1.0 [-5.0, 1.0]	2.0 [-3.0, 6.0]
MINIMAX-M2	0.5427 [0.4875, 0.5948]	0.3628 [0.3257, 0.4006]	2.0 [-1.0, 4.0]	4.0 [-1.0, 6.0]
GEMINI-2.5-FLASH	0.2560 [0.2122, 0.3002]	0.2048 [0.1701, 0.2409]	0.0 [0.0, 0.0]	0.0 [0.0, 0.0]

Table 8: Task-paired bootstrap intervals for model-level PC, FHS, and rank deltas. Positive rank deltas mean that the process metric ranks the model higher than final success.

Metric	Paired diff.	95% CI	Perm. p
Success	0.0000	[-0.0787, 0.0833]	1.0000
PC	0.2000	[0.1529, 0.2473]	< 0.001
FHS	0.0971	[0.0658, 0.1309]	< 0.001

Table 9: Paired task-level comparison for Claude 4 Sonnet minus GPT-5 High over 108 shared tasks.

H Supplemental Diagnostic Views

The supplemental figures provide additional views of the fixed result set. They are intended to make the main diagnostic patterns easier to inspect, not to introduce new benchmark objectives. We group related figures into paired panels so that rank disagreement, failure shifts, and reliability warnings can be read together.

I Terminal-Contract Evidence for PC=1 Failures

Under the metric-exposed sensitivity condition, 194 PC=1 failed trajectories are assigned finalization-contract error, and all records include terminal-contract evidence spans. Table 10 groups these violations by terminal subcategory. The table is not a gold-standard human relabeling of the failures. Instead, it reports the terminal evidence recorded by the diagnostic layer to show that the finalization label is grounded in a missed terminal contract rather than only in a category name.

Representative evidence spans show the boundary between process progress and terminal failure. In one case, the trajectory completes the core communication workflow but omits a required final delivery record. In another, the trajectory produces a structured spreadsheet update, but the submitted layout does not satisfy the required ordering and column contract. In the cooking-guidance case in Section 6, the trajectory retrieves recipe evidence and writes the two required files, but the original evaluator still returns failure. The judge-assisted attribution record describes this case as a terminal-content or finalization issue, but neither that record nor the evaluator-mapped terminal description independently establishes which task clauses were violated. These examples support the interpretation that finalization-contract errors are terminal-contract violations after measured process progress, while preserving the limitation that the terminal layer is evaluator-anchored rather than independently human-verified.

Mixed-cause attribution examples. The frozen attribution outputs retain mixed-cause information in addition to the primary category. For `deepseek-v3.2-exp_2::cooking-guidance`, the primary category is `finalization_contract_error` and the retained secondary category is `schema`. Its evidence spans record that both final files were

written successfully while all three terminal requirements remained unmet. Counterevidence records the correct file paths and broad output structures together with completion of the main measured process steps. A second trajectory, `deepseek-v3.2-exp_3::cooking-guidance`, retains `schema`, `field_or_entity_mismatch`, and `missing_artifact` as secondary categories. These records expose co-occurring descriptive diagnoses.

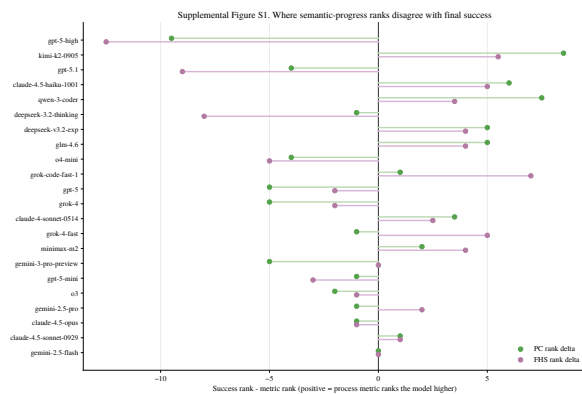
J Model Coverage

The held-out split contains 4,740 trajectories from 2,370 available model-task pairs, with two evaluation runs per pair. Nineteen models cover all 108 Toolathlon tasks. GROK-4, GROK-4-FAST, and GROK-CODE-FAST-1 each miss two tasks. Trajectory-level aggregates use all available trajectories, and task-paired model-level analyses aggregate the two runs within each available pair before task-level comparison. We do not impute missing runs.

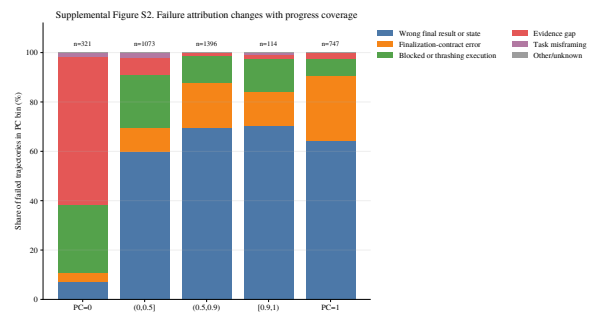
K Targeted Diagnostic Audit

We manually reviewed 60 stratified trajectories from GPT, Claude, and Gemini model families: 15 review-oriented random samples, 25 PC = 1 failures, and 20 high-UAR or evidence-gap cases. The reported proportions support calibration and error discovery. The sample contains 20 trajectories from each model family and covers 56 failed and 4 successful trajectories across 26 tasks. For each failed sample, the review records whether the failure attribution agrees with the trajectory evidence. For all samples, it records whether PC, FHS, and UAR are reasonable, whether the SRU set is adequate for the task, whether action labels are acceptable, and a short diagnostic note.

Table 11 shows that all reviewed PC, FHS, UAR, SRU-quality, and action-label decisions were judged reasonable. Failure attribution agrees in 54 of 56 failed cases. The four successful trajectories have no failure-attribution denominator. The two disagreements are boundary cases rather than metric failures. One PC=1 ADD-BIBTEX failure labeled as finalization-contract error could also be read as wrong final result or state if the BibTeX entry content is incorrect. One high-UAR WOOCOMMERCE-NEW-PRODUCT case labeled as task misframing could also be read as evidence gap

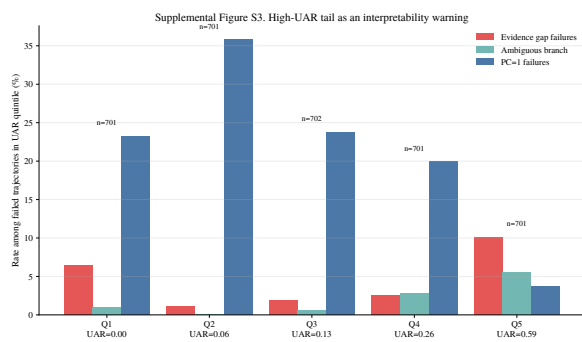


(a) Rank disagreement

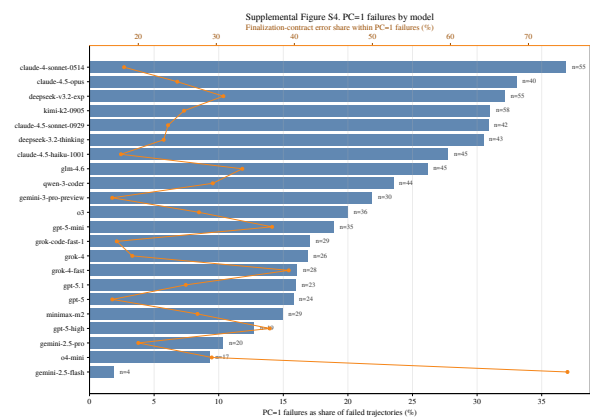


(b) Failure category by PC bin

Figure 4: Supplemental views of rank disagreement and failure-category shifts. Panel (a) shows that process diagnostics can rank models differently from final success. Positive deltas indicate that a process metric ranks a model higher than final success. Under the metric-exposed sensitivity condition, panel (b) shows that evidence gaps dominate $PC=0$ failures (59.81%), while wrong final result or state and finalization-contract errors dominate the $PC=1$ endpoint. Because the categories are judge-assisted rather than human-certified, these views support diagnostic shifts rather than gold-standard causal distributions.



(a) UAR-stratified profiles



(b) $PC=1$ failures by model

Figure 5: Supplemental views of residual action coverage and $PC=1$ failures. Failure-category rates use the metric-exposed sensitivity condition. Panel (a) shows that the highest-UAR quintile has the largest evidence-gap rate (10.13%) and ambiguous-branch rate (5.56%), but a lower $PC=1$ failure rate (3.71%) than lower-UAR quintiles. This is a descriptive UAR-stratified profile and does not identify why individual actions remain unexplained. Panel (b) shows that failures with full strict coverage of the evaluated process SRUs occur across model families. It establishes a recurring scored endpoint, not benchmark near-completion.

Terminal-contract violation	Count	Evidence pattern
Missing final delivery	85	Required submission, message, or delivery record absent after core workflow steps.
Invalid final schema or layout	64	Structured output violates required fields, ordering, or format.
Unspecified terminal gap	27	Terminal goal remains unmet, but the recorded span does not identify a finer subtype.
Numeric or sorting error	11	Final values or ordering violate the task contract.
Required final artifact missing	5	A required final deliverable is absent.
Field or external-state mismatch	2	Final fields or external state disagree with the required target.

Table 10: Terminal-contract evidence among PC=1 failed trajectories assigned finalization-contract error under the metric-exposed sensitivity condition. Counts sum to 194 trajectories.

Audit item	Accepted	Rate
PC reasonable	60/60	100.00%
FHS reasonable	60/60	100.00%
UAR reasonable	60/60	100.00%
SRU quality acceptable	60/60	100.00%
Action labels acceptable	60/60	100.00%
Failure attribution agrees	54/56	96.43%

Table 11: Targeted diagnostic audit outcomes for 60 stratified GPT, Claude, and Gemini trajectories: 15 review-oriented random samples, 25 PC = 1 failures, and 20 high-UAR or evidence-gap cases. The reported proportions support calibration and error discovery.

or wrong final result or state because the agent appears to stop early after empty or incorrect data. The audit is used only to assess metric reasonableness and attribution boundaries; it does not relabel the reported ASRG scores or failure categories.