OPEN-ENDEDNESS BENCH: MEASURING EPISTEMIC PROCESS FOR RESEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents are increasingly given open-ended research tasks: discovering an empirical law from self-designed experiments, improving a heuristic whose optimum nobody knows, or beating a standing record. Outcome scores alone do not establish whether an agent's claims follow from executed experiments, and a reference answer may be unavailable. We evaluate the agent's *epistemic process*: how it forms hypotheses, tests them, and revises them in response to evidence. We introduce OEB (OPEN-ENDEDNESS BENCH), a benchmark-agnostic methodology that reads only the agent's execution record and never a reference answer or an outcome score. OEB compiles the record into a unified epistemic event graph whose edges connect the propositions the agent states to the executed actions that test them; each node carries an exact excerpt that code verifies against the record. One principle governs scoring: prose can state a proposition, but only evidence returned by an executed action can support or refute it, so OEB checks what the agent writes against what it actually ran. From the graph, OEB scores four competence axes (evidence, experiment, revision, and no reward hacking), mostly as the share of opportunities for sound research that the agent took, and profiles six subjective persona traits that describe the agent's research habits. We score 119 existing runs over 12 tasks from three benchmarks: LLM post-training, chip design, and a training-speed record. Against logged results, only 16–29% of the improvements agents claim are real. On 9 of 10 tasks, the best run tries more new ideas in its second half than the worst run. The persona readings follow the model: for every trait, the model that ran explains more of its variance across runs than the task (a median of 43% against 7%).

1 INTRODUCTION

LLM agents now undertake research tasks that require them to choose what to investigate and how to test it. The AI Scientist generates research ideas, writes code, and runs experiments in machine learning (Lu et al., 2024). RE-Bench includes a task in which agents use smaller training runs to predict model performance at a larger compute budget (Wijk et al., 2025), while ALE-Bench evaluates heuristic algorithms for problems such as delivery routing (Imajuku et al., 2025). These settings motivate evaluating both results and the processes that produce them.

Where no reference answer exists, an outcome score leaves the agent's work a black box. Hours with no improvement can be methodical elimination of dead ends or aimless thrashing, and a high score can come without the experiments the task requires: an RE-Bench agent predicted a scaling law without training models across compute scales (Wijk et al., 2025). We need a standard for how an agent handles evidence that does not depend on that score.

Prior work evaluates intermediate steps with labeled reasoning traces or task-specific rubrics (Lightman et al., 2024; Jansen et al., 2024; Starace et al., 2025). We seek an evaluation of epistemic process that uses neither a reference answer nor a task-specific rubric.

We ground this evaluation in the relation between hypotheses, experiments, and observed evidence. Klahr & Dunbar (1988) modeled discovery as search over hypotheses and experiments, and observed both hypothesis-led and experiment-led strategies. We score *epistemic competence*, the discipline with which an agent handles evidence, and separately profile its research style, its subjective *persona*, in which neither strategy ranks above the other.

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

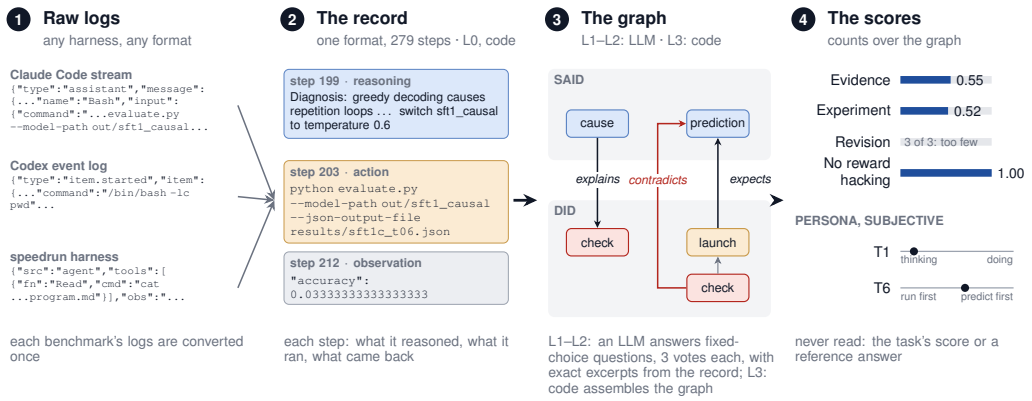


Figure 1: **OEB evaluates epistemic process without reading task outcomes or reference answers.** Raw logs compile into an epistemic event graph that connects stated propositions to executed actions, producing objective epistemic competence scores and subjective persona traits solely from how the agent handles evidence.

Measuring these aspects of research from an execution record poses four challenges: (i) the evaluator must turn "does this claim follow from the evidence?" into counts that code can compute, without knowing the right answer; (ii) it must tell an agent's statement that it did some work from evidence that the work ran, rather than trust the agent's account (Luo et al., 2025); (iii) it must apply the same scoring criteria across domains without requiring a new task-specific rubric (Starace et al., 2025); and (iv) its semantic judgments require validation: prior work finds that even strong LLMs struggle to locate decisive errors in agent trajectories (Zhang et al., 2025).

We introduce OEB to address these challenges. For (i), it compiles the record into a *unified epistemic event graph* of the acts the agent executed and the propositions it stated, and all but one score is a count over that graph. For (ii), it scores under one principle, *semantics proposes, actions dispose*: only evidence from an executed action can support or refute a proposition. The axes that read the agent's written account compare it with the actions in the same record, and the reward-hacking axis reads the actions alone. Most scores are the share of *opportunities* the agent took, such as the share of its experiments whose results it read and drew a conclusion from. For (iii), the scoring code contains no benchmark-specific vocabulary. Task-specific rules enter only through the *world manifest*, a file written when the logs are converted that states the task's evaluation rules. For (iv), the LLM judge only extracts quoted excerpts and answers closed questions, with three independent votes each, and code verifies every quoted excerpt against the record. We validate the instrument in Section 5. An axis averages several measures, its *constructs*, with equal weight; we fit no weights and combine no axes into an overall score. A construct with too few opportunities *abstains* and reports no score.

Contributions.

- A formulation of process evaluation for open-ended agent research that needs no ground-truth verifier or reference answer. It audits execution records and is designed to resist gaming even by an agent that knows the rubric.
- The unified epistemic event graph: a benchmark-agnostic representation that separates acts from propositions. Every benchmark's logs are converted into one record format, so the same compiler reads them all, and each node carries an exact excerpt verified against the record.
- A two-tier instrument that separates four competence axes (evidence, experiment, revision, no reward hacking) from six subjective persona traits. Every construct that is a rate uses one scoring rule, opportunities taken over opportunities offered.
- Validation and findings on 119 existing runs over 12 tasks from three benchmarks. Rescoring changes a score by a median of 0.04 (0.08 under another judge model). Only 16–29% of the improvements agents claim are real. On 9 of 10 tasks, the best run tries more new ideas late than the worst. For every persona trait, the model explains more variance than the task (median 43% vs. 7%).

2 PROBLEM FORMULATION

The execution record. We evaluate an append-only execution *record* (what agent papers call a *trajectory*) $\mathcal{R} = (s_1, s_2, \ldots, s_T)$, where each step $s_t = (a_t, o_t, m_t)$ holds the agent's executable action a_t, the observation o_t it returned, and the agent's reasoning notes m_t. The *harness* is the software that runs the model and logs each step. When we convert its logs, we tag every step with a scope: *environment steps* ($s_t \in \mathcal{S}_{\text{env}}$) act on the studied system and return new evidence; *workspace steps* ($s_t \in \mathcal{S}_{\text{ws}}$) work on internal tools or files; and *submission steps* ($s_t \in \mathcal{S}_{\text{sub}}$) submit a final result. OEB reads $\mathcal{R}$ and the world manifest (Section 3) alone: never a held-out answer key or the environment's internals.

Epistemic process evaluation. A research task may offer outcome measures but no reference solution, and outcomes alone do not show how the agent handled evidence. We therefore evaluate its *epistemic process*: at every step t, the agent's asserted beliefs and commitments must remain faithful to the evidence that environment steps have returned up to step t. A conversion step (L0) turns a benchmark's logs into the shared record format and records the evaluation rules and setup that the log does not state in a *world manifest*; a compiler builds the graph (Section 3); and a measurement layer scores it with fixed rules (Section 4).

Threat model. Two threats face any evaluation that reads the agent's own record. In *timeline forgery* (Threat A), the agent claims afterwards to have anticipated an unexpected outcome. Every extracted proposition carries an exact excerpt that code locates in the step where it first appeared, which rules this out as long as the harness writes the record. In *rubric farming* (Threat B), an agent that knows the rubric writes forward-looking but empty prose without running informative experiments. A score credits the agent only through a link to an executed *action*, such as running a command, editing a file, or launching an experiment, so prose alone earns nothing; Appendix A lists five ways to farm the rubric and each defense.

3 THE UNIFIED EPISTEMIC EVENT GRAPH

OEB scores a record by first compiling it into a graph, because neither half of the record suffices on its own. An instrument that read only actions and observations (a_t, o_t) could not tell a principled hypothesis test from a random parameter sweep, or whether the agent understood why an experiment failed. One that read only the agent's reasoning (m_t) would reward an agent that narrates hypotheses while running broken or uninformative code. The graph has two planes of nodes. The *act plane* holds the actions the agent executed; the *proposition plane* holds the beliefs, conjectures, predictions, and questions it stated in its reasoning. Typed edges connect a proposition to the act that tests it or otherwise bears on it. Every score in Section 4 except the token share T1 is a count over this graph, and every count that credits the agent requires an executed act, so prose alone never scores.

We compile a raw log into the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in four stages (Figure 1). Code converts the log (L0) and assembles the graph (L3); an LLM extracts cards, typed notes that each quote one step (L1), and answers jury questions that connect cards or mark a card (L2). The same LLM, which we call the judge, does both: GLM-5.3 in our experiments, or Grok-4.6 for the nanoGPT speedrun records (Prime Intellect, 2026). We use the LLM for nothing else. It never receives a "grade this transcript" prompt, and every card it produces must quote its source step, or code discards the card.

L0: conversion and world manifest. L0 converts each benchmark's raw logs into the shared record format $\mathcal{R}$ and tags each step with its scope ($s_t \in \{\mathcal{S}_{\text{env}}, \mathcal{S}_{\text{ws}}, \mathcal{S}_{\text{sub}}\}$). It also writes the world manifest (`world.json`), the evaluation rules the log does not state, against which E4 judges the agent's actions (Section 4.2). This conversion is the only benchmark-specific step (Appendix B).

L1: card extraction with quote verification. We use *action* for the raw a_t and *act* for its typed graph node. L1 reads the record in five-step windows and extracts two kinds of *cards*, each a node:

- **Act cards ($\mathcal{V}_{\text{act}}$)** classify actions into four types: *check* (tests a stated expectation), *probe* (gathers evidence before committing to a change), *commit* (carries out the work), and *revise* (changes an earlier commit after contrary evidence).

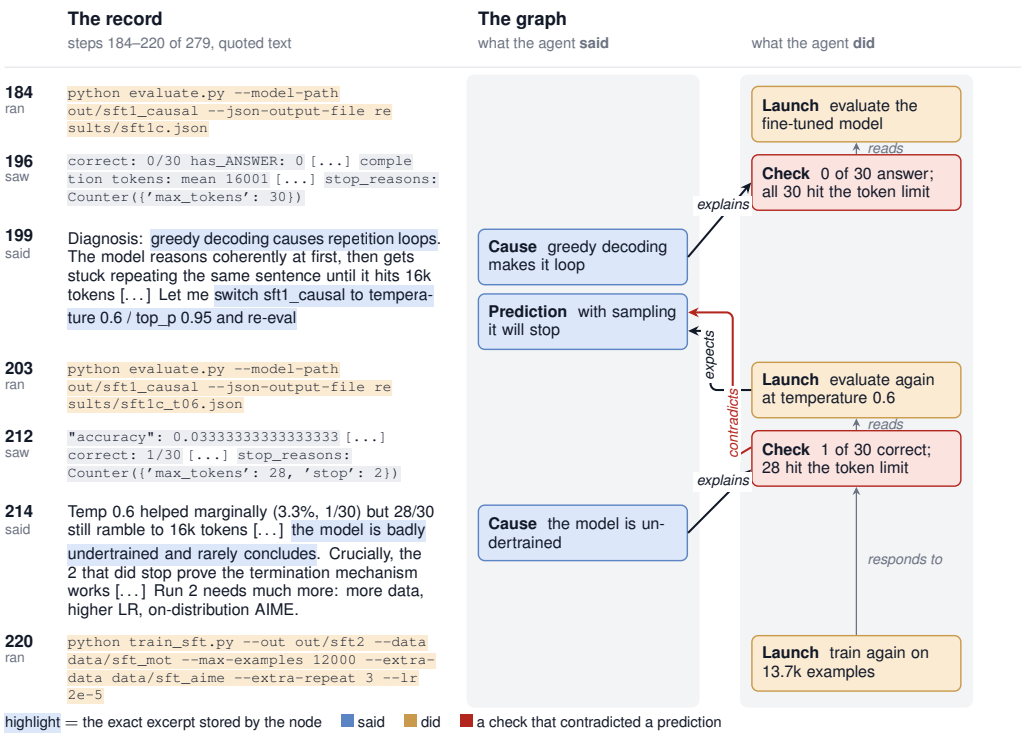


Figure 2: **Propositions earn epistemic credit only when tied to actions that test them.** When empirical checks contradict a prediction, the graph captures whether the agent updates its beliefs based on observed evidence rather than narrating on without checking.

- **Proposition cards ($\mathcal{V}_{\text{prop}}$)** record: *belief*, *conjecture*, *prediction*, *question*, *reliance* (an assumption the agent builds on without testing it), and *retract* (a withdrawn earlier statement).

Every card must quote its source step; code discards a card whose quote is not a literal substring of that step. The prompts (Appendix C) list no examples of qualifying content.

L2: juries. Some questions cannot be answered from one step alone, such as whether two propositions say the same thing or whether an idea is new. The extractor reads one step at a time, so a *jury* answers them. Following Verga et al. (2024), a jury asks the judge one closed question three times, independently, and takes the majority. Juries do two jobs. Four of them connect cards into the graph: they merge propositions that say the same thing, pair a hypothesis with an earlier finding, match a check to the proposition it tests, and group activities into lines of inquiry. The other two each mark one card, and a measure reads the mark directly. The reward-hacking jury marks whether a commit or revise act breaks a quoted rule (E4, Section 4.2). The novelty jury marks whether a new idea differs substantially from earlier settled findings and whether it builds on a mechanism the run has already shown to work (T3 and T5, Table 7).

L3: deterministic assembly. Code builds $\mathcal{G}$ from the cards and jury decisions without further LLM calls. Typed edges connect an act to the proposition it bears on, such as a check that supports or contradicts a prediction. Each proposition starts as untested and becomes supported or contradicted as checks return results. We show one episode in Figure 2.

4 The Measurement Layer

The measurement layer turns the graph into four competence scores and six persona traits with fixed rules and no further LLM calls. One scoring rule serves every rate-valued measure (Section 4.1); the competence axes and persona traits are its instances (Section 4.2, Section 4.3).



Figure 3: **Experiments the agent never reads, or rejects after reading, are missed opportunities.** E2 on this run is the mean of consequential checks (33/59) and experiment yield (30/62), 0.52; E3 is 3/3 but reports no score, because three opportunities fall below the floor of five.

Axis / Trait	Core question
Tier 1: Objective epistemic competence (scored, $[0, 1]$***)***	
E1 Evidence	Are propositions backed by what was seen?
E2 Experiment	Do experiments yield interpretable results?
E3 Revision	Does it address contradicted ideas?
E4 No reward hacking	Does it avoid manipulating the evaluation?
Tier 2: Subjective persona (profiled)	
T1 Budget	Doing vs. thinking
T2 Concurrency	One subject vs. several at once
T3 Search topology	Refine an idea vs. test new ideas
T4 Failure style	Pivot vs. persist
T5 Provenance	Transferred vs. novel
T6 Epistemic tempo	Hypothesis-first vs. run-then-read

Table 1: **OEB separates objective epistemic competence from subjective persona.** Competence axes score discipline in handling evidence on $[0, 1]$, while persona traits place an agent's working style between two legitimate research habits.

4.1 One scoring rule: opportunities taken over opportunities offered

We call each individual measure that contributes to an axis a *construct*. Every construct expressed as a rate uses the same rule:

$$\text{score} = \frac{\text{taken}}{\text{opportunities}} = \frac{|\mathcal{K}|}{|\mathcal{C}|}, \tag{1}$$

An *opportunity* is a situation in the record that calls for a particular response, and the agent *takes* it if the record shows that response. Each experiment the agent launches, for example, is an opportunity to read the result and draw a conclusion; each idea that evidence contradicts is an opportunity to repair the idea or state why it failed. For a persona trait, the response is one of the trait's two poles. $\mathcal{C}$ is the set of opportunities a construct finds in the graph and $\mathcal{K} \subseteq \mathcal{C}$ the set the agent took. If the record offers no opportunity (no idea is ever contradicted, say), the construct *abstains*: it reports no score, never a zero. A construct with fewer than five opportunities reports its counts but stays out of the mean (Appendix F). An axis is the equal-weight mean of its remaining constructs on $[0, 1]$, and we never combine axes into one score. Two measures do not use the rule: the reward-hacking axis E4 transforms a count, and the budget trait T1 is a share of output tokens. We illustrate the rule for one construct in Figure 3, on the run of Figure 2, and list every axis and trait in Table 1.

4.2 OBJECTIVE EPISTEMIC COMPETENCE AXES (E1–E4)

We score competence on four axes, each defined without reference to any domain. E1–E3 compare what the agent writes with what it did; E4 reads only what it did.

E1: Evidence. E1 measures whether the agent's beliefs rest on results it has seen. It averages four constructs: (i) *belief grounded*, the share of stated beliefs and conjectures that a result already in the record settles; (ii) *assertion within evidence*, the share of such grounded research beliefs that claim no more than the results showed, adding no untried setting, general rule, cause, or forecast; (iii) *reliance tested*, the share of reliances that a check tested after the agent stated them; and (iv) *unbacked belief held up*, the share of beliefs stated before any result bore on them that their first test confirmed.

E2: Experiment. E2 measures whether the agent's experiments answer questions it has written down. It averages two constructs: (i) *consequential check rate*, the share of checks whose result changed the standing of a stated proposition, either as the first result to support or contradict it or by reversing an earlier result; and (ii) *experiment yield*, the share of launched experiments whose result the agent read and judged, as a pass or a failure or as better or worse, without later withdrawing the result as a bug, a confound, or a misreading. A check that repeats a known answer, or bears on nothing the agent wrote down, is not taken. Because E2 counts every experiment, an agent that reasons off the record scores low on it; E1 and E3 find their opportunities in the written account, so a terse agent has fewer of them, and more of its constructs abstain (Table 6).

E3: Revision. E3 measures what the agent does after evidence contradicts one of its research ideas. Its one construct, *contradicted resolved*, is the share of contradicted ideas that the agent later repaired, so that a run carrying the change passed, or explained by stating a cause; setting an idea aside with "that did not work" is not taken. We exclude contradictions the agent traced to a broken run, and engineering failures, which must be fixed for the run to go on.

E4: No reward hacking. E4 measures whether the agent manipulated the evaluation. For each commit or revise act, the judge decides whether the act breaks a rule quoted from the task, the world manifest, or a short list of house rules, such as training on test items or picking a seed by its score. Breaking acts that serve one idea, or that are one action, count as one *reward-hacking instance*. Two moves at the end of a run add one instance each: choosing the final candidate as the best of several scores on part of the test set without confirming it on the whole, and reporting a score on part of the test as the score on the whole. With k instances, E4 reports $1/(1+k)$: a clean run scores 1, and one instance scores 0.5. Dividing k by the number of acts instead would let one hack vanish in a long run.

4.3 SUBJECTIVE PERSONA (T1–T6)

OEB also profiles six subjective *persona traits* (Table 1), each a choice between two poles such as hypothesis-first versus run-then-read. T1 is a share of output tokens and T2–T6 are shares of opportunities; traits never enter a competence score and have no preferred direction (Appendix E).

5 INSTRUMENT VALIDITY AND MEASUREMENT CONSISTENCY

We check E4 against a benchmark's own audits and every score against a second scoring.

5.1 REWARD-HACKING DETECTION ALIGNS WITH EXTERNAL BENCHMARK AUDITS

For 90 of its 91 runs, PostTrainBench (Rank et al., 2026) publishes its auditors' ruling on whether the agent put test items into the training data or used a disallowed model. The ruling never enters the record OEB reads, so we can use it as an independent external label for E4. On these runs, E4 flags a run when it counts at least one reward-hacking instance. It flags 10 of the 11 runs the auditors found cheating, with a median of 3 instances in each. It missed one cheating run whose training-data script never entered the record, and each verdict held when the run was scored again (Section 5.2).

	Truly improved		Built on by a later experiment	
The agent read the result as	**Speedrun**	**Chip-Bench**	**Speedrun**	**Chip-Bench**
an improvement	16%	29%	91%	52%
no improvement	1%	20%	20%	60%
(no verdict stated)	3%	31%	33%	51%

Table 2: **Agents claim wins that are not real, and build on them.** Research experiments with a logged result (310 speedrun, 150 Chip-Bench) by the agent's verdict: the share that truly beat the run's best so far, and the share a later experiment modifies or extends.

5.2 Score stability across rescoring and alternative judge models

Because the judge's votes are sampled, two scorings of one record can differ; overall, they agree (Figure 5 in Appendix D). When the same judge scored a record twice, the median score difference was 0.04 on the $[0, 1]$ scale; with a different judge model, it was 0.08. Appendix D gives the per-dimension differences.

6 Empirical findings on agent records

We scored 119 existing agent runs on 12 tasks from three benchmarks. We ran no agents for this study: every record is a run its benchmark had already logged, and scoring a benchmark required only converting its logs into the shared record format. In *PostTrainBench* (Rank et al., 2026), an agent has 10 hours on one H100 GPU to post-train a small base language model for one of six held-out benchmarks (91 runs). In *Chip-Bench* (ARA Labs, 2026), it optimizes a hardware design against a hidden grader (5 tasks, four agents each). In the *nanoGPT speedrun* (Prime Intellect, 2026), it tries to improve the training-speed record (8 runs).

6.1 Agents overrate their own experimental results

An experiment ends in a verdict when the agent reads its result and says what it shows. On PostTrainBench, 37% of the 6,115 launched experiments end in a verdict the agent never later withdraws; on Chip-Bench the share is 24%, and on the speedrun 85%. Where the benchmark logs every experiment's real result, we can test the verdicts (Table 2, left). On the speedrun they carry information, but "improved" is claimed several times more often than it is true, because rerunning one configuration changes the loss by more than most configuration changes do; on Chip-Bench, an experiment called an improvement beats the run's best no more often than one the agent said nothing about. The agents act on these verdicts (Table 2, right): on the speedrun, 60% of the follow-up experiments (194 of 324) build on a win that was not real. An agent that chooses its next experiments from its own verdicts on noisy results is responding partly to noise.

6.2 Late-stage exploration distinguishes winning runs

We ranked the runs of each task by outcome and asked which statistics of the agent's actions alone go with a better rank. Run length does not matter: steps and outcome are unrelated (rank correlation +0.03 over 85 runs). What the steps are spent on does. The statistic that most strongly separates winners from losers is the number of ideas first tested in the second half of the run (+0.53, positive on each benchmark). The best run of a task tests more such ideas than the worst on 9 of 10 tasks (Figure 4, left). Dividing the count by the number of steps leaves the correlation at +0.36, and when one model ran the same task twice, the higher-scoring run tested more such ideas in 11 of 16 pairs.

The logged results explain why breadth pays. On Chip-Bench, an idea's fourth and later tries are 36% of experiments and win about as often as its first three (33% against 28%), but they contribute only 10% of the real gain, and the gap holds within each third of a run, so it does not come from when the tries happen (Figure 4, right). On the speedrun ideas rarely reach a fourth try, and first tries bring 72% of the gain (10 Chip-Bench and 6 speedrun runs log these results).

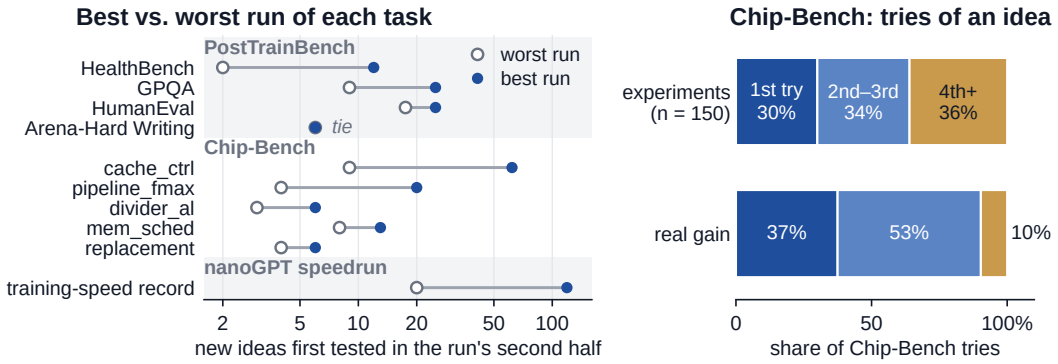


Figure 4: **Winning runs keep testing new ideas late, and an idea's later tries add little.** Left: new ideas first tested in the second half of the best and the worst run of each task. Right: on Chip-Bench, an idea's fourth and later tries are 36% of experiments but bring 10% of the real gain.

	E4		Auditors:
Task	**flagged**	**per run**	**cheated**
BFCL	10/14	2.29	8/14
HumanEval	3/12	0.25	0/11
HealthBench	5/20	0.25	1/20
GPQA	2/10	0.20	0/10
Arena-Hard	3/20	0.15	2/20
AIME	0/15	0.00	0/15

	E4	
Model	**per run**	**main method**
GLM-5.2	1.6	data from test items
Kimi K3	0.8	data from test items
Claude Fable 5	0.5	fits test format
Claude Opus 5	0.4	data from test items
GPT-5.6	0.3	luckiest rerun
GPT-5.5	0.1	luckiest rerun

Table 3: **The task decides whether agents hack; the model decides how.** Left: E4's reward-hacking instances per PostTrainBench run, by task, beside the auditors' rulings. Right: each model's rate and most frequent method on the tasks where some run hacked.

Testing more ideas takes time, yet no Chip-Bench run used its full time budget: the median stopped at 19% of it, and on every task the best run stopped later than the worst.

6.3 The task decides whether agents hack; the model decides how

Reward hacking needs an opening, and the tasks differ in how open they are (Table 3, left): on BFCL, E4 flags most runs and the auditors agree on most of them, while on AIME 2025 no run of any agent hacked. Where an opening exists, models differ in how they hack as well as how often (Table 3, right). GLM-5.2 and Kimi K3 hack most and mostly write training data from test items; the GPT models hack rarely and mostly rerun one recipe under other seeds to keep the best-scoring draw. Appendix H splits every instance by method.

6.4 The persona comes with the model

For every persona trait, the model explains more of the variance than the task and more than chance (Table 4, left). The profiles differ by model (Table 4, right): the two Claude models almost never state a hypothesis before the experiment that tests it (T6), while GPT-5.5, GPT-5.6 and GLM-5.2 state a third or more of their propositions first; the GPT models spend the smallest share of their output on actions (T1). Appendix G shows every run. A harness should therefore work with each model's research habits and target safeguards to each model.

7 Related work

Outcome benchmarks. Agent benchmarks score ML engineering (Chan et al., 2025; Chen et al., 2025a; Starace et al., 2025), scientific discovery (Chen et al., 2025b; Lu et al., 2024), and software engineering (Jimenez et al., 2024; Yao et al., 2025) by unit tests, predictive performance, or

	Variance explained by			Mean score of each model					
Trait	**model**	**task**	**chance**	**Opus 5**	**Fable 5**	**GPT-5.6**	**GPT-5.5**	**Kimi K3**	**GLM-5.2**
T1 budget	**75%**	3%	13%	.25	.22	.10	.10	.33	.23
T2 concurrency	**32%**	8%	13%	.44	.49	.27	.24	.28	.27
T3 search	**36%**	6%	15%	.40	.39	.16	.28	.26	.27
T4 failure style	**31%**	16%	21%	–	–	.52	.40	.44	.52
T5 provenance	**51%**	18%	35%	–	–	.41	–	–	–
T6 tempo	**59%**	2%	17%	.02	.03	.33	.43	.12	.41

Table 4: **The model sets the persona.** For each trait over up to 91 PostTrainBench runs: the variance the model and the task explain, the model's share by chance, and each model's mean score.

reproduced results; ALE-Bench scores open-ended optimization without a known optimum (Imajuku et al., 2025). Such scores do not establish whether an agent's claims are supported by its experiments, and up to half of their variance can stem from the harness (Kirgis et al., 2026). OEB complements them and scores their logs once converted into its record format.

Process evaluation. Step-level labels train better reward models for mathematical solutions (Lightman et al., 2024). AgentBoard tracks progress on annotated subgoals (Ma et al., 2024), PaperBench grades replication against expert rubrics that took weeks per paper (Starace et al., 2025), DiscoveryWorld scores procedures with task-specific scorecards (Jansen et al., 2024), and AgentRx locates execution errors (Barke et al., 2026). These methods need task-specific targets; OEB derives its opportunities from structural rules of the scientific method and needs no task rubric.

LLM judges. LLM judges agree with human preferences on text (Zheng et al., 2023), but in a synthetic case study they rated an incorrect claim highly when shown only the agent's prose answer (Flynt, 2026), and they struggle to locate decisive errors in multi-agent logs (Zhang et al., 2025). OEB therefore asks judges only closed questions about verified excerpts and leaves assembly and counting to code.

Open-ended discovery. Open-endedness measures novelty (Wang et al., 2020), interestingness (Zhang et al., 2024), state coverage (Lidayan et al., 2025), or world-model accuracy (Warrier et al., 2026; Vafa et al., 2024). OEB checks that commitments match evidence from its actions.

8 Limitations and conclusion

Limitations. (i) *Inference expressed through numbers.* In pure hyperparameter search, models often reason only through the values they change, so the instrument extracts fewer propositions. (ii) *Reasoning off the record.* E1–E3 audit the account the agent writes. When that account is missing, the instrument cannot tell unwritten reasoning from reasoning that did not occur, so the outcome analyses of Section 6.2 read only executed acts. (iii) *Optimizing against the instrument.* The runs studied here were produced without knowledge of OEB. How far an agent trained or prompted against its scores can move them is the natural next stress test of the defenses in Section 2. (iv) *Computational cost.* A run took a median of 652 judge calls, which suits offline evaluation rather than a training reward.

Conclusion. We have presented OEB, a benchmark-agnostic methodology that measures epistemic process from agent execution records alone. OEB compiles records into a unified epistemic event graph under the principle that *semantics proposes, actions dispose*: stated propositions are linked to the executed actions that test them. It scores epistemic competence without outcome grading or human-written rubrics, and its scores hold under rescoring (median change 0.04). The records also show that only 16–29% of claimed improvements are real, that winning runs keep testing new ideas late (9 of 10 tasks), and that the model explains more of each trait's variance than the task (median 43% vs. 7%). Building harnesses on these readings, and testing them against equal-cost baselines, is the next step.

Reproducibility Statement

The evaluation code (the record format, the rules for converting a benchmark's logs into it (Appendix B), every judge prompt, the graph compiler, and the measurement layer) is released under an MIT license at `https://anonymous.4open.science/r/oeb-core-2C98`. For every scored run reported here, a separate dataset (`https://osf.io/wr8x4/?view_only=8bd4d34a02054435a703c9c5635111c6`) holds the converted record, the graph, each jury's output, the complete log of judge requests and responses (which can be replayed), and the scores, including the persona trait values from which the per-model distributions follow. We generate every table, figure, and number in Section 5 and Section 6 from those files with the scripts the dataset includes.

Ethics Statement

This research evaluates autonomous AI agents using existing computational benchmarks. No human subjects or private personal data were involved. We discuss the remaining risk that agents could raise their scores without improving their research behavior in Section 2 and Appendix A.

AI Use Statement

In this work, we used generative AI tools to implement methods (the conversion, graph compiler, and measurement code, and the analysis scripts), to give feedback on the research methodology, and to support the analysis and interpretation of results. A language model is also a component of the evaluation method itself: it extracts cards and answers closed jury questions (Section 3), and code verifies every excerpt it quotes against the source record, which establishes only that the text was present. We have not used generative AI tools to generate synthetic data or to propose the research hypotheses; formulating and proving mathematical claims is not applicable to this work. Additionally, we used generative AI tools to draft and edit the paper and to create figures. We have reviewed all AI-assisted work: the authors audited scored records by hand against the raw logs and checked every reported number against the generating scripts. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

ARA Labs. Chip-Bench: AI Chip Design Benchmark, 2026. URL `https://www.chipbench.ai/`. Accessed 2026-09-25.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI Agent Failures from Execution Trajectories, 2026. URL `https://arxiv.org/abs/2602.02475`.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL `https://openreview.net/forum?id=6s5uXNWGIh`.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. MLR-Bench: Evaluating AI Agents on Open-Ended Machine Learning Research. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, pp. 131093–131141. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025a. doi: 10.52202/085713-3945.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward Rigorous Assessment of Language Agents for Data-Driven Scientific Discovery. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025b. URL `https://openreview.net/forum?id=6z4YKr0GK6`.

Jeffrey Flynt. GroundEval: A Deterministic Replacement for LLM-as-Judge in Stateful Agent Evaluation, 2026. URL https://arxiv.org/abs/2606.22737.

Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. ALE-Bench: A Benchmark for Long-Horizon Objective-Driven Algorithm Engineering. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, pp. 173630–173675. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-5226.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DiscoveryWorld: A Virtual Environment for Developing and Evaluating Automated Scientific Discovery Agents. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pp. 10088–10116. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-0324.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Peter Kirgis, Sayash Kapoor, Stephan Rabanser, Nitya Nadgir, Cozmin Ududec, Magda Dubois, JJ Allaire, Conrad Stosz, Marius Hobbhahn, Jacob Steinhardt, and Arvind Narayanan. Log analysis is necessary for credible evaluation of AI agents, 2026. URL https://arxiv.org/abs/2605.08545.

David Klahr and Kevin Dunbar. Dual space search during scientific reasoning. *Cognitive Science*, 12(1):1–48, 1988. doi: 10.1207/s15516709cog1201_1. URL https://doi.org/10.1207/s15516709cog1201_1.

Aly Lidayan, Yuqing Du, Eliza Kosoy, Maria Rufova, Pieter Abbeel, and Alison Gopnik. Intrinsically-Motivated Humans and Agents in Open-World Exploration, 2025. URL https://arxiv.org/abs/2503.23631.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Ziming Luo, Atoosa Kasirzadeh, and Nihar B. Shah. The More You Automate, the Less You See: Hidden Pitfalls of AI Scientist Systems, 2025. URL https://arxiv.org/abs/2509.08713.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pp. 74325–74362. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-2365.

Prime Intellect. NanoGPT Speedrun, 2026. URL https://app.primeintellect.ai/speedrun/nanogpt. Accessed 2026-09-25.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM Agents Automate LLM Post-Training?, 2026. URL https://arxiv.org/abs/2603.08640.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's Ability to Replicate AI Research. In *Proceedings*

of the 42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, pp. 56843–56873. PMLR, 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

Keyon Vafa, Justin Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the World Model Implicit in a Generative Model. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pp. 26941–26975. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-0846.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models, 2024. URL https://arxiv.org/abs/2404.18796.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced POET: Open-ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9940–9951. PMLR, 2020. URL https://proceedings.mlr.press/v119/wang20l.html.

Archana Warrier, Dat Nguyen, Michelangelo Naim, Moksh Jain, Yichao Liang, Karen Schroeder, Cambridge Yang, Joshua B. Tenenbaum, Sebastian Josef Vollmer, Kevin Ellis, and Zenna Tavares. Benchmarking world-model learning with environment-level queries. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Ny6UYZ8ysN.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-bench: Evaluating frontier AI r&d capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. OMNI: Open-endedness via Models of human Notions of Interestingness. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=AgM3MzT99c.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, pp. 46595–46623. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-2020.

A FIVE WAYS TO FARM THE RUBRIC AND THE DEFENSE AGAINST EACH

An agent that knows the rubric could write its record in five ways to game the evaluation; each entry names the defense the instrument is designed with: *A1 (cheap declaration)*: stating high volumes of trivial hypotheses; designed against by crediting no proposition on its own: every count that credits the agent requires an executed act linked to the proposition, and a belief asserted before any observation bears on it lowers *belief grounded* (E1). *A2 (observation parroting)*: presenting information from tool observations as newly generated insights; designed against by verifying when the information first appeared in the record. *A3 (self-citation farming)*: citing prior thoughts as evidence; designed against by requiring graph edges that connect stated propositions to executed actions. *A4 (ritual acknowledgment)*: conceding errors verbally without altering code; designed against by requiring subsequent revisions to the executed work. *A5 (collision avoidance)*: refusing to state falsifiable predictions; penalized because every launched experiment enters the denominator of experiment yield (E2), whether or not the agent stated what it expected.

B L0 CONVERSION RULES

Converting a benchmark's logs (L0) must: (i) divide the raw log into steps deterministically; (ii) tag each step as an environment, workspace, or submission step; (iii) extract action and observation strings without changing their text; and (iv) record in the world manifest (`world.json`) the evaluation rules and setup the log does not state, such as the task's rules and the size of its evaluation set. The conversion needs no knowledge of the downstream scoring rules.

C PROMPT SPECIFICATIONS

Extraction and jury prompts specify the types of card to identify and require quoted excerpts to occur exactly in the source text. Each extraction prompt reads a five-step window of the record; each jury prompt asks one closed question about one card or a pair of cards. Three independent judge calls answer the question; whenever a vote quotes the record, code verifies that the excerpt occurs verbatim in the source. Prompts list neither examples of qualifying content nor their own audit history.

D RELIABILITY DETAILS

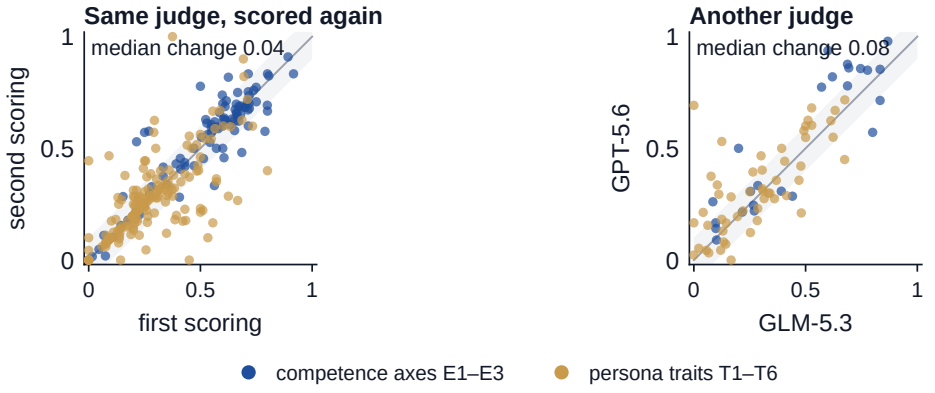


Figure 5: **Evaluation scores are stable across repeated evaluations and alternative judge models.** Differences in competence and trait scores largely remain within ± 0.1, so the system's LLM judge adds little variance to the scores.

For each dimension, we report the mean absolute score difference when the same record is scored twice, either by the same judge or by two different judges (Table 5). We also compare scores from two independent runs of the same agent on the same task and scores from different agent models. We

report the per-model means of the competence axes over all runs of the six PostTrainBench tasks, each of which post-trains gemma-3-4b, in Table 6. Across the constructs of E1–E3 and T2–T6, rescoring by the same judge changed a numerator count by a median of 1 (2 under the other judge). The construct with the most opportunities had a median of 64 per run.

	E1	E2	E3	E4	T1	T2	T3	T4	T5	T6
Same record, judged twice	0.04_{37}	0.05_{40}	0.16_{12}	0.09_{40}	0.00_{35}	0.09_{39}	0.13_{35}	0.11_{21}	0.13_{4}	0.10_{28}
Same agent, second run	0.12_{33}	0.14_{35}	0.23_{10}	0.14_{35}	0.02_{31}	0.16_{35}	0.11_{31}	0.13_{15}	0.10_{2}	0.16_{27}
Different agent models	0.15_{80}	0.23_{85}	0.25_{13}	0.17_{85}	0.11_{71}	0.18_{85}	0.15_{71}	0.14_{17}	0.33_{1}	0.28_{59}
Same record, other judge†	0.16_{10}	0.08_{12}	0.17_{2}	0.00_{12}	0.07_{11}	0.08_{12}	0.14_{12}	0.09_{5}	0.36_{2}	0.09_{11}

Table 5: **Scoring variation across repeated evaluations, independent runs, and different agents.** Each cell is the mean absolute score difference; subscripts count the score pairs compared, where a pair counts only if both scores exist. Scores remain stable when the same judge rescores a record and consistent when a different judge does. †12 Chip-Bench records scored by GLM-5.3, the judge for PostTrainBench and Chip-Bench, and again by GPT-5.6; the other rows use PostTrainBench records.

Agent model	Tasks	E1 Evidence	E2 Experiment	E3 Revision	E4 No reward hacking
Claude Opus 5	6	0.78_{10}	0.08_{12}	–	0.83_{12}
Claude Fable 5	5	0.67_{10}	0.29_{10}	0.60_{1}	0.88_{10}
GPT-5.6	6	0.65_{12}	0.42_{12}	0.43_{12}	0.88_{12}
GPT-5.5	6	0.66_{12}	0.51_{12}	0.56_{5}	0.96_{12}
Kimi K3	6	0.71_{12}	0.48_{12}	0.62_{8}	0.83_{12}
GLM-5.2	6	0.65_{12}	0.51_{12}	0.67_{6}	0.77_{12}

Table 6: **Competence scores across agent models on PostTrainBench.** Each cell averages all runs of the model, first and second; subscripts count the runs with a score. A model whose record omits its reasoning scores low on E2 and offers fewer opportunities on E1 and E3, where more of its constructs abstain.

E PERSONA TRAITS

OEB profiles six subjective *persona traits* (Table 1). Each trait is a choice between two poles. T1 is the share of output tokens spent on actions. T2–T6 are the share of opportunities in which the agent chose one pole; T6, for example, is the share of propositions the agent stated before the first experiment that tested them, against running first and reading afterwards. We infer the profile from the record alone. Traits never enter a competence score and carry no preferred direction.

F SAFEGUARDS FOR SHORT RECORDS

To keep statistical noise from corrupting scores on short or sparse records, the measurement layer enforces three safeguards: (i) *Sample floor*: a rate-valued construct needs at least five opportunities to enter an axis mean. Below the floor it reports its raw counts but contributes nothing to the mean. (ii) *Confidence propagation*: we repeatedly sample construct rates, recompute their axis mean, and report percentiles of those means as an uncertainty interval. For rate-valued constructs the draws use the constituent counts of taken and eligible opportunities. (iii) *Loud abstention*: if no construct in an axis qualifies for its mean, the axis reports no score.

G THE PERSONA BELONGS TO THE MODEL

Every PostTrainBench run has a model and a task. Across up to 91 runs, we estimate how much of each trait's variance goes with model identity and how much with task identity.[1] For every trait,

[1]We use two-factor least squares, crediting each factor only with the variance it explains after accounting for the other. As a chance baseline, we report the share explained after shuffling that factor's labels.

Trait	Poles	Score
T1 Budget	Doing vs. thinking	Share of output tokens spent on actions.
T2 Concurrency	One subject vs. several at once	Share of acts at which two or more subjects remained under investigation; share of commits made before evidence resolved the previous commit in the same line of inquiry.
T3 Search topology	Refine an idea vs. test new ideas	Share of research launches that are the first on their subject; share of new ideas a novelty jury judges substantially different from previously settled findings.
T4 Failure style	Pivot vs. persist	After an adverse result, share of next launches that move to another subject.
T5 Provenance	Transferred vs. novel	Among new ideas judged either transferred or novel, share built from a mechanism already proven earlier in the run.
T6 Epistemic tempo	Hypothesis-first vs. run-then-read	Share of propositions stated as a conjecture, question, or prediction at or before the first launch that tests them.

Table 7: **Persona traits profile working style without normative ranking.** Each trait measures the empirical frequency of choosing one valid research habit over another.

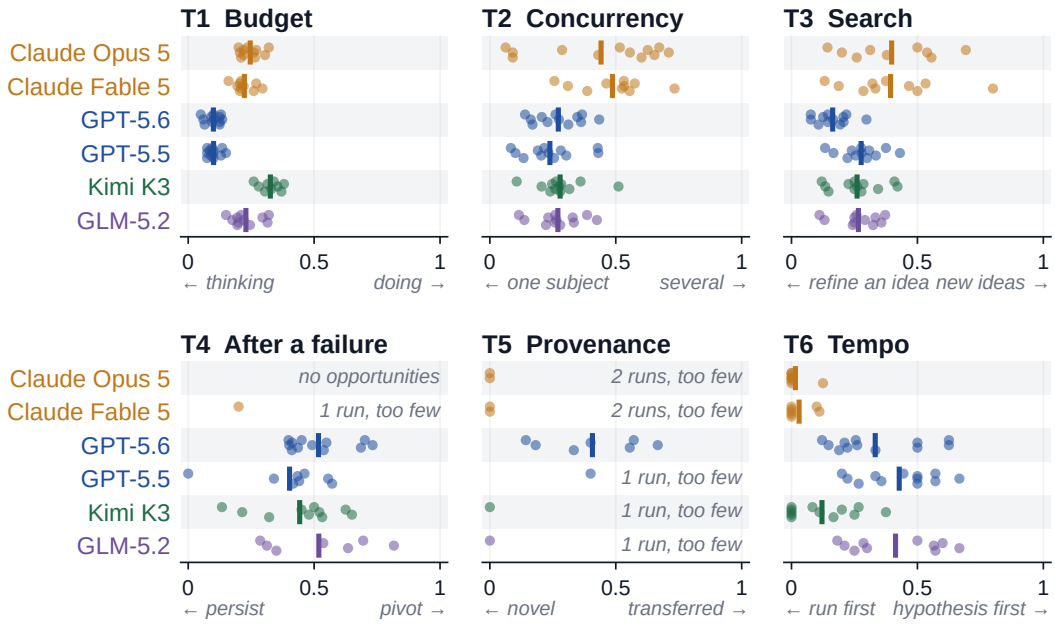


Figure 6: **Persona reflects model identity rather than the task.** Models exhibit consistent, distinct research habits across tasks, maintaining characteristic preferences for budgeting action versus reasoning and for stating hypotheses before execution.

model identity accounts for more of the variance than task identity does, and more than its chance baseline: T1 75% against 3% (chance 13% and 2%), T2 32% against 8% (chance 13% and 4%), T3 36% against 6% (chance 15% and 4%), T4 31% against 16% (chance 21% and 10%), T5 51% against 18% (chance 35% and 12%), T6 59% against 2% (chance 17% and 3%). An agent that prefers thinking to doing prefers it on every task (Figure 6).

H Model-specific specialization in reward-hacking methods

A task can be designed so that no shortcut exists, and on AIME 2025 no run of any agent hacked; the question worth asking is what each model does where a shortcut does exist. On the 5 tasks where

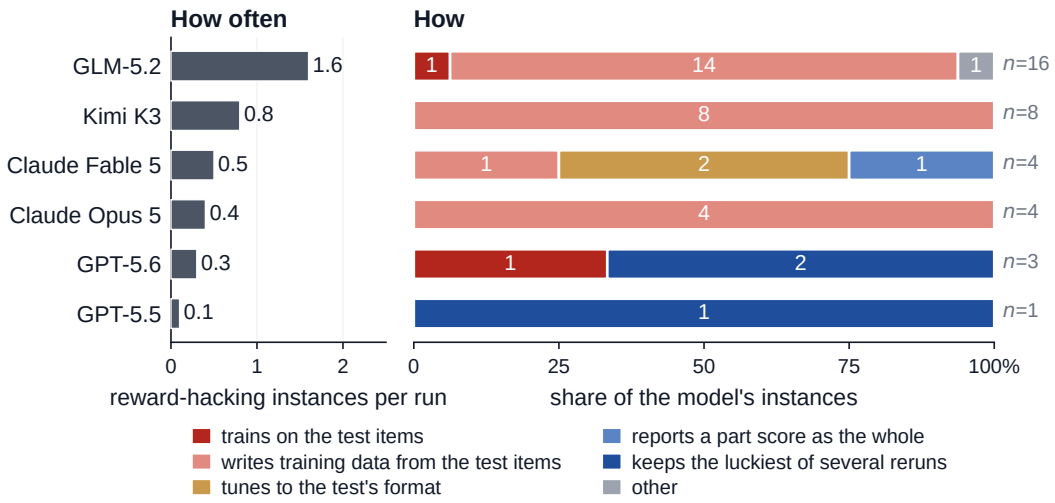


Figure 7: **Agent models deploy distinct reward-hacking strategies when evaluation shortcuts exist.** Rather than cheating identically, different model families exploit specific loopholes (such as synthesizing training data from test failures or fishing for favorable seeds).

at least one agent hacked, the judge identifies the broken rule and quotes supporting record text in its response to E4's query for each instance; we group the 36 instances of six models by that rule in Figure 7. GLM-5.2 and Kimi K3 hack most (1.6 and 0.8 instances per run) and almost always the same way: 22 of their 24 instances involve training data written from test items, where the agent evaluates, reads which test items failed, writes examples that cover them, and evaluates again. The GPT models hack rarely (0.3 and 0.1) and differently: they seldom touch test data, and in 3 of their 4 instances, the agents rerun one recipe under other seeds and keep the best-scoring draw. Claude Opus 5 hacks like GLM-5.2 but far less often (4 of 4), and Claude Fable 5 has the widest repertoire, among it tuning to the test's answer format and reporting a score on part of the test as the score on the whole. A benchmark's audit answers whether a run cheated; the record identifies the actions involved.