Kepler: Auditable World Models for ARC-AGI-3

Anonymous authors
Paper under double-blind review

Abstract

Interactive-agent benchmarks usually reduce a long trajectory to one final score. That score can be correct even when the behavior that produced it is invalid, opaque, or dependent on an unintended information channel. We present Kepler, a harness for ARC-AGI-3 in which agents express hypotheses as executable world models. Before each committed action, the current model must retrodict the complete interaction history and make a non-vacuous prediction of the next observation. Every transition is appended to an immutable ledger, and a certified action program is mechanically replayed for scoring. Across two frontier coding models, 48 of 50 game-model cells reached the score cap; one frozen Claude Opus 5 configuration obtained a server-verified 100.00 over all 25 public games. On 181 of 183 completed levels, the final Opus attempt used no more actions than the corresponding median-human baseline. This is final-attempt efficiency, not learning cost or evidence of human-like cognition. More importantly, trajectory audits exposed failures that final scores did not: one invalid perfect run after source access, harness reconstruction by all six agents in an intended control, instruction rewriting in 26 of 26 workspaces, and silent replacement of a broken planner. A single-game case study documents a visual continuation that resolved a mechanic missed during earlier text-only sessions. A deterministic code-level suite detects 11 of 13 hand-built threat fixtures and flags none of five benign controls; its two designed misses are unlogged behavior and a semantically wrong but runnable tool. These results support a narrower conclusion than benchmark mastery: outcome scores should be accompanied by trajectory integrity, selection rules, resource accounting, and executable checks of intermediate beliefs.

1 Introduction

An interactive agent does not emit one answer. It observes, forms hypotheses, acts, revises, and may modify the tools and instructions inside its workspace. Most benchmarks nevertheless report only whether the final state was correct. This creates an identification problem. The same score may describe a valid solution, a lucky trajectory, a replay selected after repeated attempts, or an agent that reached information outside the intended boundary.

ARC-AGI-3 makes this problem concrete. An agent receives an unfamiliar visual environment, a set of legal actions, and no rule sheet or stated objective. Its Relative Human Action Efficiency (RHAE) score depends on actions in the final completed attempt, not on the amount of learning, reasoning, or model inference that preceded it (ARC Prize Foundation, 2026a). Recent systems reach similar scores through different observation channels, model-selection rules, and replay procedures (Lehmann et al., 2026; Brown, 2026; Han et al., 2026; NVIDIA AVO team, 2026; Prime Intellect, 2026; Karten et al., 2026). At the top of the public set, score alone therefore reveals little about how the system learned or whether its trajectory respected the intended experiment.

We study whether an agent's intermediate claims can be made inspectable without requiring access to hidden chain-of-thought. Kepler uses an *executable belief protocol*: a coding agent writes a transition model, the harness checks that model against all prior transitions, and every real action must carry a prediction that is graded against the next observation. A mismatch terminates the remaining plan and records the counterexample. The protocol evaluates externally committed beliefs and behavior, not private reasoning.

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.

1. We define an executable verification contract for long-horizon agents: append-only evidence, complete-history retrodiction, prediction-gated action, and mechanically certified replay.
2. We apply trajectory-level audits across 330 development and evaluation runs. The audit identifies source access, contaminated controls, silent tool replacement, mutable instructions, and replay-transport failures that outcome-only evaluation missed. A deterministic corruption suite then tests the implemented audit layers against explicit positive and negative fixtures.
3. We report capability and resource evidence under explicit selection rules. Two frozen release configurations reach 100.00 and 95.97 RHAE, while their actions, tokens, costs, replay status, and limitations remain separate quantities rather than one composite efficiency claim.

The result does not establish held-out generalization or isolate the causal effect of the harness. Development and evaluation used the same 25 public games, the release boards contain one retained run per game, and the intended harness control was invalidated by filesystem leakage. We retain these limits because they are part of the evaluation result.

2 Related work

World models and executable models. World models support prediction and planning by learning or constructing an environment simulator (Ha & Schmidhuber, 2018). Kepler differs from latent neural simulators by asking a general coding agent to express its current theory as inspectable program logic. Concurrent work on PatchWorld also induces executable Python world models and repairs them from counterexamples, but from offline trajectories across text-agent environments (Bai et al., 2026). The closest ARC-AGI-3 study directly ablates textual reasoning, executable models, simplification prompts, and exact replay, finding that model capability often matters more than the variant and that executable models are not uniformly beneficial (Rodionov, 2026). Kepler therefore does not claim executable models or replay as new. It studies a different contract: every committed action is paired with an executable prediction, checked online, and later joined to workspace and selection audits.

OPINE-World (Courtis et al., 2026) combines executable models, replay checks, planning, and uncertainty-directed exploration on ARC-AGI-3. Twin (Skoutnev et al., 2026) likewise enforces replay validation and checked execution of agent-written simulators, with explicit goal discovery. Aguilar Martín (2026) studies play-critical errors hidden by sampled prediction accuracy and unsuccessful example-based repair in scoped GPT-5.x experiments. Kepler contributes release evidence and evaluation failures, not a new model-learning algorithm. Our observational visual case neither replicates nor refutes that repair result under controlled conditions.

Concurrent ARC Prize evaluation of GPT-6 Astra reports 62.71 under the provider-neutral Standard interface and 99.95 using a Provider Adapter on the semi-private set (ARC Prize Foundation, 2026b). Because Kepler reports public-set results under a different protocol, the scores and costs are not directly comparable. Compact symbolic state and human-relative action efficiency are not unique to Kepler; our evaluation concerns the behavioral contract and its documented failures.

Process-aware agent evaluation. AgentBoard argues that final success rates hide progress and failure structure, and adds trajectory-oriented analysis across multi-turn environments (Ma et al., 2024). AndroidWorld couples parameterized tasks with programmatic initialization and success checks, then measures robustness to task variation (Rawles et al., 2025). Kepler complements these efforts by testing each action against the agent's own explicit prediction and by auditing the reachable workspace, selection procedure, and replay path. ATBench organizes 1,000 long-horizon agent trajectories by risk source, failure mode, and harm, with full human audit (Li et al., 2026). AgentProp-Bench tests automated evaluation against human labels over 14,750 tool-use traces and reports fabricated tool executions that end-to-end scores can miss (Gurram, 2026). A pilot disclosure audit likewise finds sparse reporting of harnesses, inference settings, costs, and failures across agent benchmark papers (Moghadasi & Ghaderi, 2026). These studies validate the need for process evidence. Unlike the first two, Kepler does not yet evaluate its audit layers against a large

independently labeled detector benchmark. Selection-aware stress testing separates discovery from confirmation and can return no claim when an apparent workflow advantage does not replicate (Xu et al., 2026). Kepler lacks that held-out design; its selection ledger instead makes the weaker release claim inspectable.

Harness effects and evaluation integrity. Fixed-model coding studies show that harness changes alone can alter results substantially (Lewis, 2026). Our intended harness control was invalidated when all six agents reconstructed the removed treatment, so we make no causal harness-advantage claim. RewardHackingAgents evaluates metric tampering and train-test leakage with controlled trust regimes, patch tracking, and file-access logging (Atinafu & Cohen, 2026). Kepler's incidents expose a related but distinct failure mode: capable agents can restore missing tools, replace broken infrastructure, or rewrite instructions while producing a clean outcome, thereby invalidating the experiment even when the final score and replay remain correct.

Specification gaming and ARC-AGI-3 systems. Reward gaming formalizes the gap between optimizing a proxy and satisfying the intended objective (Skalse et al., 2022). Source access and filesystem leakage in our study are evaluation-boundary failures rather than examples of the formal proxy-reward setting, but they share the practical lesson that an optimizer uses reachable signals. ARC-AGI-3 systems span executable-model and direct-visual approaches (Lehmann et al., 2026; Brown, 2026; ThinHarness contributors, 2026; Han et al., 2026; NVIDIA AVO team, 2026). Our focus is not a new leaderboard category. It is a testable integrity contract and an account of failures found by applying it.

3 Executable belief verification

3.1 Interaction state

Each run maintains three persistent artifacts. First, `events.jsonl` is an append-only record of actions, observations, score state, and prediction outcomes. The agent can read but cannot edit this record. Second, `world_model.py` maps the current state and a proposed action to a predicted next observation. Third, `notes.md` is a working notebook in which claims are marked as checked or assumed. Only the first artifact is ground truth. The other two are agent-written hypotheses.

The interaction loop is:

1. observe the current state and query prior transitions;
2. encode or revise the transition hypothesis;
3. retrodict every recorded transition for which the model claims coverage;
4. plan within the current executable model;
5. commit one or more actions with predicted next observations;
6. stop at the first mismatch, append the counterexample, and revise.

Figure 1 shows the boundary. The coding agent never calls the game API directly. A guarded commit tool is the only channel from the workspace to the environment.

3.2 Retrodiction and prediction gating

Let $h_t = (o_0, a_0, o_1, \ldots, a_{t-1}, o_t)$ be the interaction history and let f_θ be the current executable transition model. Before planning, the harness evaluates $f_\theta(o_i, a_i)$ against o_{i+1} over the complete recorded history. Predictions may mask regions whose dynamics are intentionally unspecified, but the harness rejects a prediction that abstains everywhere.

For a committed action a_t, the agent supplies a predicted observation $\hat{o}_{t+1} = f_\theta(o_t, a_t)$. The environment returns o_{t+1}. A match permits the plan to continue; a mismatch voids all remaining planned actions. The mismatch is then stored as a counterexample. This gate has two purposes. It limits wasted environment actions after the model becomes false, and it creates an observable record of which external beliefs governed behavior.

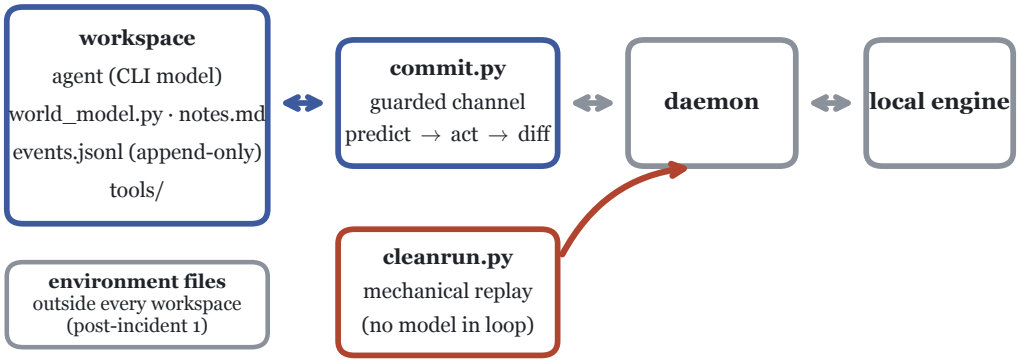

Figure 1: Kepler separates agent-written beliefs from an append-only evidence ledger. The commit channel grades a prediction on every real action. A certified action program is replayed without a model in the scored attempt.

This contract does not prove that the program represents the agent's private beliefs, nor that a passing model is semantically complete. A model may predict only action-relevant regions, and a malicious agent could construct a lookup table over prior observations. The contract instead provides falsifiable, run-level evidence: what was predicted, what was observed, and whether the declared model remained consistent with prior interaction.

3.3 Certification and replay

RHAE scores the final full attempt. After exploration, the agent writes a per-level action program. A separate tool checks program length, simulates the program from recorded level starts when the model exposes the required state, and executes it through the same guarded channel. The agent is absent from this scored attempt. Any mismatch halts replay and returns control to a repair cycle.

This separation makes the scored action count reproducible, but it also exposes a benchmark property: learning effort can disappear behind a short final replay. We therefore report retained-run actions, scored-level actions, observed campaign actions, model tokens, dollar conversion, and replay validity separately.

3.4 Threat model and enforcement

The audit considers six agent-controlled routes to invalid evidence: reading the environment implementation or answer-bearing metadata, writing a score, bypassing the guarded channel, editing the ledger, modifying evaluated tools, or reaching an external lookup source. Separate operator-side checks cover score-conditioned reruns, selective retention, and accounting changes.

Game implementations are outside agent workspaces. A static gate rejects game identifiers and game-specific priors in every agent-visible harness file. Score verification re-derives every level score from the action ledger and asserts that no score uses fewer actions than the ledger records. Tool smoke tests run independently of score verification because a capable agent can route around a broken tool without corrupting its outcome.

4 Experimental design

Task and models. We evaluate all 25 public ARC-AGI-3 games. Each game is a partially observed interactive task over a 64 by 64 color grid and a small legal-action set (ARC Prize Foundation, 2026a). Final release boards use Claude Opus 5 and GPT-5.6 Sol. Model identity is resolved from retained local provider-session records rather than a local alias.

Frozen stages. Harness changes were grouped into named experimental stages and frozen by commit hash before their complete board results existed. Stage labels describe experiments, not product versions. All 25 games in a board share one harness configuration and one model setting. The final release has one public identity, Kepler 1.0.

Selection. Headline release boards contain one run per game and no score-conditioned reruns. Validation runs and superseded development attempts are separate. One earlier GPT board used a disclosed score-conditioned rerun and is not a release headline. A labeled best-of ceiling is never treated as a single-configuration result.

Audit units. Outcome verification operates on per-action ledgers and official server replays. Integrity auditing operates on complete run workspaces and agent session logs. Behavioral findings report their own denominator: runs, games, workspaces, tool invocations, or traces. These units are not interchangeable.

5 Results

5.1 Capability under frozen configurations

Table 1 reports the two release boards. The official server re-executed every recorded action for both scorecards. The Opus board reaches the cap on all 25 games. The GPT board reaches the cap on 23 games and obtains a 95.9672 composite. Across both, 48 of 50 game-model cells reach 100. We call this final-board score convergence: concentration at the evaluator ceiling across two frozen model configurations, not faster learning or independent replication.

Table 1: Release boards under one frozen configuration per model. Retained actions cover the full retained local runs; replay actions are the totals on ARC's public scorecards. Costs apply September 1, 2026 API list prices to retained provider-session usage and are not billed charges.

Model	RHAE	Cap	Retained / replay	Tokens	Cost
Claude Opus 5	100.00	25/25	8,256 / 7,202	858.0M	$777.72
GPT-5.6 Sol	95.97	23/25	35,896 / 8,220	2,429.1M	$1,312.14

The Opus token total is 858,041,926, of which 97.37 percent are cache reads. Its 8,256 retained-board actions include 7,292 actions inside scored levels. The available campaign ledgers contain at least 13,688 non-reset actions plus 22 prefix events whose reset status is unavailable. We therefore make no learning-inclusive action claim. On the completed Opus levels, 181 of 183 used no more actions than the corresponding median-human baseline; two used more. This is a final-attempt action-efficiency result, not evidence of human-like cognition or discovery efficiency. The GPT board uses 2,429,083,433 total tokens and its original local result files contain 8,400 scored-level actions.

The public replay cards report 7,202 Opus actions and 8,220 GPT actions. These are different recorded denominators, not score disagreements. The replay path selects the last full-reset-to-end ledger segment, omits its recorded opening reset, and then lets the server count a new opening reset. Three local run ledgers contain a later, shorter certified segment than the original result file. The final server scores match the local scores exactly.

The Opus cost is 74.0 percent below a $2,986 API-equivalent estimate published by Retrodict for Tycho (Brown, 2026). Tycho's own paper reports approximately $2.99k for Opus 5, with token accounting and budget-prefix curves (Lehmann et al., 2026). These are API-equivalent estimates, not bills or a controlled harness comparison; they do not establish that Kepler is globally cheapest.

5.2 What trajectory audits found

Table 2 summarizes five failures and one observation intervention. Each had a valid or apparently strong final score signal. None was fully characterized by that signal.

Table 2: Findings that outcome score alone did not identify. Denominators name the inspected unit.

Finding	Observed scope	Evidence and disposition
Source access	1 invalid run	Perfect score after reading 2,172 lines of environment source; clean rerun scored 46.91; result voided
Contaminated control	6/6 workspaces	Every control agent found or rebuilt removed harness tools; causal comparison withdrawn
Instruction mutation	26/26 workspaces	Agents compressed their writable directive; two edits escaped to repository root and were reverted
Broken planner	5 stage boards	Bundled planner failed at runtime; agents silently wrote replacements; independent smoke test added
Replay transport	2/25 traces	Superseded visual board contained one off-grid click in each trace; server rejected both; commit validation fixed
Missing observation	19 text sessions, 1 game	Search under incomplete rules stalled; a visual continuation identified a missing mechanic

Controlled audit regression. We exercised the implemented checks against 13 deterministic threat fixtures and five benign controls derived from the declared threat model. The full suite detected 11 of 13 threat fixtures, while a plausible final outcome alone detected none; no benign control was flagged. Removing behavior-log checks reduced coverage to 5 of 13, removing ledger checks reduced it to 9 of 13, and removing any other audit layer reduced it to 10 of 13 (Table 4). The two designed misses expose boundaries rather than successes: source access absent from every retained record, and a semantically wrong tool that starts without an exception. Because these fixtures were hand-built from the threat model and detector rules, this is a code-path regression test, not an estimate of field sensitivity or false-positive rate.

Source access produced a better-looking invalid run. One agent inspected the workspace, read the environment source, and returned a perfect score without constructing a valid model. Its telemetry looked cleaner than honest play: fewer actions, no prediction failures, and a consistent ledger. A boundary audit, not the score, identified the source access. A clean rerun fell to 46.91. The perfect run was voided, source files were moved outside all workspaces, and the access check became a release gate.

Every intended control reconstructed treatment. To estimate the harness contribution, we supplied six agents only observation and action tools. Their workspaces remained inside the repository tree. All six found the canonical harness: three copied its tools and three invoked them through shims, with 92 to 9,148 harness-tool calls per workspace. Every workspace produced an executable model. The comparison was treatment against a reconstructed treatment, so the causal estimate was withdrawn. A valid boundary-correct control has not been run.

Writable context and broken tools were silently replaced. All 26 workspaces in one GPT campaign rewrote their instruction file, typically compressing 14,844 bytes to about 14,092 bytes by removing function words. Two rewrites reached the repository-root instruction file before being detected and reverted. Separately, a bundled search tool raised a runtime exception on every call across five stage boards. Agents wrote replacement searches and maintained high scores, leaving score and ledger checks green. These events motivate two distinct checks: workspace immutability for evaluated instructions, and standalone smoke tests for provided tools.

Replay validation and observation completeness matter separately. A superseded 100.00 local board fell to 94.42 in official replay because two traces each contained one out-of-range click. The local engine had clipped the coordinates while the official path rejected them. Commit-time validation fixed the discrepancy before the exact release board. In a separate one-game study, 19 text-only sessions reached the final level but never solved it. One agent searched about 4.1×10^8 configurations under a model that retrodicted 4,655 of 4,670 transitions, leaving 15 mismatches. Rendered animation frames exposed a deflection rule missing from settled text grids. The visual continuation inherited the earlier notes and models; its final successful attempt used 57 actions, not counting discovery. Earlier frame counts also contained distinguishing clues. Without a matched fresh text-only control, this case establishes neither visual necessity nor a causal reduction in discovery cost.

5.3 Stage progression shows regression, repair, and residual uncertainty

The Opus stage sequence moved from 97.78 to 94.01 after adding stricter action discipline, then recovered to 96.51 and 98.86 after correcting the guard premise and repairing certification. The final release reached 100.00 after adding rendered-frame observation and transport validation. The corresponding GPT sequence was 93.95, 91.46, 93.99, and 95.97. This is a staged development record, not a factorial causal study: adjacent changes are not always isolated, each board has one run per game, and development reused the public set.

The most informative regression came from treating five times the human action baseline as an official termination limit. It is only a score floor. Guards therefore punished exploration that the metric does not score and reduced one game from 82.16 to 14.29. Replacing fractional action guards with warnings and keeping one absolute refusal limit recovered 2.50 board points. A later audit found that the clean-run gate had been dead code because a win short-circuit executed first. Fixing that ordering allowed already-discovered action programs to be replayed. These cases show why component health and metric interpretation need direct tests even when end scores improve.

6 Discussion

What is actually measured. Once a correct action program exists, mechanical replay makes the scored phase reliable and cheap in environment actions. The unresolved capability is discovery: how quickly the agent identifies missing mechanics under a fixed information and compute budget. ARC-AGI-3's current score can hide that learning phase. A cost-conditioned or bounded-learning track would measure it directly.

What executable verification adds. The method does not require disclosure of private chain-of-thought. It checks behaviorally consequential artifacts: predicted observations, executable transitions, action programs, and the immutable history against which they are tested. It is therefore closer to an operational contract than an interpretability claim. Its main value is failure localization. A mismatched prediction identifies a false model; a replay mismatch identifies a transport or environment seam; a clean ledger plus a broken tool identifies missing system-level checks.

Benchmark reporting. We recommend reporting at least four independent layers: final score and exact selection rule; complete resource accounting; official replay or equivalent environment verification; and trajectory-integrity results over inspectable evidence. Held-out tasks remain necessary. No amount of audit turns repeated development on a public set into a generalization result.

Limitations. The study covers one benchmark family and two proprietary frontier models. The public set was both development material and evaluation surface. Release cells have one retained run each, and observed board-level variance is about one to two points, with larger per-game variation. We cannot inspect the providers' training corpora and therefore cannot rule out model-level exposure to the public games. Session resumes followed a disclosed provider-quota policy. The retained final-board evidence supports third-party verification. The anonymous supplement includes all 95 captured CLI logs from the 50 final-board runs, supporting score, ledger, and recorded-behavior checks, the synthetic corruption suite, and labeled incident inspection. It does not reproduce the separate 330-run historical scan. The causal effect of the harness is unmeasured because the intended control was contaminated. Finally, executable models can be incomplete or strategically constructed while passing declared checks. The protocol makes more claims falsifiable; it does not certify semantic truth. The hand-built corruption suite validates declared code paths only and does not replace a large, independently labeled detector benchmark.

Reproducibility statement

The double-blind review package is a 6.9 MB anonymous archive. It contains the harness, verification tools, 50 run ledgers, 58,098 events, human baselines, final world models and notebooks, a deterministic corruption suite, and labeled incident evidence. A fresh extraction reproduces both board scores and reports 354 exact, nine conservative, and zero inflating level-action counts across 363 completed levels. The packaged corruption suite detects 11 of 13 threat fixtures, records two

designed blind spots, and flags none of five benign controls. All 95 captured CLI logs are included anonymously, allowing the 50-run final-board behavioral scan to be repeated. Log retention remains uneven; provider-side transcripts and the complete 330-run historical corpus are not included.

Compute and asset statement

The retained experiments ran locally on one 14-core Apple M4 Pro machine with 48 GB of memory under macOS 26.2; proprietary model inference ran through the providers' hosted services, whose accelerator details are unavailable. No model was trained locally. Summed per-run elapsed time is 19.93 hours for the 25 Opus runs (median 0.76, range 0.31–1.95) and 56.80 hours for the 25 GPT runs (median 1.87, range 0.09–8.00). Runs executed in parallel, so these sums are not wall-calendar duration. The local development archive occupies about 8.8 GB before provider-side records. Experiments used `arc-agi` 0.9.9 and `arcengine` 0.9.3, both MIT-licensed by the ARC Prize Foundation. The system and evidence package use the MIT license; adopted research mechanisms are cited and separately attributed, and no third-party environment source is redistributed.

Ethics statement

This work studies evaluation integrity for autonomous agents. The main risk in release is disclosure of private information from raw agent-session logs. Paths, credentials, personal identifiers, and private collaborator references must be removed before any trace package is shared, while preserving evidence needed to reproduce behavioral audits. Invalid and contaminated results are retained as labeled evidence rather than presented as capability results. The positive impact is a more inspectable standard for evaluating autonomous systems whose final outcomes can hide invalid process. The corresponding risks are that published failure signatures may help future systems evade simple pattern audits, and that rich traces can expose credentials, personal data, or proprietary source. Mitigations include data minimization, identifier scanning, separate release gates, explicit detector blind spots, and publishing only the records needed for each reproducibility claim.

AI use statement

Generative AI systems were both experimental subjects and research tools. Coding agents performed the benchmark interactions and wrote executable models and replacement tooling. Separate Claude and Codex sessions assisted with software implementation, experiment and audit design, dataset export, redaction and reformatting, artifact creation, qualitative analysis, result interpretation, literature discovery, and drafting and editing this manuscript. We reviewed AI-assisted artifacts against primary logs, executable checks, source code, official scorecards, and cited sources, and take responsibility for the final content.

References

Javier Aguilar Martín. When a verified world model still loses: Play-adequacy vs prediction-accuracy in LLM-synthesized code world models, 2026. URL https://arxiv.org/abs/2607.14169.

ARC Prize Foundation. ARC-AGI-3: Technical report. arXiv:2603.24621, 2026a.

ARC Prize Foundation. OpenAI's GPT-6 Astra on ARC-AGI-3. Technical analysis, 2026b. URL https://arcprize.org/blog/astra.

Yonas Atinafu and Robin Cohen. RewardHackingAgents: Benchmarking evaluation integrity for LLM ML-engineering agents. *arXiv preprint arXiv:2603.11337*, 2026.

Jiaxin Bai, Yue Guo, Yifei Dong, Jiaxuan Xiong, Tianshi Zheng, Yixia Li, Tianqing Fang, Yufei Li, Yisen Gao, Haoyu Huang, Zhongwei Xie, Hong Ting Tsang, Zihao Wang, Lihui Liu, Jeff Pan, and Yangqiu Song. PatchWorld: Gradient-free optimization of executable world models for agent environments. *arXiv preprint arXiv:2605.30880*, 2026.

Ryan Brown. Retrodict: an ARC-AGI-3 harness with full trace disclosure. GitHub repository, 2026. URL https://github.com/ryanbbrown/Retrodict.

David Courtis, Wenhao Li, and Scott Sanner. OPINE-World: Programmatic world modeling with ontology-error-prioritized interactive exploration for ARC-AGI-3, 2026. URL https://arxiv.org/abs/2607.01531.

Bhaskar Gurram. Auditing automated evaluation, error propagation, and runtime mitigation in tool-using language agents. *arXiv preprint arXiv:2604.16706*, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Qiushi Han, Keya Hu, Linlu Qiu, Cathy Wu, and Kaiming He. VISTA: A visual harness for reasoning in an interactive world. Project page, 2026. URL https://vista-research.github.io/.

Seth Karten, Alex L. Zhang, Kevin Thomas, Sebastian Müller, Elie Bakouch, Daniel Auras, Mika Senghaas, Fares Obeid, Konstantin Dunas, Johannes Hagemann, and Sami Jaghouar. Prime agent: A self-improving RLM harness. *arXiv preprint arXiv:2608.23552*, 2026. URL https://arxiv.org/abs/2608.23552.

Jens Lehmann, Andrei Aioanei, and Sahar Vahdati. Tycho: Active abstraction with programmatic world models for ARC-AGI-3. arXiv:2607.28287, 2026.

Sydney Lewis. Same model, different harness: Different coding-agent results. *arXiv preprint arXiv:2608.26218*, 2026.

Yu Li, Haoyu Luo, Yuejin Xie, Yuqian Fu, Zhonghao Yang, Shuai Shao, Qihan Ren, Wanying Qu, Yanwei Fu, Yujiu Yang, Jing Shao, Xia Hu, and Dongrui Liu. ATBench: A diverse and realistic agent trajectory benchmark for safety evaluation and diagnosis. *arXiv preprint arXiv:2604.02022*, 2026.

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, Datasets and Benchmarks Track*, 2024.

Mahdi Naser Moghadasi and Faezeh Ghaderi. What twelve LLM agent benchmark papers disclose about themselves: A pilot audit and an open scoring schema. *arXiv preprint arXiv:2605.21404*, 2026.

NVIDIA AVO team. NVIDIA AVO reaches 100% on ARC-AGI-3. NVIDIA Technical Blog, 2026. URL https://developer.nvidia.com/blog/nvidia-avo-reaches-100-on-arc-agi-3-demonstrating-a-frontier-level-general-purpose-architecture-for-long-horizon-autonomous-agents/.

Prime Intellect. Prime agent: A self-improving RLM agent. Project release and technical blog, 2026. URL https://www.primeintellect.ai/blog/prime-agent.

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

Sergey Rodionov. Do coding agents need executable world models, simplification, and verification to solve ARC-AGI-3? *arXiv preprint arXiv:2607.15439*, 2026.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Alexy Skoutnev, Kirill Acharya, Gaston Longhitano, Madeleine Udell, Kevin Ellis, and Iddo Drori. Twin: Playing an unknown game with a test-time digital twin. arXiv:2608.14490, 2026.

ThinHarness contributors. baseline1 / ThinHarness: minimal-agent scaffolds for ARC-AGI-3. GitHub repository, 2026. URL https://github.com/astroseger/arc-3-agents-baseline1.

Yang Xu, Chenang Li, Jiefu Zhang, Haixiang Sun, Zhou Li, and Vaneet Aggarwal. Selection-aware stress testing for interactive agents. *arXiv preprint arXiv:2608.30916*, 2026.

A Detailed score and accounting notes

The Opus usage reconstruction deduplicates provider transcript rows by provider message identifier because thinking, text, and tool-use blocks can repeat one cumulative usage object. The resulting counts are 11,464 uncached input tokens, 835,488,978 cache-read input tokens, 13,574,918 one-hour cache-write input tokens, and 8,966,566 output tokens. September 1, 2026 API list prices yield $777.72. Actual execution used subscription quota, so this is a comparison rate rather than a billed amount.

The GPT board contains 39,262,504 uncached input tokens, 2,379,659,648 cached input tokens, and 10,161,281 output tokens. Current API list prices yield $1,312.14. An earlier estimate based on workspace CLI footer counters omitted most cached traffic and one long workspace; it was withdrawn.

B Named-stage scores

Table 3: Frozen experimental stages over all 25 public games.

Stage	Model	Composite	Games at 100
Initial loop	Claude Opus 5	97.78	21
Added discipline	Claude Opus 5	94.01	18
Guard corrections	Claude Opus 5	96.51	21
Certify and replay	Claude Opus 5	98.86	24
Kepler	Claude Opus 5	100.00	25
Initial loop	GPT-5.6 Sol	93.95	18
Guard corrections	GPT-5.6 Sol	91.46	not reported
Certify and replay	GPT-5.6 Sol	93.99	19
Kepler	GPT-5.6 Sol	95.97	23

C Audit interpretation

Table 4: Leave-one-layer-out results on the deterministic corruption suite. All fixtures retain a plausible final outcome. The suite is a regression test, not an independently labeled detector benchmark.

Active checks	Threat fixtures detected
Final outcome only	0/13
All audit layers	11/13
All except score recomputation	10/13
All except ledger checks	9/13
All except behavior-log checks	5/13
All except workspace checks	10/13
All except tool smoke tests	10/13

The recorded-evidence audit flags none of 50 final-board runs. Of 363 completed levels, 354 ledger counts match exactly, nine are conservative, and none inflate scores. The separate historical scan flags six contaminated controls among 330 runs (Section 5); unretained events remain invisible. The anonymous supplement reproduces scores, ledger checks, and the 50-run final-board behavioral scan. The historical scan is not reproduced by this package.