Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

The World Is a Hypothesis: Epistemic Control for Executable World-Model Agents in Novel Interactive Environments

Bruno Laureano dos Santos

blaureanosantos@gmail.com

independent researcher

Research Article

Keywords: executable world models, coding agents, ARC-AGI-3, hypothesis revision, epistemic control, interactive environments, large language model agents

Posted Date: July 28th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10462949/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

The World Is a Hypothesis: Epistemic Control for Executable World-Model Agents in Novel Interactive Environments

Bruno Laureano dos Santos *blaureanosantos@gmail.com*

Abstract

Executable world-model agents turn interaction histories into executable models that can be tested and searched, but predictive agreement does not establish causal correctness or planning adequacy. We introduce THEA (Typed Hypothesis-driven Epistemic Agents), a prompt-defined revision policy that treats the executable world model as a falsifiable task theory. A parent agent preserves causal context and retains exclusive authority over environment actions. Observable failures and declared checkpoints activate bounded specialists (subagents) for framing, generalization, raw-evidence review, rival-hypothesis testing, causal reinduction, and transition observation. Typed artifacts record competing hypotheses, discriminating probes, and verdicts, while the executable theory, logs, and interaction archive preserve the evidence behind each revision.

We evaluate THEA, with a Claude Code Opus 4.8 parent at `xhigh` effort and provider-native Sonnet 5 specialists, on the 25 public ARC-AGI-3 games through a historical portfolio recording each game's final trajectory from across the harness's development. It fully solves 22/25 games, averages 92.63 per-game Relative Human Action Efficiency (RHAE), and has an API-equivalent cost of $5,007.85 versus $6,222.18 for the strongest published agent on this benchmark, the version 1.5 Executable World Models release (hereafter v1.5), a 19.52% reduction concentrated in games that v1.5 leaves incomplete. Archived cases show exact replay preserving a wrong causal proxy, exhaustive search closing under an incomplete model, a faithful engine feeding an invalid planner, and a correct theory arriving after an irreversible decision. THEA makes hypothesis revision explicit, falsifiable, and auditable while exposing where that revision succeeds, fails, and incurs cost, offering a control layer for complex, orchestrated agent systems. The harness, evidence archive, and per-game run records are available at `https://github.com/bruno353/thea`.

1 Introduction

An executable world model can reproduce every observed frame and still encode the wrong causal mechanism. A search procedure can exhaust every state reachable under its model and incorrectly conclude that no solution exists. A faithful dynamics model can feed a planner whose coordinate convention makes its output invalid. Even a correct theory can arrive after the agent has spent an irreversible action. Predictive agreement is valuable evidence, but it does not by itself establish causal correctness, planning adequacy, or timely revision.

ARC-AGI-3 (ARC Prize Foundation, 2026a) makes this problem unusually visible. An agent enters an abstract interactive environment without natural-language instructions, a stated goal, or a supplied transition model. It must discover what objects and actions mean, infer what constitutes progress, carry that knowledge across increasingly difficult levels, and plan before its action budget is exhausted. The benchmark's Relative Human Action Efficiency (RHAE) metric rewards both completion and action efficiency: later levels receive greater weight, incomplete coverage caps the game score, and completed levels are scored relative to human action counts. The resulting task is to acquire and use a workable theory with limited interaction.

Executable world-model agents address this setting by turning the current theory into a program. State reconstruction, transition dynamics, rendering, terminal conditions, and planning can then be executed, tested, and revised instead of being held only in an opaque conversational trace. Our agent's default mode of operation, which we call the fast path (Section 3.2), is exactly such a direct loop: observe, reconstruct, revise, verify, plan, act, and compare prediction with outcome. THEA grew out of a multi-month sequence of our own architectures, absorbing components from several public systems as evidence favored them (Section 3.8); the contribution of this paper is the control layer around that loop.

The unresolved question is what should happen when this path stops producing progress. A failed plan may indicate an incorrect goal, a missing transition, a perception error, an implementation defect, an intractable planner, or a correct model searched under the wrong abstraction. Repeating the same analysis can harden a coincidental proxy into accepted fact. Restarting the theory or convening a standing committee after every mismatch can instead destroy useful causal context and spend more computation than the game requires. The design problem is therefore selective revision: how can an agent make revision explicit and falsifiable while preserving a direct operational path when the current theory is adequate?

We introduce THEA, Typed Hypothesis-driven Epistemic Agents, as a control method for this problem. THEA treats the executable world model as a falsifiable task theory rather than a settled representation. A parent agent retains the full causal context and exclusive authority over environment actions. Bounded specialists are activated by observed failure modes or checkpoints and return evidence, rival hypotheses, probes, or verdicts; they never act on the environment themselves. Decision-critical hypotheses and dispositions are externalized in typed artifacts, while the executable theory, logs, and interaction records preserve the evidence against which they can be challenged. This composition can be viewed as a dynamic graph of epistemic control implemented by prompts, workspace contracts, guarded transitions, and verdict joins. Section 3 defines that mapping precisely.

We study THEA through an archive covering all 25 public ARC-AGI-3 games: a historical portfolio of each game's final development-line trajectory, which solves 22/25 games at a mean 92.63 RHAE and is 19.52% lower in aggregate API-equivalent cost than v1.5. These are complete-system comparisons, and the portfolio spans successive harness versions, not one frozen agent.

The aggregate hides substantial heterogeneity. Savings are concentrated in games that v1.5 leaves incomplete, while several tractable games cost more without improving coverage or score. We analyze end-to-end outcomes, negative results, and archived revision mechanisms together and do not treat additional deliberation as uniformly beneficial.

This paper makes three contributions. First, it presents a prompt-defined method for falsifiable revision of executable task theories through typed artifacts, bounded specialists, and centralized action authority. Second, it develops an empirical taxonomy of failures across modeling, goal inference, observation, planning, implementation, and timing, supported by archived prediction-observation-revision chains. Third, it reports coverage, RHAE, actions, model cost, negative results, and evidence integrity across the development-line portfolio.

THEA organizes revision as an inspectable scientific process: competing explanations are stated, evidence is preserved, discriminating tests are declared, and control returns to action after a verdict.

2 Background and Related Work

2.1 ARC-AGI-3 and Action Efficiency

ARC-AGI-3 evaluates adaptation in abstract, turn-based environments whose instructions, dynamics, and goals are hidden from the test taker. Its public set contains 25 games organized into sequential levels. An agent observes a 64 by 64 grid, selects from the actions exposed by the current game, and must infer both what changes and what constitutes success. The official RHAE score combines level coverage with action efficiency relative to human baselines, weights later levels more heavily, and caps the game score by completed coverage (ARC Prize Foundation, 2026a). This makes inefficient exploration and late mastery visible even when a game is eventually completed.

The public set supports reproducible mechanism studies because environments, human baselines, and replayable interactions are available. It does not by itself establish private generalization. We therefore use it to study how an agent revises a task theory during a trajectory and keep public-set adaptation separate from claims about unseen benchmark environments.

2.2 Executable World Models and Play Adequacy

Programmatic world models use code as an explicit transition theory that can be executed and searched. WorldCoder (Tang et al., 2024) learns world models by writing code from interaction, while GIF-MCTS (Dainese et al., 2024) uses LLM-generated code world models inside model-based search. These systems establish code as a practical representation for test-time model induction rather than a final policy or answer format.

Executable World Models for ARC-AGI-3 (Rodionov, 2026a) adapts this paradigm to long interactive games. Its coding agent maintains a Python transition model, reconstructs state, renders predicted observations, verifies recorded trajectories, and plans through the model before acting. Our ARC implementation incorporates this system's released MIT-licensed code for the executable substrate. The substrate entered a line that had already absorbed several sources: a monolithic REPL agent and an earlier code-executing RLM prototype built on DSPy (Khattab et al., 2023) established persistent single-context autonomy in our architectures, TELL (Dots Post-train Team, 2026) demonstrated durable per-game mechanical memory, and RGB (Fox, 2026) contributed full textual state with explicit diffs (Section 3.8). We learned from each of these systems and absorbed the mechanisms that proved strongest in our experiments.

Execution and predictive testing still leave an adequacy gap. Recent work on verified but play-inadequate executable world models (Martín, 2026) shows that high sampled transition accuracy can omit rare but pivotal dynamics and therefore fail systematically under planning. Our archived cases expose a complementary problem inside long trajectories: exact agreement with every recorded observation can preserve a coincidental causal proxy, and exhaustive search can close relative to an incomplete transition theory. That work asks whether a synthesized model is adequate for play; we study how an acting agent can make the reopening and revision of such a model explicit and auditable.

2.3 Test-Time Learning and ARC Agents

Several ARC-AGI-3 systems adapt external state around a fixed base model. TELL (Dots Post-train Team, 2026) uses one persistent LLM conversation, programmatic tools, and a per-game manual that accumulates goals, mechanics, and solved-level knowledge. It demonstrates that a single agent can perform hypothesis-driven test-time learning without weight updates, and its abstract per-game manual motivated our doctrine of carrying a control's functional role across levels while re-locating the control by its effect. DreamTeam (Sarafian et al., 2026) frames the editable workspace as the trainable substrate and assigns observation, modeling, planning, probing, critique, and routing to specialized roles. It is the closest architectural neighbor to THEA's specialist layer and makes clear that multi-agent roles or workspace optimization alone are not a novelty claim.

Other systems structure exploration rather than revision authority. AERA (Han, 2026) separates exploration, verification, and planning while using the public suite to study a speed-depth trade-off. Graph-Based Exploration (Rudakov et al., 2025) maintains a graph of observed environment states and transitions to prioritize untested state-action pairs. Its graph represents the environment; THEA's graph represents control over epistemic work. The Read-Grep-Bash Agent (Fox, 2026) contributes a lightweight engineering pattern based on a full ASCII state, textual diffs, and batched plans, but its published evaluation covers only the three earlier preview games. We treat it as an interface inspiration rather than a 25-game comparator.

THEA combines a single context with bounded specialist re-examination. Unlike a standing committee, specialists do not share action authority or participate in every turn. Unlike a phase-only pipeline, their activation depends on the current failure locus: framing, causal contradiction, repeated planning failure, implementation inconsistency, or rule accretion. The returned object is incorporated into the same executable theory that the parent continues to test and use.

2.4 Adaptive Deliberation and Routing

The question of when to spend additional inference has been studied through dynamic planning and model routing. Learning When to Plan (Paglieri et al., 2025) shows that always planning can be both costly and harmful in long-horizon tasks, and trains agents to allocate planning effort dynamically. Query-routing methods similarly choose between a fast and a deliberative path using difficulty or uncertainty estimates. For example, ODAR (Ma et al., 2026) routes benchmark queries between heuristic and deliberative agents under a cost-accuracy objective.

THEA shares the premise that uniform deliberation is undesirable but operates at a different granularity. It does not route an independent query or choose a larger model once. It redirects epistemic work within an ongoing trajectory, after an observable claim about the task theory has failed or reached a declared checkpoint. The destination is also role-specific: a planner that repeatedly returns no route calls for a different investigation than a model that predicts correctly while making no goal progress. The present policy is prompt-programmed and training-free; learning a better escalation policy is future work.

2.5 Graph Workflows and Typed Deliberation

Representing multi-agent systems as graphs is established practice, and nodes, edges, loops, and specialist handoffs are not contributions of this paper.

THEA uses graph semantics to describe a narrower object: control over the revision of an executable theory. Nodes correspond to the parent fast path and bounded epistemic roles; edges are guarded by observed anomalies or declared checkpoints; graph state is the theory, evidence, logs, and typed hypotheses; and joins return a verdict without transferring environment-action authority. The implementation uses prompts and workspace contracts rather than a graph scheduler.

Typed collective reasoning is also adjacent. We replace unstructured multi-agent debate with typed epistemic acts and preserved disagreements: a hypothesis fork and its disposition are inspectable, the acts are centered on an executable environment oracle, and control stays in one acting parent. The specialists form a temporary epistemic branch rather than a standing voting body.

2.6 Positioning THEA

THEA's contribution is the selective, falsifiable, and auditable revision of an executable task theory during a long trajectory. It brings together established elements, executable world models, persistent typed claims, specialist roles with adversarial critique, graph-structured control, adaptive deliberation, and hypothesis testing, around that single operational problem. A direct model-plan-act path retains control until a specific failure or checkpoint justifies specialist review. Rival hypotheses name the observation that would distinguish them, evidence remains archived, and action authority stays with the parent. These contracts are not tied to an ARC-specific planner, but the evaluation in this paper is.

3 Method

3.1 THEA and Executable Task Theories

THEA, short for Typed Hypothesis-driven Epistemic Agents, is a control method for agents that construct and revise an executable theory of an initially unknown task. We use *executable task theory* for a collection of programs and records that jointly specify what state the environment is in, how actions change that state, how states produce observations, what constitutes a terminal outcome, and how action sequences are selected. An executable world model is one instance of such a theory.

THEA's ARC-AGI-3 fast path runs on a persistent workspace with fixed executable interfaces for dynamics, reconstruction, rendering, and planning; the agent fills and revises those interfaces from interaction. This substrate follows the executable world-model design demonstrated on ARC-AGI-3 by Rodionov (2026a) and

entered our line as its final absorption step (Section 3.8). THEA adds explicit revision state, specialist review roles, and a policy for leaving and re-entering the direct model-plan-act path.

At interaction step t, we denote the current executable task theory by T_t. In the ARC instantiation, its required components are listed in Table 1.

The observation oracle consists of the settled frame returned after a real action together with its terminal status. Intermediate animation frames are retained as evidence about how a transition occurred, but the principal prediction target is the final settled frame. Environment actions consume the benchmark action budget, whereas editing, executing, and searching the local theory do not. The method spends unscored computation to reduce poorly justified real interaction.

Section 3.4 defines the prompt-programmed control graph used to leave and re-enter this executable fast path; Figure 1 summarizes it. The graph is an orchestration abstraction; it does not represent every inference the agent makes.

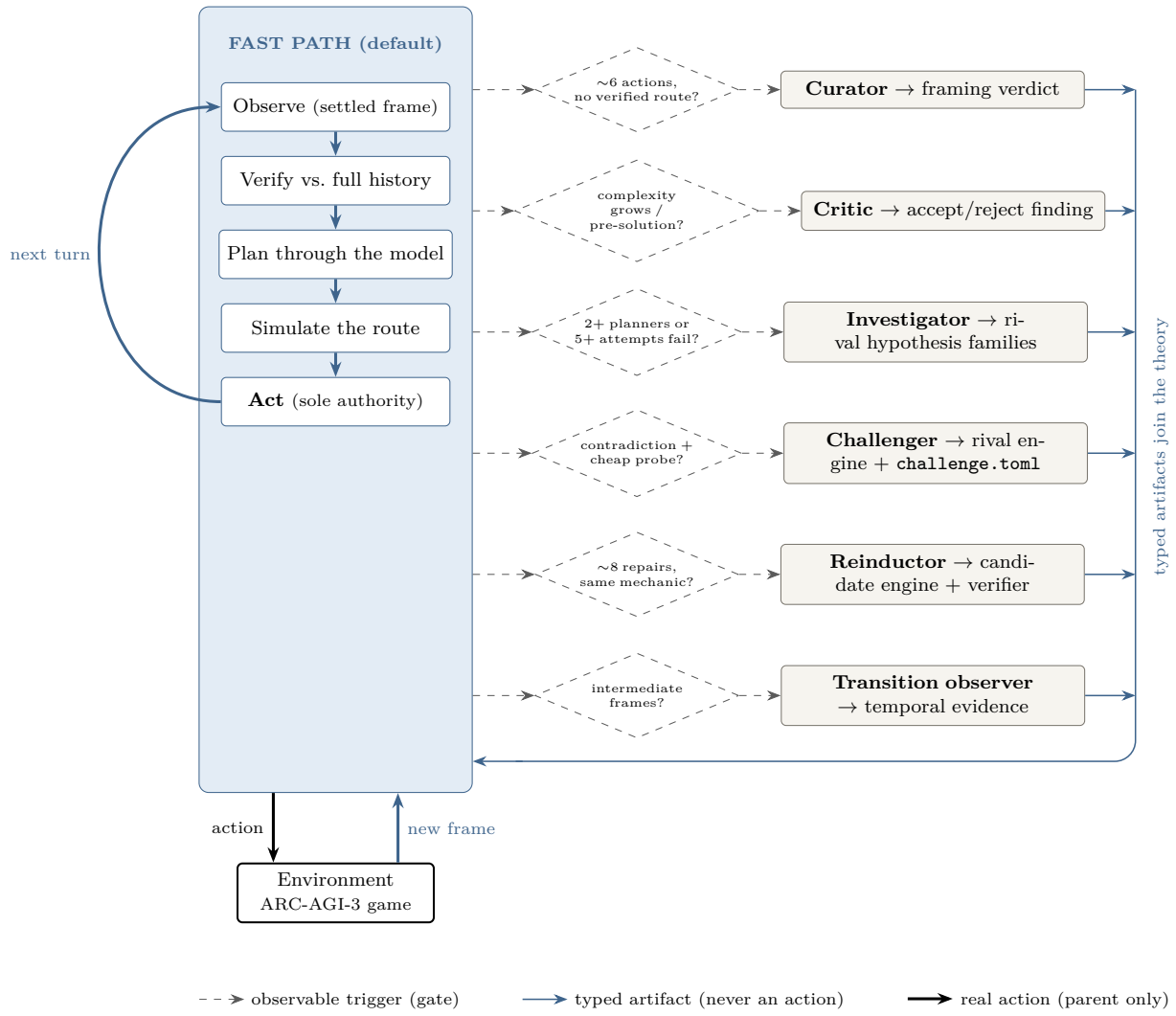


Figure 1: THEA's control structure. The fast path (left) runs by default on every turn: observe the settled frame, verify the executable theory against the full recorded history, plan and simulate through it, and act. Each specialist (right) sits behind an observable trigger (dashed gates); when invoked, it returns a typed artifact that joins the executable theory. Real environment actions are issued only by the parent.

Component	Runtime artifact	Contract
Transition and terminal model	`world_model_engine.py`	Maps an internal state and action to a successor state and one of `RUNNING`, `LEVEL_COMPLETED`, or `GAME_OVER`.
State reconstruction	`initial_state_reconstruction` in `world_model_state_io.py`	Reconstructs an internal state from the initial observation and, when needed, the recorded attempt history.
Observation model	`state_renderer` in `world_model_state_io.py`	Renders an internal state as the predicted settled ASCII frame.
Planner	`planner` in `world_model_main_planner.py`	Returns a sequence of model actions or `None` from a supplied internal state.
Declared theory and evidence trail	`world_model.md` and `level_N_reasoning_log.md`	Records ontology, mechanics, target hypotheses, evidence, mismatches, rejected explanations, and planner choices.

Table 1: Required components of the executable task theory in the ARC-AGI-3 instantiation.

3.2 The Executable Fast Path

The default path is a single-context loop: the parent maintains the causal context, edits the theory, plans through it, and controls all scored actions. Specialist review is not a committee that must approve every step. While predictions and plans remain adequate, the parent follows this sequence:

1. **Observe.** Read the latest settled ASCII frame and terminal status, and inspect the corresponding PNG or animation when visual type or transition semantics are ambiguous.
2. **Reconstruct.** Map the observation and attempt history into the current internal state.
3. **Revise.** Update dynamics, reconstruction, rendering, goal predicates, or the planner, while recording the hypothesis and supporting evidence.
4. **Replay-verify.** Run `verify_theory.py` against every recorded attempt through the current level. The verifier checks the reconstructed initial frame, predicted terminal status, and rendered settled frame at each step.
5. **Plan and simulate.** Ask the main planner for a route to `LEVEL_COMPLETED`, or use a saved auxiliary planner for a bounded information target. Simulate the route entirely inside the executable theory.
6. **Execute with prediction.** Submit a planned action through `plan_executor.py`, which predicts and executes the same action, compares the complete settled frames, and stops on a mismatch or terminal state.
7. **Repair or continue.** If prediction agrees with observation, continue the verified plan. Otherwise, stop real interaction, inspect the discrepancy, revise the theory, and replay-verify before acting again.

The default verification contract is intentionally strict. Agreement is checked over the complete settled frame, including regions outside the controlled object, and an unexplained mismatch blocks further execution. Evidence-gated exceptions for confirmed cosmetic displays and proven under-determined autonomous hazards are described later; neither exception permits silently ignoring an unknown causal region.

THEA distinguishes two contracts that are easy to conflate. World-model verification asks whether T_t reproduces the recorded interaction history. Planner verification asks whether the planner, starting from a reconstructed initial state of a completed level, returns a sequence that the engine predicts will reach `LEVEL_COMPLETED`. The first is implemented by `verify_theory.py`; the second by `verify_planner.py`. The planner verifier only evaluates levels already completed in the real environment, so a `None` result on the current level remains a diagnostic event rather than a proof that its goal is unreachable.

Passing either contract does not promote the whole theory to truth. A model can fit every recorded transition through a coincidental proxy, omit a variable that never varied in the trace, or encode a correct transition law behind an invalid planner state key. Likewise, a simulated plan can appear successful if the encoded completion predicate is wrong. Verification therefore grants authority only relative to the evidence and contract exercised. This limited meaning of verification is what makes explicit revision necessary even after a model has passed replay.

The prompt requires prediction before every in-attempt action and directs the parent to use the plan executor as soon as the model exists. The game client is still technically reachable from the coding workspace, so this discipline is not enforced by a capability boundary. We treat that residual gap as an implementation limitation and do not describe the harness as formally action-safe.

3.3 Externalized Epistemic State

THEA makes revision inspectable by separating four kinds of state that are often collapsed into a language model's conversational context:

1. **Raw evidence:** actions, settled ASCII frames, PNGs, animations, and real terminal statuses stored in the session archive.
2. **Executable hypothesis:** the current transition, reconstruction, rendering, and planning programs.
3. **Interpretive memory:** the ontology and target in `world_model.md`, plus per-level reasoning logs that record tested hypotheses, mismatches, and rejected explanations.
4. **Decision contracts:** structured artifacts that make a disputed claim, its alternatives, and the observation that will decide among them explicit before another costly action is taken.

We use *typed epistemic artifact* in the narrow fourth sense. It does not mean that every belief in the system is represented by a global schema. The runtime combines executable Python, durable Markdown, and TOML contracts for decision-critical forks. This is a design choice: typing the entire control flow was restrictive in earlier prototypes, while typing the claim that must be resolved exposes evidence without dictating how the parent reasons.

The principal contract is a causal challenge. When the standing theory H_{current} and a rival H_{rival} differ on one causal axis, the challenger writes `hypotheses/level_N/challenge.toml` with the following schema:

```
[hypothesis_current]
claim = "..."

[hypothesis_rival]
claim = "..."
differs_in = "actor | control_effect | win_relation | cosmetic_vs_causal | goal_predicate | hidden_state"

[probe]
action = "..."
predicate = "..."
prediction_under_current = "..."
prediction_under_rival = "..."
discriminating = true
```

The rival inherits every undisputed part of the standing theory. Before the probe can be executed, its declared predicate must take different predicted values under the two hypotheses. The parent then performs the single probe and compares the real outcome against that predicate; no minimal rival engine has to reproduce irrelevant parts of the full frame. If both hypotheses match or both miss, the probe has not discriminated the fork and the challenge remains open. The parent must refine the fork instead of returning to route variation under an unresolved causal model.

Goal uncertainty uses a related artifact. When no completion predicate has been observed or confirmed and the corresponding guard fires, the workflow prescribes `hypotheses/win_hypotheses.toml`: two or three

candidate completion predicates, evidence for and against each candidate, and the cheapest discriminating probe. Candidate goals involving a monotonic counter or meter cannot be tested by saturating it with real actions, because the same visual feature may encode a loss budget. Recorded evidence or one reversible probe is preferred.

We use the analytical relation `assumed -> verified`, where `verified` means supported by the relevant observation oracle under a declared contract, not globally correct. A simulated terminal state cannot verify a goal predicate; real completion must do so. Losing claims and their falsifying evidence remain in the reasoning record so later observations can reopen them.

Because escalation is sparse, not every run materializes every contract. THEA types decision-critical forks while leaving ordinary control state in prompts, programs, and durable records. The authority labels are analytical vocabulary.

3.4 Prompt-Programmed Epistemic Escalation

THEA preserves the executable fast path while making costly revisions conditional. An observable anomaly or declared checkpoint can invoke a specialist that reads the shared evidence under a different epistemic contract, returns a bounded artifact or verdict, and exits. The parent incorporates that result and alone retains authority over real environment actions.

This orchestration admits a graph abstraction. A node is the parent fast path or a specialist role prompt; an edge is an observable guard or checkpoint; graph state is the executable theory, interaction history, logs, and artifacts; a transition invokes a role; and a join returns its artifact or verdict to the parent. Prompts and the shared workspace implement these semantics, either through provider-native subagents or a persistent specialist interface. The graph describes control flow rather than every inference the model makes.

Before deeper escalation, the parent applies a cheap-probe coverage gate. For each new object family or visual token type, it checks which available actions have been tested on that family in the current level. If a salient pair remains untested, it performs one representative probe unless the action is clearly irreversible or likely terminal. The probe must declare a hypothesis, expected observation, and model update. This gate avoids expensive reinterpretation when one direct interaction can resolve the uncertainty.

The policy combines failure predicates with checkpoints. Approximate counters, such as repeated failed planners, attempts, or repairs, are escalation boundaries, not game-specific constants. Specialist calls are suppressed while a verified goal-reaching plan is executing, and synchronous calls have bounded waits. A stalled role is not repeatedly relaunched. These rules keep epistemic escalation subordinate to task progress.

The parent remains responsible for the theory, but each join has force: critic findings require an explicit disposition, causal challenges remain open until a declared predicate separates the alternatives, and reinductor candidates must match recorded history before promotion. Section 3.5 specifies the guards and outputs for each role.

3.5 Specialist Roles in the Revision Workflow

The specialists are bounded operations over shared evidence. Each reads the current theory, code, traces, and reasoning logs directly. Their activation and join contracts are summarized in Table 2.

The curator checks whether the agent has mistaken an indicator for an objective or preserved contradictions across artifacts. The critic asks whether the implementation encodes a mechanic or merely the layouts already seen. The transition observer supplies a second reading of visual event order.

3.5.1 Independent Investigation

The Policy Investigator treats the standing hypothesis family as suspect after cheap interactions have been exhausted. This matters when ASCII preserves cell values but hides visual type: outlined, solid, hollow, and fixed objects may share a token while having different causal roles. It reports the missed observation, the conflict with the standing frame, and a structurally different interpretation. In the `g50t` chain, re-

Specialist	Activation	Evidence read	Required output and join
Framing Curator	Roughly six exploratory actions without a verified route, before committing to a solution, or after `GAME_OVER`	Theory, model code, planner, and recent reasoning	Confirm continuation or identify a concrete framing defect and next check; suppress during verified execution and back off after repeated unchanged confirmations
Generalization Critic	Growing complexity, a proposed special case, or a pre-solution checkpoint	Cross-level ontology, state variables, planner assumptions, and special cases	A structural finding that the parent must `ACCEPT` and fix or `REJECT` with evidence
Policy Investigator	At least two failed auxiliary planners or five failed attempts, after cheap-probe coverage is exhausted	Raw frames, animations, traces, earlier mechanics, and parent rationale	A missed observation and one or two structurally different hypothesis families
Epistemic Challenger	A real contradiction, unresolved goal, or causal-control mismatch for which a cheap discriminating probe exists	Standing theory, rival-consistent evidence, and relevant trace	A one-axis rival plus a declared predicate in `challenge.toml`; the parent executes the probe and resolves the fork
Reinductor	No planner route after roughly eight repairs to the same mechanic	Full relevant transition history and current engine	A simpler candidate and verifier, or a residual report naming missing evidence
Transition Observer	Intermediate frames make a recorded transition ambiguous	The exact animation sequence and settled endpoints	An independent account of what changed and in what order, reconciled with the parent's reading

Table 2: Specialist roles: activation guards, evidence read, and required output and join contracts.

examination of archived frames exposed continued motion and latent state that an exact-replay proxy had omitted. In `ls20`, the same operation found salient controls that had never been tested.

3.5.2 Rival-Hypothesis Testing

The Epistemic Challenger turns an unresolved alternative into a test. It states H_{current}, changes one axis in H_{rival}, and declares a predicate on which their next predictions differ. Supported axes include `actor`, `control_effect`, `win_relation`, `cosmetic_vs_causal`, `goal_predicate`, and `hidden_state`. The rival inherits every undisputed part of the current theory.

For dynamics disputes, the role may write a minimal rival engine; for goal or cosmetic disputes, it can reuse current dynamics and change only the relevant predicate. The parent must confirm that the probe discriminates before acting. In `ls20`, a horizontal approach separated fixed rotation from a direction-dependent mirror after all vertical observations had left them correlated. In `tr87`, a goal-axis contract separated exact-sequence matching from set membership. An earlier `sp80` call instead spent about 44 minutes on physics during a goal dispute, motivating the current one-axis scope and effort ceiling.

3.5.3 Causal Reinduction

The Reinductor addresses theory accretion. It extracts the relevant history, implements a proposed law in a candidate engine, and compares it with the live model using a separate verifier. The candidate can be promoted only if it matches or exceeds historical fidelity with materially fewer rules. The parent then reruns model and planner verification.

A failed induction still returns the residual transitions, what they share, the hidden state or missing observation that could explain them, and the cheapest discriminating probe. In `su15`, additional evidence supported a single law that replaced growing hazard exceptions. In `ls20`, the role identified a bounded autonomous-motion model. We do not attribute the successful `sk48` chain to this component because the role was available but not invoked.

Across archived runs with complete provider-native accounting, specialists represented approximately 0.29% to 5.42% of full-session API-equivalent cost. These are cost measurements, not evidence that every call was useful or that any role caused an outcome difference.

3.6 Evidence-Gated Selective Fidelity

Exact prediction of the complete settled frame is THEA's default fidelity contract. A mismatch stops execution because a remote icon, counter, door, or single changed cell may encode causal state that the current theory has missed. The parent must explain the discrepancy, update the executable theory, and replay the recorded history before continuing. Selective fidelity is not permission to ignore inconvenient pixels. It is an evidence-gated change to the verification contract after the decision relevance of a region has been tested.

The principal exception concerns irregular display regions. A HUD, score, move counter, progress strip, or decorative animation may change according to a cadence that depends on action type, color, position, or recent history while remaining irrelevant to the task dynamics. Reproducing every increment can then dominate the interaction even after the causal route is understood. THEA allows such a region to be excluded from exact render comparison only under the following conditions:

1. The agent first models the region well enough to determine whether a threshold can cause `LEVEL_COMPLETED` or `GAME_OVER`.
2. At least one level has completed through another mechanism, and positive evidence shows that the display's own threshold is non-terminal. Positive evidence requires an observed near-full or full display without a terminal event, or a mechanistic reason that no threshold effect exists.
3. The ignored region is limited to the confirmed display. Every other cell remains under exact blocking verification.
4. Masking changes rendering verification only. Until the threshold is proven harmless, the planner still treats the display as a possible action, shot, or time budget and prefers an efficient route.

This separates render relevance from strategic relevance. The absence of failure on a short level does not prove a counter neutral if it never approached its maximum. Likewise, masking an irregular cadence does not make additional actions free. A region may be cosmetic as a pixel process while its accumulated value remains terminal.

The gate was motivated by contrasting trajectories. In `re86`, exact modeling of an irregular display consumed about 200 additional actions after the route was known. In `cn04`, masking too early hid a real move budget. The resulting rule separates pixel-level render fidelity from strategic relevance: a region may be masked only after its terminal significance has been tested, and route efficiency remains required. Section 7 reports these failures in detail.

THEA contains a second, narrower fidelity contract for an autonomous mechanic whose exact transition is proven under-determined by settled frames. It permits planning against a conservative behavioral envelope only when five hard gates hold: the object has its own state-dependent dynamics; repeated reinduction classifies the residual as observationally under-determined; a worst-case bound contains every recorded transition; success does not depend on the object's exact final state; and a loss-side invariant can be checked from each real frame. The agent re-anchors the object's observed state after every action and returns to exact repair if reality exits the bound. Counters and responses to the agent's own actions are categorically excluded. This branch did not activate positively in the evaluated ARC archive, so no result claim depends on it.

3.7 Verification Contracts and Their Limits

THEA uses verification to assign bounded authority to an executable theory. Three contracts must remain separate; Table 3 summarizes them.

Contract	Question answered	Oracle	What a pass does not establish
Historical world-model fidelity	Does the theory reconstruct and reproduce every recorded action, terminal status, and settled frame it claims to cover?	Archived real interaction history	Causal identification outside the observed trajectories
Planner adequacy	Does the planner return a sequence that the engine predicts will reach `LEVEL_COMPLETED` from a reconstructed state?	Executable engine and encoded completion predicate	Correctness of the engine or the goal predicate in the real environment
Goal confirmation	Does the encoded completion predicate agree with real terminal outcomes?	Real `LEVEL_COMPLETED` and non-terminal observations	Reachability of that goal under the current dynamics and planner

Table 3: Verification contracts, their oracles, and the limits of a pass.

`verify_theory.py` implements the first contract by replaying every recorded attempt through the current level. `verify_planner.py` implements the second for levels already completed in the real environment. The third is enforced by requiring every proposed goal predicate to agree with the terminal status of all recorded actions. A simulated state cannot promote a goal claim to verified status when the corresponding real state remained non-terminal.

The separation matters because each contract can pass while another fails. A faithful dynamics model may be paired with a planner whose state key omits a causal variable, whose coordinate system is inconsistent, or whose search is too shallow. Conversely, a planner can reach `LEVEL_COMPLETED` inside an engine whose encoded status rule is wrong. A `None` result on the current unfinished level is a diagnostic event, not proof of impossibility, because the planner verifier has no completed trajectory against which to validate that claim.

THEA therefore complements fit coverage with discriminating coverage. For a specific disputed axis, a useful probe must produce different predicted values under H_{current} and H_{rival}. The challenge contract operationalizes this locally: it rejects a probe when both hypotheses predict the same declared predicate, even if the action would add another perfectly modeled frame. The goal is a controlled desynchronization between theories, not the accumulation of observations under conditions where their predictions already coincide.

The harness enforces discriminating coverage locally rather than collapsing claim-specific evidence into a global scalar. At decision-critical forks, `challenge.toml`, independent investigation, and boundary-condition checks identify the variable that must change. Exact replay constrains arbitrary theories, but its authority is greatest when paired with an intervention that separates rival explanations.

3.8 Design Evolution and Rationale

THEA emerged from several architectural lines. The design genealogy is summarized in Table 4 by the mechanisms that survived.

Our line began in April 2026 with a multimodal agent inspired by Symbolica's approach, whose before/after visual analysis established observational change as an explicit evidence channel. Over the following two months it produced an early code-executing RLM prototype built on DSPy (Khattab et al., 2023), a monolithic REPL agent that proved single-context autonomy, a Gemini-based typed pipeline whose hypothesis and route ledgers became THEA's belief typing, a terminal-bridge phase that exposed context compaction as an epistemic risk, and a world-model committee whose investigator, critic, curator, and challenger roles became THEA's bounded specialists. External systems were absorbed as evidence favored them: TELL demonstrated durable per-game mechanical memory and motivated our doctrine of transferring a control's functional role across levels and relocating it by effect; RGB contributed the pattern of full textual state plus explicit diffs. In June 2026, we took the executable fast-path substrate of the EWM line as the seed of THEA's

Stage	Architectural move	Lesson retained in THEA
Multimodal roles	Separate before/after visual analysis	Treat observational change as an explicit evidence channel
Programmatic REPL	One agent inspects, codes, tests, and acts in a persistent runtime	Preserve causal context and programmatic autonomy in the core loop
Typed routes	Schemas for claims, evidence, route state, and review	Type decision-critical forks, not the entire reasoning process
World-model committee	Separate investigation, tactics, criticism, memory, and curation	Retain adversarial review but centralize action authority
Executable model with guarded specialists	Unified model-plan-act loop with conditional revision branches	Escalate from observable epistemic state rather than permanent orchestration

Table 4: Design genealogy: architectural stages and the lessons retained in THEA.

direct path, after it performed strongly under our adversarial-subagent concept, and grew THEA around it: rewritten operating prompts that tested better, a revised escalation policy, and its scheduled simplification passes removed. Our implementation includes modified MIT-licensed source from `arc-3-agents-baseline1` for that substrate. DreamTeam and related multi-agent systems bound our novelty claims. Our experiments with typed phase machines improved provenance but also accumulated handoffs and constrained reframing; committee architectures made facts, hypotheses, and criticism explicit, yet repeatedly lost causal context between investigation and action.

This evolution was disciplined rather than exploratory. Changes entered the harness under a ceteris paribus protocol: one deliberate variable per experiment, with the baseline configuration, the expected effect, and the observed trajectories recorded. Across the four generations this produced 1,038 preserved development runs, each stored with its commit hash, configuration, and full logs; mechanisms that survived contact with multiple games were retained, and the rest were reverted and recorded. The genealogy above therefore reflects tested judgments rather than design intentions.

The final design is asymmetric. The executable loop remains centralized and fluid; specialist branches are bounded, evidence-driven, and auditable. This history is design rationale rather than a controlled comparison among predecessor architectures.

4 Experimental Protocol

4.1 Evaluation Scope

Our evidence view is the historical portfolio: it records, for each of the 25 public games, the last trajectory the development line produced for that game. It is drawn from the development record of 1,038 preserved runs described in Section 3.8: each entry is the final preserved trajectory for its game, not an outcome-selected best attempt. When a run exposed a limitation, we abstracted the lesson into a game-agnostic revision of the harness and continued; a game's entry is its final recorded run under the then-current harness, drawn from a single trajectory rather than chosen among repeated samples. Because work on a game typically stopped once the harness handled it well, the final run usually reflects the most mature harness version applied to that game, and the portfolio spans the successive versions produced along the way.

4.2 Evaluated THEA Configuration

The final locked agent-facing configuration is as follows. The parent is Claude Code Opus 4.8 at `xhigh` effort. Specialist roles use provider-native Sonnet 5 subagents under `advisor_mode=native`. The Framing Curator, Generalization Critic, Policy Investigator, Epistemic Challenger, Reinductor, and Transition Observer are available through the guards described in Section 3. A separate automatic visual-inspector path is disabled.

Table 5: Final locked THEA configuration.

Dimension	Locked value
Parent	Claude Code Opus 4.8, `xhigh` effort
Specialists	Provider-native Sonnet 5 roles
Specialist routing	Native role invocation, guards from Section 3
Separate visual inspector	Off
Provider environment	Clean settings; strict empty external-tool configuration
Workspace	Fresh isolated directory outside the repository
ARC runtime	Competition mode, `arc-agi==0.9.8`

The evaluated controller starts with clean provider settings and an empty external-tool configuration, uses an isolated workspace, and runs the ARC server in competition mode with `arc-agi==0.9.8`. The exact configuration is summarized in Table 5.

Three portfolio entries (`ft09`, `sb26`, `tn36`) report trajectories recorded under a GPT-5.5 codex parent rather than the Opus 4.8 parent used for the other games. We used these three as a deliberate cross-provider check: we ran them under both parent configurations to test whether different parent models produce comparably good results under the same specialist-and-verification harness. The two configurations reached comparable results, so we keep the codex trajectory as the reported entry while noting that the harness reaches full solutions under both a GPT-5.5 codex parent and an Opus 4.8 parent. This supports the control policy being provider-general rather than tuned to a single model.

No value in this configuration is selected per game. The prompt contains no game identifier, game-specific mechanic, target coordinate, solution trace, or prewritten game planner. The agent creates all game-dependent executable theories and planners inside its fresh workspace. THEA itself is the prompt-programmed revision method defined in Section 3; the parent and specialist models above are the evaluated ARC instantiation of that method.

4.3 Systems and Comparators

The main comparison is among complete systems. The strongest direct quantitative reference is `secure_baseline1_v1.5`, an executable world-model agent evaluated on the same 25 public games with GPT-5.5 at `xhigh` effort, zero run-time memory transfer across games, and the same 1,500-action limit on the current level.

We also compare against TELL, DreamTeam, and other public ARC-AGI-3 agents when their released artifacts support the metric being shown. Results from RGB and other Preview Competition systems are restricted to the games present in the public preview overlap; they are not extrapolated to the 25-game suite. A missing action, cost, or per-game score field remains missing; we never infer it from an unrelated aggregate.

4.4 Outcome and Action Metrics

We report levels completed, full-game completion, per-game Relative Human Action Efficiency (RHAE), and the arithmetic mean of per-game RHAE across the 25-game suite. Levels completed is shown alongside RHAE because a low score can mean either poor efficiency on completed levels or incomplete coverage.

Following the official ARC-AGI-3 scoring methodology (ARC Prize Foundation, 2026b), let $h_{g\ell}$ be the upper-median human reference action count from first-time play for level ℓ of game g, and let $a_{g\ell}$ be the agent's cumulative counted actions for that level. For a completed level, the level efficiency is

$$r_{g\ell} = \min\left(115,\ 100\left(\frac{h_{g\ell}}{a_{g\ell}}\right)^2\right).$$

An unfinished level contributes zero. Levels receive ordinal weights $w_\ell = \ell$, making later levels progressively more important. We compute the weighted efficiency and coverage ceiling as

$$\bar{r}_g = \frac{\sum_{\ell=1}^{L_g} w_\ell r_{g\ell}}{\sum_{\ell=1}^{L_g} w_\ell}, \qquad c_g = 100 \frac{\sum_{\ell=1}^{L_g} w_\ell \mathbb{1}[\ell \text{ completed}]}{\sum_{\ell=1}^{L_g} w_\ell}.$$

The reported per-game score is

$$\mathrm{RHAE}_g = \min(100, \bar{r}_g, c_g).$$

The level cap of 115 rewards superhuman action efficiency, while the game cap and coverage ceiling prevent efficiency on early solved levels from compensating for later unfinished levels. We prefer the official environment score. A reconstructed score is used only when the archived trajectory contains per-level action counts and the corresponding official human references.

For action accounting, we report three complementary quantities: actions used on completed levels, actions used on the final unfinished level, and total actions at termination. We also report resets as a decomposition of the total. The official `total_actions` already includes `RESET`, so resets are never added a second time. When an official total is unavailable, the fallback counts one incoming benchmark action request per action and records that source. All efficiency comparisons use per-level counts when available because identical game totals can hide very different distributions across levels.

4.5 Resource Accounting

ARC actions measure interaction efficiency but not the computational cost of building and revising a theory. We report wall-clock runtime, model tokens, and an API-equivalent cost in US dollars. Token accounting is split by provider and model into non-cached input, cache creation by available cache class, cache reads, output, and reasoning output when the provider exposes it. Native parent and specialist sessions are identified separately.

Totals deduplicate native sessions and provider message identifiers. For a cumulative session counter, only the largest valid cumulative value is used; snapshots of the same session are not summed. Reported game costs use the complete archived native session, including model activity after the terminal environment event. The specialist cost fraction is API-equivalent cost attributed to specialist sessions divided by the complete deduplicated cost for the game. Through-terminal figures are stated explicitly when used as a secondary diagnostic.

API-equivalent cost applies a versioned price table, frozen by model and date, to the native token categories. It is a normalization for comparing systems, not a claim about literal subscription spending. When native telemetry is partial, the record is marked `partial` or `unresolved`; any computable amount is treated as a lower bound and is not silently compared as a complete total.

4.6 Evidence Integrity and Reporting Boundaries

Each reported trajectory is linked to its source configuration and an evidence archive. The archive contains the action sequence, resets, frame and terminal metadata, `results.json`, the scorecard response when available, agent and server logs, the final workspace, executable theory and planners, workspace version history, and native parent and specialist transcripts.

The isolated workspace reduces accidental access to repository artifacts. We also audit command histories and native transcripts for reference access. Trajectories with detected access to a game solution are excluded outright, never merely flagged with a warning.

Table 6: Public reference systems and THEA result views.

System or result view	Model and configuration	Scope	Fully solved	Mean RHAE	Interpretation
Executable World Models v1	GPT-5.5 high	25 games	15/25	58.12	Original code-world-model baseline
TELL	Claude Opus 4.6, single agent	25 games	11/25	43.90	One trajectory per game
DreamTeam	Multi-agent workspace optimization	25 games	not emphasized	38.36	Mean of two trajectories per game
Executable World Models v1.5	GPT-5.5 xhigh	25 games	20/25	82.01	One clean trajectory per game
THEA historical portfolio	Opus 4.8 xhigh, Sonnet 5 specialists	25 games	22/25	92.63	Final development-line trajectory per game

5 Results

5.1 Reporting Views and Benchmark Context

We report the historical portfolio of Section 4.1 at the archive freeze of July 19, 2026: each game's final recorded trajectory from the development line.

The comparison in Table 6 places the portfolio beside selected public reference systems. The systems differ in base model, test-time compute, action policy, number of trajectories, and whether the reported row is a single configuration. The original executable-world-model system reports 15/25 wins and 58.12 mean RHAE with GPT-5.5 (Rodionov, 2026a); its later v1.5 release reports 20/25 and 82.01 under a single clean GPT-5.5 xhigh playthrough per game (Rodionov, 2026b). TELL reports 11/25 and 43.9 with one Claude Opus 4.6 trajectory per game (Dots Post-train Team, 2026), while DreamTeam reports 38.36 mean RHAE averaged over two multi-agent trajectories per game (Sarafian et al., 2026). These are the closest public systems for architectural context. Results published only on the three-game preview, including RGB, are not pooled with the 25-game rows.

Three of the games above (`ft09`, `sb26`, `tn36`) report trajectories recorded under a GPT-5.5 codex parent and were cross-checked under the Opus 4.8 parent with comparable results; see Section 4.2.

5.2 Historical Portfolio and Hard-Tail Outcomes

The historical portfolio evaluates all 25 public games and fully solves 22, with a mean of 92.63 per-game RHAE across final recorded trajectories. Table 7 reports the complete per-game record.

The portfolio remains incomplete on `bp35`, `dc22`, and `lf52`. Section 6 examines several of these trajectories, and Section 7 reports mechanisms that produced overhead or regression.

5.3 Aggregate Cost and the Location of Savings

Native-session accounting is complete for all 25 selected portfolio trajectories. THEA totals $5,007.85 in API-equivalent cost across the portfolio; a secondary cut that stops accounting at the final environment action totals $4,853.20. Cost is not a monotonic function of score or coverage: expensive successes and cheap failures both occur, and a high-cost success does not show that its additional deliberation was necessary. Against the strongest published executable-world-model release (v1.5: 20/25 games, 82.01 mean RHAE), the portfolio's aggregate API-equivalent cost is 19.52% lower ($5,007.85 versus $6,222.18), cheaper on 14 of the 25 games and concentrated in the five games that release leaves incomplete. These are complete-system comparisons; per-game head-to-head tables are maintained in the released repository, which regenerates them as comparators evolve.

Table 7: Complete per-game record of the THEA historical portfolio.

Game	THEA levels	THEA RHAE	THEA actions
ar25	8/8	100.00	255
bp35	7/9	62.22	421
cd82	6/6	83.32	178
cn04	6/6	100.00	324
dc22	5/6	71.43	1010
ft09	6/6	100.00	101
g50t	7/7	100.00	432
ka59	7/7	100.00	305
lf52	5/10	27.27	617
lp85	8/8	100.00	123
ls20	7/7	98.29	759
m0r0	6/6	100.00	270
r11l	6/6	100.00	78
re86	8/8	100.00	718
s5i5	8/8	84.78	421
sb26	8/8	100.00	157
sc25	6/6	97.63	293
sk48	8/8	100.00	374
sp80	6/6	100.00	213
su15	9/9	90.72	243
tn36	7/7	100.00	124
tr87	6/6	100.00	269
tu93	9/9	100.00	257
vc33	7/7	100.00	214
wa30	9/9	100.00	1443

6 Failure-Mode Case Studies

6.1 Reading the Cases

The cases below are retrospective selections from the historical development portfolio. They were chosen because their archives expose different failure loci, not because they form a representative sample of ARC-AGI-3 games or an ablation of THEA's components. Outcomes provide experimental context, while the primary evidence is the sequence of predictions, observations, executable artifacts, and revisions inside each trajectory. A specialist contribution is described when its returned artifact changed the investigation, but no case is used to estimate the isolated causal effect of that role.

To situate the cases, Table 8 counts specialist activations in the archived native transcripts of all 25 selected trajectories, one row per game (parent provider in parentheses; the three codex-parent runs predate the role-labeled spawn interface, so their critic and transition-observer columns are classified from each child transcript's content and carry that caveat).

Three descriptive regularities emerge. The curator is the most invoked role (68 of 194 activations) and the only one active in nearly every trajectory. The investigator concentrates in grinding runs: the two trajectories with six investigator activations each (`g50t`, `ls20`) are also the busiest Claude-parent runs overall. The reinductor is rare (seven activations in four games), and the two codex-parent outliers (`ft09`, `sb26`) owe most of their volume to per-step visual reading that the multimodal Claude parent later performed itself, invoking the transition observer only eight times across 22 games. These figures count activations; they do not estimate effects.

Each case follows the same analytic pattern: the standing executable theory, an observable failure signature, the contract that failed, the evidence that changed the diagnosis, the resulting revision, and the boundary of the claim. The cases span six distinct loci (Table 9).

Game	Curator	Critic	Investigator	Challenger	Reinductor	Transition obs.	Total
ar25	1	2	0	0	0	0	3
bp35	3	0	1	1	0	0	5
cd82	0	1	0	1	0	0	2
cn04	2	1	1	2	1	0	7
dc22	1	1	2	0	0	0	4
ft09 (codex)	14	7	1	2	0	4	28
g50t	3	1	6	0	0	0	10
ka59	1	2	0	0	0	0	3
lf52	2	1	2	0	0	1	6
lp85	2	0	0	0	0	0	2
ls20	5	1	6	1	2	0	15
m0r0	2	1	1	0	0	0	4
r11l	1	1	0	0	0	1	3
re86	2	2	1	0	0	0	5
s5i5	1	0	2	1	0	0	4
sb26 (codex)	10	8	2	0	0	22	42
sc25	2	0	2	1	0	1	6
sk48	1	1	2	2	0	1	7
sp80	0	0	1	0	0	2	3
su15	3	0	2	0	2	1	8
tn36 (codex)	5	0	0	1	0	4	10
tr87	1	0	1	2	0	0	4
tu93	1	1	1	0	0	1	4
vc33	3	0	2	0	0	0	5
wa30	2	0	0	0	2	0	4

Table 8: Specialist activations per archived trajectory in the historical portfolio.

Case	Apparent assurance	Hidden defect	Decisive evidence
g50t	Exact replay of every recorded settled frame	A correlated proxy replaced the causal state	Raw-frame motion continued after the avatar stopped
s5i5	Exhaustive search closed under the encoded dynamics	A legal transition was absent from the theory	A load-bearing predicted refusal was tested directly
bp35	The model reproduced the archived trajectory	Search identity and coordinates did not represent the live problem	Structural audit of state keys and heuristic coverage
ls20	A mismatch appeared to demand new environment physics	The executable theory corrupted its own state	Adversarial code inspection found reused variables
su15, with sk48 contrast	Local patches preserved fit to accumulated observations	The theory had become a collection of contextual exceptions	Reinduction found a smaller law; open-field evidence exposed a hidden boundary condition
lf52	A corrected model eventually explained the observed dynamics	The correction arrived after irreversible resource loss	A state-complete search proved the remaining trajectory dead

Table 9: Failure loci and decisive evidence across the six case studies.

6.2 Exact Replay, Wrong Cause: g50t

In g50t, the agent had to infer how an avatar, a key, a gate, and a recorded path interacted. Its standing theory treated the path as an elastic, stateless relation. Along the trajectories observed at that point, the length of the recorded route happened to equal the separate distance that controlled the gate interaction. The proxy therefore remained synchronized with the causal mechanism. The executable model reproduced every

archived settled frame exactly, so additional replay of the same trajectories increased confidence without increasing identifiability.

The model appeared fully verified, yet planning kept failing, a signature distinct from a replay mismatch. This triggered an independent policy investigation. Instead of extending the parent's theory, the investigator returned to raw PNGs and intermediate animation frames that had already been captured. It found two observations suppressed by the standing abstraction: the key changed position, and visible motion continued for several frames after the avatar had stopped. A stateless elastic relation could not account for that temporal separation. The later discriminating trajectory varied the quantities that had previously moved together and revealed the omitted recorded-path state.

The completed trajectory solved 7/7 levels with a game score of 100 in 432 actions (per-game comparisons appear in Table 7). The supported claim is that historical exactness establishes consistency with sampled trajectories, not causal identity. When latent variables remain correlated throughout the archive, a wrong causal theory can be perfectly replay-compatible. Independent reinspection and a probe designed to desynchronize rival explanations add a different kind of evidence from another matched frame.

6.3 A Closed Search Under an Open Theory: `s5i5`

The `s5i5` trajectory shows the same limitation under exhaustive search. In level 7, the agent modeled a chain of articulated bars moving through narrow gaps. Under the standing engine, the delivery route required two orientations that appeared disconnected. It enumerated the reachable state space: 819,208 states under an initially permissive edge rule, followed by 247,940 states after a boundary correction. Both searches closed without a winning state, and the closest modeled terminal remained six cells short.

The searches were computationally correct relative to the engine. It represented a bar orientation that folded back onto its parent as an ordinary colliding destination, so the corresponding turn should be refused. The impasse broke when the agent audited which load-bearing predictions had actually been observed. Many predicted refusals had been confirmed, but the transition sequence around the supposed trap had not. Three direct clicks produced relative orientations `3 -> 0 -> 1 -> 3`: the environment skipped the fold-back orientation rather than entering it or refusing the whole turn. Adding that transition semantics connected the previously separated regions of state space and yielded a verified 45-click plan for the level.

Level 8 provided a related warning. A measured growth limit observed for one bar was initially promoted to a global cap. Once a different obstruction was removed, another bar exceeded that apparent cap, falsifying the generalization and opening the final route. The trajectory ultimately completed 8/8 with a score of 84.78 in 421 actions and one reset. The contribution of `s5i5` is epistemic: exhaustive closure proves reachability only under the supplied transition system. A proof of impossibility should inherit the uncertainty of the theory on which it is built, especially when decision-critical transitions lack positive coverage.

6.4 Verified Dynamics, Invalid Planner: `bp35`

`bp35` separates world-model fidelity from planner validity. By level 8, the executable theory reproduced the archived interactions it claimed to cover, but multiple planners failed to return an executable policy. One search used a mostly positional heuristic; another recomputed distances under changing door topologies. Both were active computations rather than deadlocks, yet they failed to improve the decision frontier. The more detailed search also grew to approximately 9 GB of physical memory in an observed sample.

The eventual audit found defects below the level of environment dynamics. First, the main search key omitted the complete door configuration. Two action orders could therefore receive the same key while producing boards that differed by 90 doors, allowing valid scaffolds to be pruned as duplicates. A related distance cache had an analogous omission and was repaired earlier. Second, the heuristic graph used a local coordinate range of roughly `-5..67`, while the live map extended to approximately `-144..62`. The live pose was outside the graph, so the supposedly structure-aware planner often fell back to a much weaker relaxed distance. Third, a hierarchical subplan was summarized by its final pose even though its residual scaffold determined whether the next stage remained feasible.

The agent stopped the unproductive searches before executing their prefixes in the environment. The trajectory finished at 7/9 after 421 total actions, including 112 on the unfinished level; its reconstructed score is 62.22. The case establishes that model fidelity, causal sufficiency of the search state, heuristic coverage, and operational tractability are separate contracts. More search time cannot repair a key that merges states with different futures or a heuristic whose graph excludes the current state.

6.5 When the Contradiction Is in the Agent: `ls20`

In `ls20` level 2, the agent observed an arithmetic contradiction between the predicted and rendered locations of a dock. A natural response was to posit an additional environmental displacement rule. That response would have expanded the causal theory around evidence produced by the theory's own implementation error. An adversarial inspection instead found that reused `bh` and `bw` variables corrupted the dock coordinates inside the executable engine. Once the code was corrected, the apparently impossible frame no longer required new physics.

The broader trajectory illustrates why escalation is typed. In level 6, the residual problem was a growing collection of autonomous-motion rules, and the reinductor returned a usable two-phase movement law. In level 7, the agent was committed to the frame that no rotation source existed. The policy investigator reported a different coverage failure: a salient switch and pad had never been touched. A cheap direct probe of the untested control revealed the missing operation. These were three superficially similar instances of stuck planning, but their repair targets differed: implementation, theory compression, and unobserved control coverage.

The final trajectory completed 7/7 with score 98.29 in 759 actions. `ls20` supports a diagnostic distinction: a contradiction may falsify the world hypothesis, the code that instantiates it, or the evidence coverage on which it rests. Treating all three as requests for more environmental rules encourages exactly the branch accretion that an executable theory is meant to prevent.

6.6 Reinduction Versus Rule Accretion: `su15`, with `sk48` Contrast

`su15` contrasts two responses to a theory that no longer compresses: local rule accretion and explicit reinduction. An earlier `su15` trajectory reached 4/9 and then spent 401 actions across 28 attempts on level 5. Its engine accumulated local clauses for autonomous chasers. A simulated 15-action plan still diverged at action 7, after which the agent returned to another local repair cycle. The diagnostic signature was a fidelity plateau: additional branches preserved some past cases without producing a stable predictive law.

In the later successful trajectory, the reinductor was invoked twice. It classified whether the remaining residuals were derivable with more evidence, genuinely underdetermined, or irrelevant to the terminal decision. With new observations, it rejected the tempting approximation path and derived a compact distance-conditioned hazard law together with the relevant loss threshold. The trajectory completed 9/9 with score 90.72 in 243 actions.

The supported observation is that explicit reinduction produced an actionable smaller law at the documented impasse. The same run also contains a premature judgment that a visual region was neutral, and two levels remained relatively action-expensive. Thus, revision quality depends on the evidence supplied to it.

`sk48` provides a useful causal boundary. An earlier development trajectory stalled at 3/8 with an accreted engine. A later trajectory represented the game with compact laws for articulated rods, cascades, ordering, and rail limits, then completed 8/8 with score 100 in 374 actions. A rule learned under confinement nevertheless failed in open space because the earlier geometry had hidden a boundary condition. This supports the broader hypothesis that compact causal laws and boundary tests can replace coordinate replay.

6.7 Correct Too Late: `lf52`

`lf52` is the main negative case. The first five levels exposed several errors in temporal state reconstruction under a moving camera. Objects outside the visible region were omitted, a pixel operation erased state underneath a peg, and the first appearance of a cart was treated as its creation. After these corrections, the

executable theory reproduced all 125 archived frames from level 5. The corrected reconstruction improved local fidelity but did not prevent the consequential failure in level 6.

At that level, the planner initially returned no route from the opening. The result depended on an incomplete camera law, a state identity that omitted the causal `camera_peg`, and a probe that queried a graph with the wrong key contract. The agent later invalidated the graph conclusion, derived the correct camera relation, and showed that the supposedly sealed eastern region was reachable from a fresh state. By then, however, discovery actions had irreversibly consumed material in the live attempt.

A corrected search over 21,018 states never reduced the remaining requirement below three pegs. The current state no longer contained enough material, and the environment offered no reachable restart. The corrected theory therefore explained both how the level could have been solved and why the live trajectory could no longer solve it. The agent stopped at 5/10 with score 27.27 after 617 total actions. It had used 361 actions on the first five completed levels, but that local efficiency did not compensate for zero coverage of the later levels.

This case places a necessary limit on the paper's thesis. Auditability can expose an invalid state key, retract a false reachability conclusion, and explain a dead trajectory. It does not make revision timely, restore consumed resources, or guarantee control success. In environments with irreversible actions, the value of a correct theory depends on when it becomes available. Probes must therefore be assessed not only for expected information but also for the states they may permanently remove.

6.8 Cross-Case Synthesis

The cases show why verification cannot be reduced to a single pass bit. In `g50t`, historical replay passed while causal identity failed. In `s5i5`, search closure passed while the transition system was incomplete. In `bp35`, the world model passed while search identity and heuristic coverage failed. In `ls20`, the apparent environmental contradiction originated inside the executable artifact. In `su15` and `sk48`, local fit concealed either theory accretion or an unvaried boundary condition. In `lf52`, the final theory was more accurate but arrived after the useful control frontier had closed.

A common pattern nevertheless emerges. Each productive revision began by typing the failed contract before adding complexity. Raw-frame inspection challenged omitted causal state; a load-bearing transition test challenged a model-relative proof; state-key audits challenged planner identity; adversarial code review challenged the implementation; reinduction challenged the theory's description length; and irreversibility analysis challenged whether further exploration retained control value. The specialist roles matter here as different prompts for evidence, not as a permanent committee that votes on one undifferentiated notion of correctness.

Without establishing population frequencies or component effect sizes, these observations show that code world-model agents expose separable contracts and that contradictions can be routed to the model, planner, goal, implementation, or evidence coverage they actually threaten. The next section reports where the added machinery failed to earn its cost or introduced another failure.

7 Negative Results and Non-Results

7.1 Reporting Scope

THEA emerged through a sequence of hypotheses about memory, simplification, fidelity, and escalation. We report a negative result when an implemented mechanism produced an auditable outcome that contradicted its motivating hypothesis. We report a non-result when a mechanism remained available but the evaluation produced no positive instance from which to infer efficacy. This is not an inventory of every discarded experiment. Each item below changed the system design or constrains a claim made elsewhere in the paper.

The evidence is heterogeneous. One comparison is a direct mechanism test, others are closely observed development contrasts, and the final comparison is between complete systems with different models and

controllers. We state the corresponding boundary in each case, without treating them as a common ablation table.

7.2 Fresh Context Did Not Reduce Token Use

We hypothesized that starting a fresh model thread at each level would reduce token use by 30-50%. The executable theory and durable state already lived in files, so a new thread could in principle reconstruct only the state needed for the current level. The mechanism was functionally successful in `ft09` run 018: the agent completed 6/6 levels in 101 actions after repeatedly reconstructing its state.

It did not provide the expected efficiency gain. Aggregate accounting yielded approximately 750,000 tokens per action for the fresh-per-level trajectory, versus 706,000 for the continuous-thread comparison, about 6% worse. The lower raw total in run 018 was explained primarily by its shorter action trajectory, not by resetting context. Re-reading the workspace and re-establishing the current frame added overhead, while automatic context compaction already limited part of the continuous thread's growth. We reverted the mechanism as a cost optimization.

The two trajectories were not identical, and the older archive has less complete auxiliary-session accounting than later runs. The supported result is limited: level-boundary context resets did not reduce normalized token use in this measured comparison. Fresh contexts may still be useful for overflow control, but the present evidence does not support them as a general cost-saving device.

7.3 Calendar-Driven Simplification Broke a Verified Model

In `ft09` run 021, the agent had a model verified on the observed levels and a pending plan for level 5. A periodic instruction nevertheless requested that it compress the ontology and avoid defending the current model. The agent refactored the engine, broke state reconstruction, and lost the opportunity to execute the already available plan. Simplification was locally compliant with the maintenance instruction and globally harmful to the task state.

A verified goal-reaching plan should protect its supporting model from unsolicited refactoring; simplification becomes eligible again when no plan exists or real evidence has falsified the current one. The lesson is not that compact models are undesirable. It is that simplification is not monotonic and should not be scheduled independently of evidence and planning state.

Run 021 also differed from its comparison in parent model and persistent availability of specialist tools, so it supports neither a provider comparison nor a clean estimate of the guard's average effect. It is direct evidence of a specific failure mode: a periodic maintenance policy can destroy a verified capability at the moment it should be executed.

7.4 Permissive Masking Removed a Strategic Constraint

The first selective-fidelity rule allowed the agent to mask an irregular HUD region once it appeared irrelevant to level completion. In `cn04` run 071, the agent applied that judgment too early. The masked region was visually awkward but encoded a movement budget. Ignoring its rendering cadence became, in practice, permission to follow an action-expensive route. Level 4 consumed 161 actions and ended in failure.

The revised rule separated two contracts. Pixel cadence may be omitted from exact rendering only after positive evidence of visual neutrality, while any possible threshold, budget, or terminal semantics remains part of the policy state. In the later clean run 073, level 4 completed in 39 actions without a reset under this distinction. Related trajectories exposed both sides of the same problem: `re86` wasted actions reproducing a neutral display exactly, whereas `m0r0` avoided a large display-modeling detour while preserving the causal objective.

These runs triangulate the design requirement but do not form a controlled effect estimate. The negative result is the permissive rule itself: visual irregularity and apparent irrelevance to winning were insufficient evidence for strategic neutrality. A display can be cosmetic for frame prediction and causal for loss avoidance.

Game	Result	Actions	API-equivalent cost
cd82	6/6, score 83.32	178	$66.46
lp85	8/8, score 100	123	$157.70
tr87	6/6, score 100	269	$75.93

Table 10: Full-stack overhead on three tractable games.

7.5 A Guarded Envelope Produced No Positive Activation

The bounded-hazard envelope was introduced for a narrow situation: an autonomous mechanic must be explicitly judged underdetermined from settled frames, a conservative bound must cover every recorded transition, exact prediction must be unnecessary for winning, and a loss-side invariant must remain directly observable. Under those gates, variation inside the envelope would no longer block the rest of an otherwise verified plan.

Across the evaluated ARC trajectories, the full gate never opened. In su15, the most tempting case, this was the correct decision: additional evidence allowed the reinductor to derive an exact general law, so the residual had been difficult rather than underdetermined. The absence of activation is evidence that the guards avoided an obvious false positive. It is not evidence that the envelope improves planning when legitimately active.

We therefore treat the envelope as a guarded design hypothesis, not a validated capability. Its success rate, cost, and effect on control remain unknown until a qualifying case produces a positive activation and a comparable exact-model trajectory exists. This distinction prevents implemented flexibility from being counted as an empirical contribution merely because it remained inert.

7.6 The Full Stack Can Overpay on Tractable Games

The aggregate comparison of Section 5.3 is cheaper on 14 of the 25 games, with its savings concentrated where the comparator fails. Three games on the costlier side illustrate the overhead of deploying the complete epistemic stack where a direct executable-model path already converges (Table 10).

In cd82, the final theory was compact and both world-model and planner verification passed, yet the game finished below the maximum score. In lp85 and tr87, THEA reached perfect coverage and score while paying for the full deliberation stack along the way. The tr87 archive locates a plausible source of overhead inside our trajectory: 39 actions were spent on the level-1 completion code before commitment. This is a mechanism candidate, not an isolated cause.

These comparisons rule out a broader claim that the richer stack uniformly improves performance. Its value appears concentrated in hard-tail cases, while already tractable games expose the price of preventive or always-available deliberation.

7.7 Synthesis

Fresh contexts did not save tokens; poorly timed simplification and permissive masking caused regressions; the bounded envelope produced no positive activation; and the full stack sometimes cost more without improving outcome. These results motivate a simple fast path with failure-gated escalation, but do not establish that the current policy is optimal or causal. They mark where added structure repaired an observable failure and where it failed to earn its cost.

8 Limitations

8.1 Public-Set Adaptation and Generalization

ARC-AGI-3 exposes a public suite of 25 games, and the project evolved through months of interaction with that suite. Our use of *novel* refers to the agent's condition at the beginning of a trajectory: it receives no game instructions, game-specific planner, or carried state from another game and must infer the current environment within the run. It does not mean that the game is private, absent from model pretraining, or unseen by the researchers.

Our results support within-run adaptation to initially unknown interactive environments, not generalization to secret tasks. A private evaluation is required for the latter claim.

8.2 Limited Repetition and Experimental Power

Most reported game outcomes correspond to one selected trajectory per system and configuration. The environment is replayable, but the model's exploration, code generation, and revision choices remain stochastic. We do not have enough repeated trajectories to estimate per-game variance, confidence intervals, or the probability that a particular failure mode recurs.

The case studies were also selected retrospectively for mechanistic clarity. They demonstrate that a failure can occur and show how it unfolded; they do not estimate its prevalence. Small differences in score or actions should therefore be interpreted cautiously, especially when systems tie in coverage. Likewise, the negative results constrain specific hypotheses without forming a complete factorial evaluation of THEA.

8.3 Interface and Domain Specificity

The ARC interface is unusually compatible with executable theories. It offers discrete visual observations, enumerated actions, settled feedback after an interaction, explicit terminal states, and an environment that can be replayed for verification. These properties make prediction, reconstruction, and machine-checkable planning comparatively well defined.

Continuous control, noisy sensing, partial observability, open-ended language, and non-deterministic dynamics weaken those assumptions. The prompt-level control structure and distinction between model, planner, and goal contracts may be portable, but the ARC results do not establish performance transfer to robotics, textual worlds, or passive video. Those domains require separate protocols and evidence.

8.4 Compute and Planning Scalability

An executable theory creates useful audit points but can be expensive to build and search. The parent session dominates token use, and cheaper specialists do not remove that structural cost. Generated planners can also scale poorly: `bp35` produced a search observed near 9 GB of physical memory, while `s5i5` closed state spaces of 819,208 and 247,940 states before a missing transition was identified. Explicit verification can thus spend substantial computation proving a statement relative to an incomplete model.

The lower aggregate API-equivalent cost in Section 5 does not remove this limitation. That total combines historical configurations, is cheaper on only 14 of 25 games, and uses a dated price table, not literal subscription spending. Cost effectiveness is heterogeneous; it is not established for one frozen agent.

9 Conclusion and Outlook

Executable world models are useful because they can be run, inspected, and contradicted, but they remain provisional. A model can replay its recorded past while encoding the wrong cause, support exhaustive search while omitting a decisive transition, or represent dynamics faithfully while feeding an invalid planner. THEA treats the agent's current world as a provisional hypothesis and makes high-cost revisions explicit.

The method preserves a direct code-world-model fast path. One parent retains causal context and action authority, while six bounded specialists enter at observable failures or checkpoints. Their prompts and shared workspace form an abstract control graph whose joins return evidence, rival hypotheses, discriminating probes, candidate laws, or verdicts. Typed artifacts expose the forks that govern costly decisions without forcing ordinary reasoning through a rigid schema.

THEA solves 22 of 25 public games at a mean 92.63 per-game RHAE across the historical portfolio.

The evidence is bounded by a public benchmark, a development-line portfolio spanning successive harness versions, limited repetition, and no paired component ablation. The case studies therefore support mechanistic diagnosis rather than isolated effect sizes. They show why revision should be selective: the same machinery that exposed omitted state, invalid search identity, and accreted rules also overpaid on tractable trajectories and could arrive too late after irreversible actions. The next step is to calibrate when a claim deserves more epistemic work without losing the audit trail.

What transfers beyond ARC-AGI-3 is the epistemic-control pattern: an executable task theory that an agent builds, falsifies, and audits, revised by adversarial specialists under a single parent that keeps causal context and action authority. That pattern targets any setting where an agent must act on a model it has to construct and correct for itself. In embodied and robotic control, a manipulator can hold an executable model of contact dynamics and constraints, plan against it, and reinduce it the moment a grasp or a collision contradicts the prediction. In scientific experimentation and laboratory automation, the same loop makes a mechanistic hypothesis executable, designs the intervention that best discriminates between rival explanations, and preserves the evidence that promoted or retired each theory. In long-horizon software agents, the theory is a model of a large codebase together with its build and test behavior, revised whenever a change falsifies an assumed invariant. These domains bring continuous dynamics, noisy sensing, partial observability, and non-deterministic feedback that relax the clean assumptions ARC-AGI-3 offers, so each will need its own contracts, verifiers, and evidence. We will carry this externalized, auditable revision policy into one such environment, and ask whether a learned or otherwise optimized escalation policy can preserve that auditability while improving average cost and success.

For an executable world-model agent, the world is a hypothesis: acted through, tested against consequence, and revised under evidence that remains open to inspection.

References

ARC Prize Foundation. ARC-AGI-3: A new challenge for frontier agentic intelligence. *arXiv preprint arXiv:2603.24621*, 2026a.

ARC Prize Foundation. ARC-AGI-3 methodology. `https://docs.arcprize.org/methodology`, 2026b. Official scoring documentation.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by Monte Carlo Tree Search. *arXiv preprint arXiv:2405.15383*, 2024.

Dots Post-train Team. TELL. `https://github.com/rednote-hilab/TELL`, 2026. GitHub repository.

Alexis Fox. Read-grep-bash agent (RGB-Agent). `https://github.com/alexisfox7/RGB-Agent`, 2026. GitHub repository.

Liew Keong Han. Explore before you solve: The speed–depth trade-off in epistemic agents for ARC-AGI-3. *arXiv preprint arXiv:2605.25931*, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Siyuan Ma, Bo Gao, Xiaojun Jia, Simeng Qin, Tianlin Li, Ke Ma, Xiaoshuang Jia, Wenqi Ren, and Yang Liu. ODAR: Principled adaptive routing for LLM reasoning via active inference. *arXiv preprint arXiv:2602.23681*, 2026.

Javier Aguilar Martín. When a verified world model still loses: Play-adequacy vs prediction-accuracy in LLM-synthesized code world models. *arXiv preprint arXiv:2607.14169*, 2026.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for LLM agents. *arXiv preprint arXiv:2509.03581*, 2025.

Sergey Rodionov. Executable world models for ARC-AGI-3 in the era of coding agents. *arXiv preprint arXiv:2605.05138*, 2026a.

Sergey Rodionov. ARC-3 agents baseline: Secure baseline v1.5 results. `https://github.com/astroseger/arc-3-agents-baseline1/blob/main/results/README_secure_baseline_v1.5.md`, 2026b. GitHub repository.

Evgenii Rudakov, Jonathan Shock, and Benjamin Ultan Cowley. Graph-based exploration for ARC-AGI-3 interactive reasoning tasks. *arXiv preprint arXiv:2512.24156*, 2025.

Elad Sarafian, Gal Kaplun, Ron Banner, Daniel Soudry, and Boris Ginsburg. Workspace optimization: How to train your agent. *arXiv preprint arXiv:2605.09650*, 2026.

Hao Tang, Darren Key, and Kevin Ellis. WorldCoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. *arXiv preprint arXiv:2402.12275*, 2024.