WHAT ACCURATE PREDICTIONS HIDE: TRACING ACTION-EFFECT ERRORS IN WORLD MODELS VIA PAIRED ROLLOUTS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models support planning and policy learning by predicting how actions change future outcomes, yet accurate predictions of the usual outcome can conceal errors in action dependence. We use *paired rollouts* to compare how alternative action sequences change environment and model outcomes from a shared history. Because each generated step conditions the next, we use *differential transport* to decompose action-effect error exactly into local discrepancies propagated by later model dynamics, measuring their contributions through controlled transition substitutions. Across ten Craftax world models, cancellation and delay reveal outcomes predicted without the required action or before its execution. Substantial action-effect errors persist in about 13% of assessments passing per-branch return checks. Supervising next-step outcomes of alternative actions raises the rate of correctly timed event predictions under delay by over 40 percentage points relative to matched replay on average. Controlled substitutions show that using the updated generator for only the first transition suppresses later false events across three training repetitions of a shared pretrained model, while correct event generation changes little and remains rare. These findings make action dependence a concrete target for evaluation and training, with controlled substitutions revealing how local model updates change delayed consequences.

1 INTRODUCTION

World models support planning and policy learning by simulating the consequences of alternative actions (Sutton, 1990; Hafner et al., 2025; Hansen et al., 2024). This use depends on predicting how a choice changes future outcomes, often several steps later, as when one action supplies a prerequisite for a later step of a plan. A model that predicts success after a necessary action has been omitted can make an infeasible plan appear viable.

Which consequences a model must preserve depends on its use. When a decision compares only returns, accurate return prediction can suffice, and value-aware objectives train models for this purpose (Farahmand et al., 2017; Grimm et al., 2020). Policies trained in imagination also choose later actions from the model's generated states, so a spurious consequence can shape imagined experience even when its return is correct. Trained world models are typically validated by the rewards and observations they predict along recorded action sequences (Dedieu et al., 2025), whereas an action's consequence is a difference between the futures of alternative actions. Intervention benchmarks for video generation compare predictions across changed conditions and find errors that individual checks miss (Cai et al., 2026). For trained control models, the open question is how faithfully predictions follow changes in actions, and whether return and observation checks detect failures of that dependence.

Figure 1 makes the distinction concrete: two branches start from the same Craftax history and follow the same continuation after different first actions. Only one branch acquires a stone sword in the environment, yet the model generates it in both futures while closely predicting both branch returns. Its return comparison still favors the correct branch, but the imagined future of the other branch contains a sword that the environment never produces. This example motivates a direct test of whether changing an action changes the predicted consequence in the way it changes the environment.

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



Figure 1: **Accurate returns conceal a lost action effect. (a)** Two branches share the initial history ($t = 0$), differ only in their first action ($t = 1$), and execute 19 identical continuation actions to $t = 20$. In the environment, only Branch 0 crafts the stone sword ($d \neq 0$). The model generates it in both branches, erasing this action effect ($\hat{d} \approx 0$). White boxes mark the stone-sword inventory slot. **(b)** Predicted branch returns $(0.363, 0.001)$ closely match environment returns $(0.363, 0.000)$, despite the spurious tool in Branch 1. Return validation thus misses the lost action consequence. This diagnostic case was selected retrospectively; means use eight environment references and 32 paired model draws, and images show a representative pair.

We evaluate this dependence using *paired rollouts*: from a shared initial history, we compare environment and model outcomes under prescribed changes to the action sequence. Changing only the first action and sharing all subsequent actions isolates its delayed effect, while moving its execution time tests whether the predicted event follows the action rather than the first step after the history. We call the difference between expected branch outcomes the *action effect*, and compare model and environment effects. An incorrect endpoint effect does not show where the error arose: the model generates each step autoregressively from its own earlier outputs, so a discrepancy at one transition can be amplified, corrected, or carried to a later event. To trace these contributions along a specified order of transition substitutions, we use *differential transport*, an exact decomposition of effect error into local discrepancies propagated by later model dynamics. Controlled substitutions make each contribution measurable without estimating complete transition kernels.

Across ten independently trained Craftax world models, cancellation and delay expose failures hidden by accurate normal predictions, including consequences predicted without the required action and resource changes predicted before delayed execution. In an evaluation of 2,560 episodes, substantial action-effect errors persist in about 13% of assessments passing per-branch return checks, while localized consequence checks filter these errors more effectively than spatially averaged metrics. Supervising immediate action alternatives changes this dependence, raising the rate of correctly timed event predictions under delay by over 40 percentage points relative to matched replay on average. Transition substitutions then trace the learned update's downstream effect: replacing only the initial transition suppresses later false events across three independent training repetitions, while correct event generation remains rare. These models can *predict what happens while missing what changes*: paired evaluation exposes the mismatch, and controlled substitutions distinguish suppressing false consequences from recovering valid ones.

2 Related Work

Our study connects three research questions: which prediction errors matter for control, how to test action dependence, and how local errors propagate through a rollout.

Prediction for control. Dyna uses learned dynamics to supply experience for policy improvement (Sutton, 1990). Imagined experience now comes from recurrent latent (Hafner et al., 2021; 2025), transformer (Micheli et al., 2023; Dedieu et al., 2025), and diffusion (Alonso et al., 2024) world models, and latent models also support online planning (Hansen et al., 2024). Task-oriented objectives make the control relevance of prediction error explicit: value-aware model learning weights transition errors through a value function (Farahmand et al., 2017), while value equivalence compares Bellman updates over specified function and policy classes (Grimm et al., 2020). Joint model–policy learning and calibrated value-aware objectives also align training with model use (Eysenbach et al., 2022; Voelcker et al., 2025). Models that emit observations also expose consequences beyond the return.

Evaluating action dependence. For Transformer World Models (TWM), rollout checks measure symbol accuracy and reconstruction error, and inspect generated trajectories for violations of game dynamics (Dedieu et al., 2025). Such checks follow each trajectory's recorded action sequence. What-If World instead compares generated videos across paired physical interventions and finds incorrect relative outcomes even among pairs that pass every individual check, with timing interventions among the hardest (Cai et al., 2026). World-in-World evaluates generative world models through closed-loop planning, where task success tracks how well predictions follow intended actions more than visual quality (Zhang et al., 2026), and World Action Verifier uses forward–inverse consistency to test action reachability and improve world models (Liu et al., 2026). We test action execution and timing in trained control models against executable environment references, then assess whether return and observation checks detect the resulting effect errors.

Tracing rollout errors. Detecting an incorrect action effect leaves open how local discrepancies affect its later consequences. Simulation bounds relate transition error to accumulated value error (Lobel & Parr, 2024), and model exploitation can reverse policy preferences between a model and its environment (Bhamidipaty et al., 2026). Training methods address exposure to generated inputs through scheduled sampling (Bengio et al., 2015), self-correcting dynamics (Talvitie, 2017), or training video models on their own histories (Huang et al., 2025). Our temporal analysis measures how later model dynamics transform an error in the difference between two branches.

3 Evaluating and Tracing Action Effects

We measure how an action changes a chosen outcome in the environment and the model (Section 3.1), then trace errors in that comparison through the rollout (Section 3.2).

3.1 Paired Action Effects

To isolate what an initial action changes, we compare rollouts that start from the same state and history, differ only in that action, and share all later actions. For horizon H, a query $q = (x_0, h_0, a^0, a^1, \mathbf{u})$ contains the environment state x_0, the model-visible history h_0, two candidate actions, and shared continuation actions $\mathbf{u} = (u_1, \ldots, u_{H-1})$. In Figure 1, the candidates are Up and Left and $\mathbf{u}$ contains 19 actions. We restore x_0 for each environment branch and give the model only h_0 and its action sequence; branch b executes $(a^b, \mathbf{u})$. A readout ϕ of the resulting history selects the consequence to compare: a vector of observation features, the occurrence or onset time of an event, or a scalar return. For an integrable readout, define

$$\begin{aligned} d(q) &= \mathbb{E}_{\text{env}}[\phi \mid q, b = 1] - \mathbb{E}_{\text{env}}[\phi \mid q, b = 0], \\ \hat{d}(q) &= \mathbb{E}_{\text{model}}[\phi \mid q, b = 1] - \mathbb{E}_{\text{model}}[\phi \mid q, b = 0]. \end{aligned} \tag{1}$$

Here the hat denotes model prediction, and the *action-effect error* is $\hat{d} - d$. Taking a branch difference removes errors shared equally by both predictions and retains errors in what distinguishes the actions. The same definition applies to any two prescribed action sequences, such as execution and delay schedules; shared continuations are the special case used in Sections 4.3 and 4.5.

Connection to action values. The same comparison underlies action values when both branches continue with a common policy π, choosing later actions from their own histories. When both branches share the initial-state distribution given h and the readout is the return over a fixed horizon, the return effect is

$$m^\pi(h; a^1, a^0) = Q^\pi(h, a^1) - Q^\pi(h, a^0), \qquad A^\pi(h, a) = \mathbb{E}_{a' \sim \pi(\cdot|h)} m^\pi(h; a, a'). \tag{2}$$

Thus the advantage used in policy-gradient methods averages return effects relative to policy-sampled alternatives (Sutton et al., 1999). A return is one projection of the trajectory: accurately estimating this projection supports the corresponding action comparison, whereas evaluating another consequence requires its own readout. Our shared-action comparisons hold the continuation sequence constant to isolate the initial action; policy continuation measures a different effect that includes later feedback.

3.2 Differential Transport

The paired comparison shows whether the predicted effect is wrong at the endpoint, but not at which transition the error enters or how later steps transform it. We construct hybrid rollouts that execute the same action sequence with environment dynamics for the first $H - L$ transitions and model dynamics for the last L. Let $\hat{d}_L$ be their action effect, so $\hat{d}_0 = d$ and $\hat{d}_H = \hat{d}$. Increasing L by one moves the switch point $s = H - L$ one step earlier, replacing one more environment transition while retaining the later model dynamics. Its endpoint contribution is directly measurable:

$$C_L = \hat{d}_L - \hat{d}_{L-1}, \qquad L = 1, \dots, H. \tag{3}$$

These contributions sum to $\hat{d} - d$, but each mixes the discrepancy introduced at the replaced transition with the model dynamics that follow it. To separate the two, we write environment and model operations as transition kernels P_t and $\hat{P}_t$ on a common state (Appendix B). For the suffix starting at s, p_t^b is the environment branch law and $\hat{p}_t^b$ the law that switches to the model at s; their branch contrasts are $\Delta_t = p_t^1 - p_t^0$ and $\hat{\Delta}_t = \hat{p}_t^1 - \hat{p}_t^0$, with error $E_t = \hat{\Delta}_t - \Delta_t$. The discrepancy introduced at step t is $F_t = \Delta_t(\hat{P}_t - P_t)$, the one-step kernel difference weighted by the environment branch contrast, and readouts act on signed measures through $T_\phi(\nu) = \int \phi \, d\nu$.

Proposition 1 (Action-effect recursion)**.** *The hybrid and environment laws agree at the start of the suffix, so $E_s = 0$. For $t = s, \dots, H - 1$,*

$$E_{t+1} = E_t \hat{P}_t + F_t, \qquad \hat{d}_L - d = \sum_{t=s}^{H-1} T_\phi\Big(F_t \hat{P}_{t+1:H-1}\Big), \tag{4}$$

where $\hat{P}_{i:j} = \hat{P}_i \cdots \hat{P}_j$ and an empty product is the identity.

As in the telescoping argument behind simulation lemmas (Lobel & Parr, 2024), the recursion separates an existing error propagated by the next model step from the new discrepancy introduced at that step, here for a branch difference. Subtracting its expansions for adjacent suffixes gives

$$C_L = T_\phi\Big(F_s \hat{P}_{s+1:H-1}\Big), \qquad s = H - L. \tag{5}$$

Because F_s weights the kernel difference by the environment branch contrast, discrepancies shared equally by both branches cancel, and the later model dynamics can amplify or damp the remainder before it reaches the endpoint. This *differential transport* attributes endpoint changes to a specified sequence of substitutions, each replacing an entire transition; other orders can give different contributions (Appendices A–B). For return margins $m = J^1 - J^0$ and $\hat{m}_L = \hat{J}_L^1 - \hat{J}_L^0$, where hybrid returns use environment rewards in the prefix and model rewards in the suffix (Appendix C), the same construction decomposes $\hat{m}_H - m$; composing a common policy with each history transition gives the policy-continuation identity, connecting the decomposition to the action-value differences of Equation 2.

The same calculation compares two learned generators: hold the history and later generator constant, replace one transition, and measure the resulting endpoint difference. Section 4.5 uses this comparison to test whether a learned local change affects a later event.

4 EXPERIMENTS

We use paired rollouts to test whether predictions follow changes in action execution and timing (Section 4.2), then examine which errors return and observation checks miss (Section 4.3). Matched training tests whether supervising action alternatives improves this dependence (Section 4.4). We then use transition substitutions to measure how the learned change affects later predictions and continue actor training to test for policy gains (Section 4.5).

4.1 EXPERIMENTAL SETUP

We study ten independently trained Transformer World Models (TWM; Systems 0–9) in Craftax-Classic (Dedieu et al., 2025; Matthews et al., 2024). Restoring simulator states lets us compare predictions with the consequences of the same actions for events with known prerequisites: iron acquisition for the action schedules and training, first appearance of crafted tools for validation, and iron-pickaxe crafting after acquisition for the downstream tests. The action tests use common held-out states for all ten systems, selected by environment prerequisites, while 2,560 separate episodes support return and observation validation; Appendix C lists these and the training and downstream sets with their budgets.

We apply the same event definition to environment and model outcomes, retaining failed attempts and terminated branches. Generated tokens that decode to several possible counts or materials produce lower and upper event estimates, called *semantic bounds*. Percentage ranges written a–b denote these bounds unless stated otherwise, while episode resampling measures sampling uncertainty.

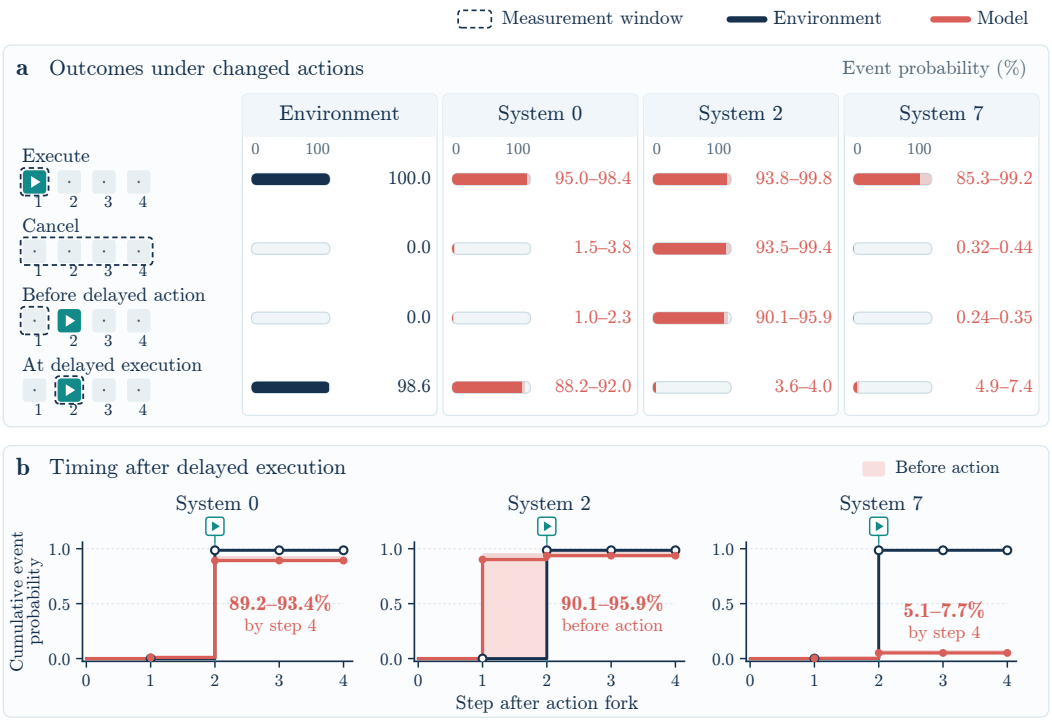


Figure 2: **Cancellation and delay reveal different failures.** Three original systems illustrate the ten-system comparison in Table 2. **(a)** Triangles denote execution, dots no-ops, and dashed boxes the measurement window. The joint event requires inventory increase and designated-resource depletion on the same transition. Rows measure normal onset, cancellation through step 4, premature onset, and first onset at delayed execution. **(b)** Cumulative probabilities under delay; pink shading marks occurrence before execution. Solid bars/curves are lower semantic bounds; lighter extensions/bands reach the upper bounds. Each system uses 64 selected states (63 valid for System 2), 64 environment and 128 model draws per schedule.

4.2 Action Dependence

Cancellation tests whether a predicted event requires its action, and delay tests whether the event follows the action's timing. From the same 20-frame observed history, we execute a resource-acquisition action (mining iron) at step 1, delay it to step 2, or omit it, using the environment's no-operation action (no-op) at all other steps in the four-step window. Acquisition requires both an inventory increase and depletion of the designated resource block in the same transition. All ten systems receive the same 64 selected acquisition states, with 61–64 valid states per system after environment readout checks. Figure 2 highlights three contrasting responses within the complete comparison in Table 2.

High event probabilities under normal execution conceal contrasting responses to changed actions: the models predict acquisition at the execution step with probabilities ranging from 76.4% to 99.9%, allowing for semantic uncertainty. Yet five of the ten systems also predict it in more than 40% of cancelled rollouts. System 2, for example, predicts it with 93.8–99.8% probability under execution, but also with 93.5–99.4% probability after cancellation. Delaying execution to step 2 still leaves the event at step 1 in 90.1–95.9% of draws. The failure therefore begins immediately after an observed history, before a long generated prefix could accumulate errors.

System 7 shows a different failure. It predicts almost no inventory increase without execution, yet it predicts resource depletion in 85.9–99.8% of cancelled rollouts and before the delayed action in 81.2–94.5% of delayed rollouts. At the delayed step, its joint acquisition probability falls to 4.9–7.4%. System 0 instead retains a delayed-onset probability of 88.2–92.0%, yet it predicts inventory increases in nearly half of cancelled rollouts while its joint event stays at 1.5–3.8% (Appendix D): an accurate joint-event probability can coexist with errors in the physical changes that compose it. Accurate execution predictions can therefore conceal distinct errors in whether a consequence depends on the action and when its physical components occur.

4.3 Validation of Action Consequences

To measure what return and observation checks miss, we switch the readout from acquisition to tool consequences and evaluate 2,560 separate held-out episodes from ten systems. In each episode, two branches start from one state with the actor's two most probable actions and then share 19 recorded actions; each initially absent tool with a crafting action in that continuation forms one assessment. A tool first appearing at step t scores γ^{t-1} ($\gamma = 0.925$), alongside the task return (Appendix G).

Coarse filtering by return accuracy. We test what return filtering misses by retaining assessments with estimated per-branch return error at most 0.05 and measuring tool-effect errors of at least 0.10 on independent samples. With equal weights for systems and episodes, the return criterion retains 37.0% of assessments. Substantial tool-effect errors remain in 13.0–13.3% of retained assessments overall, ranging from 3.5% to 46.5% across systems (Table 9). These errors concern both when a tool appears and whether it appears at all. At ten candidate states fixed in advance from an earlier case list, one per system, deeper independent sampling establishes accurate expected returns with incorrect expected tool effects at five of them (Appendix G.3). The episode sample measures what finite-sample return validation misses; the selected-state tests confirm that the discrepancy persists beyond sampling noise there.

Refining validation with localized observation checks. We then test whether localizing observation checks detects these remaining errors. On the same return-filtered assessments, we rank predictions by four token-distribution errors: whole-frame mean, heads-up-display (HUD) mean, HUD temporal maximum, and target-tool-field temporal maximum. The last check reads only the assessed tool's inventory token; independent samples again measure the remaining event error. Keeping the half with the lowest observation error (50% retention), whole-frame checking leaves a 9.34–9.65% event-error rate, compared with 0.42% for the target field (Figure 3a). The HUD checks stay close to the whole-frame check, and taking a temporal maximum does not consistently outperform averaging. Resolving the relevant field matters more here than removing time averaging alone.

Stricter checking could achieve low error simply by discarding states where actions have meaningful consequences. Figure 3b therefore measures the fraction of absolute environment-effect magnitude retained. At the same 50% retention, whole-frame and target-field checks keep 37.3% and 35.2% of

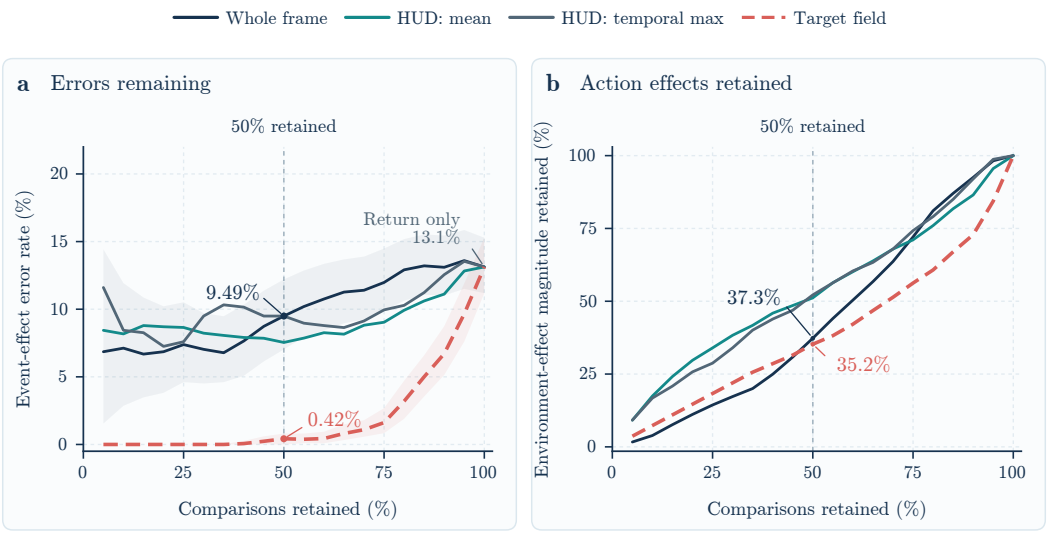


Figure 3: **Targeted observation checks detect errors that aggregate checks leave behind.** After requiring estimated return error ≤ 0.05 in each branch, we retain the lowest-observation-error assessments. **(a)** Fraction of retained assessments with independent tool-effect error ≥ 0.10. **(b)** Fraction of the return-filtered set's weighted absolute environment-effect magnitude retained. The shared horizontal axis measures retention within the return-filtered set (37.0% of the original assessments). Filtering and error measurement use independent samples. Curves use semantic midpoints; shading combines semantic bounds with pointwise 95% episode-bootstrap intervals for whole-frame and target-field checks. Dots mark 50% retention. The target-field check is event-aware (Appendix H).

that magnitude, respectively. The target-field check therefore retains more reliable predictions at a comparable environment-effect magnitude, while tighter filtering discards more of that magnitude.

4.4 Learning from Action Alternatives

Ordinary training supervises only the action that was taken. To test whether supervising what changes under alternative actions improves action dependence beyond additional training, we compare two one-step updates at the same histories: *replay* repeats the observed-action successor, whereas *action coverage* supplies successors for execution and no-op. Both use the same number of targets and updates, supervise only the next observation and termination, and leave the actor and tokenizer unchanged; all ten systems train on one common 64-state set, and Systems 0–2 repeat training on two further sets with independent optimization randomness. On held-out states disjoint from training, we then ask whether the trained event's timing transfers to the untrained delayed sequence, whether two untrained events improve under the same schedules (stone-pickaxe crafting and workbench placement), and whether normal execution is preserved (Appendix E).

Action coverage increases the probability of a correctly timed event after delay by 41.4–56.6 percentage points relative to replay, averaging the ten systems equally, with a 95% episode-bootstrap envelope of $[35.5, 59.5]$ points. The delayed schedule begins with the no-op contrast used in training, and premature and cancelled events each fall by about 40 points. Because correct delayed onset requires producing the event at step 2, its rise shows that the updated models produce the event when execution arrives one step later rather than only suppressing it. The mean timing error, combining normal and delayed onset, falls by 15.1–31.4 points relative to replay and by 13.7–30.0 points relative to the original models (Figure 4), with reliable improvements over replay in seven of ten systems (Table 4). Comparing against the original models also accounts for timing errors introduced by replay itself. Immediate action alternatives thus change predictions one step beyond the supervised transition.

The acquisition timing gain varies across systems: System 2 improves in all three training repetitions, whereas System 0's joint-event timing changes little. Transfer to other events is weaker: neither

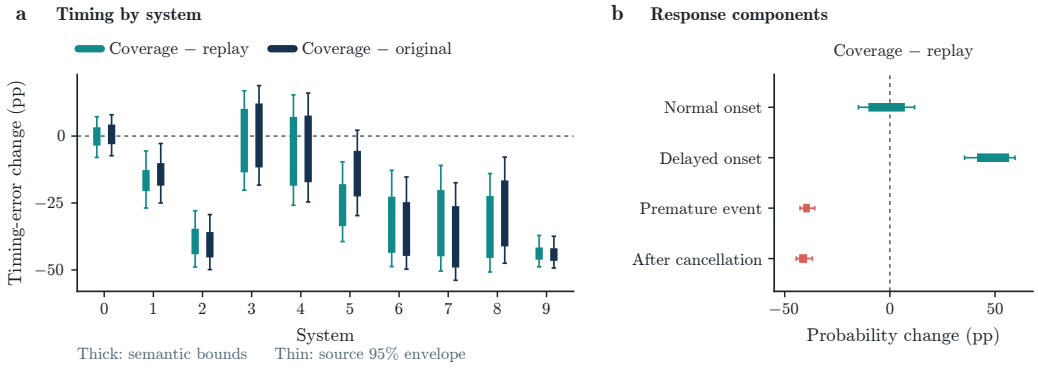


Figure 4: **Immediate action alternatives improve delayed acquisition timing. (a)** Coverage minus replay and original: normal/delayed-onset absolute error, averaged within each of ten systems after training on set 0. Negative values indicate improvement. **(b)** Coverage minus replay averaged across those systems: normal onset at step 1, delayed onset at step 2, premature occurrence under delay, and occurrence by step 4 after cancellation. Thick marks show semantic mean bounds; thin marks give 95% episode-bootstrap envelopes. Each system evaluates 61–64 valid states with 64 environment and 128 model draws per condition (Appendix E).

crafting nor placement shows a reliable average timing gain, and crafting timing is reliably worse than the original model in two system–set cases (Table 5). No system meets the five-percentage-point preservation criterion for normal acquisition against both references.

Unlike delayed timing, aggregate metrics show no consistent advantage: on 128 test states, average observation-likelihood and token-error differences from replay are small and uncertain, while relative to the original model, observation likelihood worsens slightly and token mismatch improves (Appendix E). Action alternatives improve a dependency that aggregate prediction quality would not reveal.

4.5 Downstream Prediction and Policy Learning

The coverage update supervises one transition, whereas planning and imagination use multi-step predictions. The substitutions of Section 3.2 measure how one transition affects a later consequence under a fixed continuation. We illustrate this comparison with an environment transition before applying it to the learned update.

In an illustrative case selected from an earlier diagnostic set, two branches begin by mining stone or placing a plant and later attempt to craft a stone pickaxe at step 8. The environment produces the pickaxe only after mining, whereas the model generates it in both branches across all 32 draws. Replacing the whole first transition with the environment's, including its stone-inventory difference, changes what the remaining 19 model steps produce: the tool appears in 32 of 32 versus 0 of 32 draws, and the final return margin moves from -0.009 to -0.558, close to the environment's -0.579 (Figure 5a). Appendix I gives a second, later substitution and measurement details.

We apply the same comparison to the replay and coverage models to test whether a learned transition change affects an event beyond the training target. The initial action mines iron, and the route ends 4–20 steps later by crafting an iron pickaxe that consumes the acquired iron; 128 new episodes provide matched execution and cancellation branches. For Systems 0–2, carried forward from an earlier diagnostic study, each with three training repetitions, we cross the first-transition and continuation generators, each either replay (R) or action coverage (C). Thus CR uses coverage for the first step and replay thereafter (Appendix F).

Only System 2 generates joint crafting events in these sampled rollouts. Across its three training repetitions, changing only the first generator reduces false crafting after cancellation from 2.36–3.24% under RR to 0.012–0.062% under CR (ranges span the repetitions), while normal-branch crafting changes by less than 0.06 percentage points (Figure 5b). This first-step change reliably lowers squared error in the crafting action effect by 2–10% of the RR risk across the three sets' 95% envelopes,

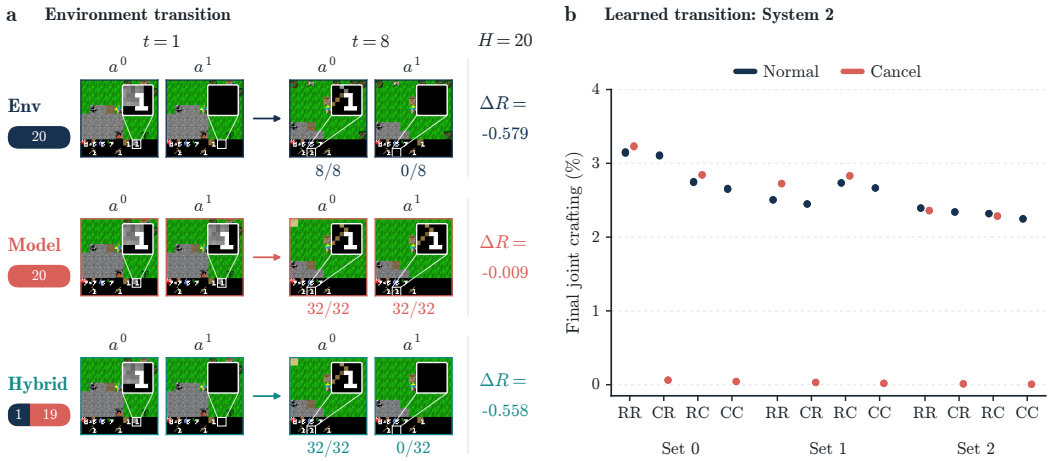


Figure 5: **A local change can alter a delayed consequence. (a)** Retrospective illustrative case: mining stone (a^0) versus placing a plant (a^1), then 19 shared actions. Hybrid replaces the first model transition with the environment. Insets mark stone inventory and the pickaxe (8 environment/32 model draws); $\Delta R = J^1 - J^0$. **(b)** R/C denote replay/coverage; the two letters specify the first and later generators. Joint crafting requires a tool increase and iron expenditure on the final action. Marks show semantic mean bounds (126 states per set); Table 7 reports uncertainty in paired risk differences. Environment crafting succeeds in 82.5% of normal and 0% of cancelled rollouts; Systems 0 and 1 predict no crafting on any path (Table 8).

whereas changing only the continuation does not (Table 7). The gain thus comes from suppressing a false consequence under the same continuation. Correct crafting remains rare: normal-branch success is about 2–3%, against 82.5% in the environment.

From prediction to policy learning. To test whether these predictive gains improve policy learning, we continue training each pretrained actor in imagination from stored episodes, without new environment interaction, using the original, replay, or coverage model. All 63 continued actors (three systems, seven world models, three seeds each) finish below their pretrained baselines, including those using the unchanged world model (Figure 7 in Appendix J). The relative performance of coverage and replay varies across systems, with neither producing an absolute policy gain under this procedure.

5 Discussion and Conclusion

World models can predict the usual outcome accurately while misrepresenting its dependence on an action. Paired comparisons expose errors that return filtering and spatially averaged observation checks miss, while transition substitutions trace how local changes affect later outcomes.

Supervising immediate action alternatives improves the trained event's timing one step after the supervised transition, beyond matched replay on average. Downstream substitutions identify a concrete route to lower effect error: a change to the first transition suppresses later false events under the same continuation. Correct events remain rare, so the lower action-effect error reflects suppression of invalid outcomes rather than recovery of valid ones. Under the tested actor-continuation procedure, actors trained with original and updated world models all underperform their pretrained baselines.

Scope and next steps. We test ten models of one architecture in one environment, with repeated training for three models. Intervals measure variation across evaluation states. Semantic bounds capture readout ambiguity but can miss definite decoding errors. Further tests should examine normal-behavior preservation and transfer across model families. The broader lesson is to evaluate and train what actions change, alongside the accuracy of individual predictions.

Ethics Statement

This study evaluates learned world models in the Craftax simulator and involves no human participants or personal data. Its intended benefit is more reliable evaluation of action consequences before model predictions inform decisions. Applications involving people or physical systems require validation of their own consequences and failure modes beyond the simulated tasks studied here.

Reproducibility Statement

Section 3 defines the paired comparisons and transition substitutions, with the proof and operational state in Appendices A–B. Appendix C specifies model checkpoints, evaluation sets, sampling, and estimators. Appendices D–H document action schedules, training, downstream comparisons, semantic readouts, and observation validation; Appendix J details actor continuation. Numerical data underlying the figures and tables, uncertainty estimates, and full-precision readout parameters are provided in the supplementary data.

AI Use Statement

We used OpenAI ChatGPT and Codex to assist with language editing, grammatical polishing, preliminary literature discovery, and auxiliary script formatting for data plotting. All theoretical derivations, empirical implementations, and scientific analyses were designed, conducted, and verified entirely by the authors, who take full responsibility for all code, results, and claims in this manuscript.

References

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Logan Mondal Bhamidipaty, Esmeralda S. Whitammer, David Abel, Mykel J. Kochenderfer, and Subramanian Ramamoorthy. Imperfect world models are exploitable. *arXiv preprint arXiv:2605.15960*, 2026. doi: 10.48550/arXiv.2605.15960.

Lars Buesing, Theophane Weber, Yori Zwols, Sebastien Racaniere, Arthur Guez, Jean-Baptiste Lespiau, and Nicolas Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019.

Kunlin Cai, Rui Song, Jinghuai Zhang, Kaiyuan Zhang, Pranav Bodapati, Alicia Yu, Fnu Suya, Mohammad Rostami, Jiaqi Ma, and Yuan Tian. What-if world: A causal benchmark for general world models in embodied scenarios. *arXiv preprint arXiv:2605.27589*, 2026.

Antoine Dedieu, Joseph Ortiz, Xinghua Lou, Carter Wendelken, J Swaroop Guntupalli, Wolfgang Lehrach, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Improving transformer world models for data-efficient RL. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12944–12968, 2025.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based RL. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23230–23243, 2022.

Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1486–1494, 2017.

Christopher Grimm, Andre Barreto, Satinder P. Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5541–5552, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yuejiang Liu, Fan Feng, Lingjing Kong, Weifeng Lu, Jinzhou Tang, Kun Zhang, Kevin Murphy, Chelsea Finn, and Yilun Du. World action verifier: Self-improving world models via forward-inverse asymmetry. *arXiv preprint arXiv:2604.01985*, 2026.

Sam Lobel and Ronald Parr. An optimal tightness bound for the simulation lemma. *Reinforcement Learning Journal*, 2:785–797, 2024.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35104–35137, 2024.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pp. 216–224. Morgan Kaufmann, 1990. doi: 10.1016/B978-1-55860-141-3.50030-4.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999. URL https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, pp. 2597–2603, 2017. doi: 10.1609/aaai.v31i1.10850.

Claas A. Voelcker, Anastasiia Pedan, Arash Ahmadian, Romina Abachi, Igor Gilitschenski, and Amir-Massoud Farahmand. Calibrated value-aware model learning with probabilistic environment models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 61745–61768, 2025.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal M. Patel, Paul Pu Liang, Daniel Khashabi, Cheng Peng, Rama Chellappa, Tianmin Shu, Alan Yuille, Yilun Du, and Jieneng Chen. World-in-world: World models in a closed-loop world. In *International Conference on Learning Representations*, 2026.

A Proof of Proposition 1

The signed-measure expansion corresponding to the main-text readout is

$$E_H = \sum_{t=s}^{H-1} F_t \hat{P}_{t+1:H-1}. \tag{6}$$

Proof. For a finite signed measure ν and a Markov kernel K, write $(\nu K)(A) = \int K(x, A)\, \nu(dx)$. Kernel action is linear in ν. Therefore

$$\Delta_{t+1} = \Delta_t P_t, \qquad \hat{\Delta}_{t+1} = \hat{\Delta}_t \hat{P}_t,$$

and

$$\begin{aligned} E_{t+1} &= \hat{\Delta}_t \hat{P}_t - \Delta_t P_t \\ &= (\hat{\Delta}_t - \Delta_t)\hat{P}_t + \Delta_t(\hat{P}_t - P_t) \\ &= E_t \hat{P}_t + F_t. \end{aligned}$$

We prove by induction that, for every $n \in \{s, \dots, H\}$,

$$E_n = \sum_{t=s}^{n-1} F_t \hat{P}_{t+1:n-1}.$$

At $n = s$, the sum is empty, and $E_s = 0$ because both hybrid branch laws coincide with the environment laws through time s. If the identity holds at $n < H$, Equation 4 yields

$$\begin{aligned} E_{n+1} &= E_n \hat{P}_n + F_n \\ &= \sum_{t=s}^{n-1} F_t \hat{P}_{t+1:n} + F_n \hat{P}_{n+1:n} \\ &= \sum_{t=s}^{n} F_t \hat{P}_{t+1:n}, \end{aligned}$$

where $\hat{P}_{n+1:n}$ is the identity. Taking $n = H$ proves Equation 6. Because the sum is finite and ϕ is integrable under its component measures, linearity of the vector integral gives

$$T_\phi(E_H) = \sum_{t=s}^{H-1} T_\phi\Big(F_t \hat{P}_{t+1:H-1}\Big),$$

which is Equation 4. □

Appendix B derives the adjacent-suffix identity in Equation 3 directly from the common environment prefix and the subsequent model kernels.

B Common Operational State

A transition substitution requires environment and model operations on the same measurable space. At time t, their common state records the simulator state x_t, the observation–action history, the internal model state c_t, the most recently emitted observation and reward, and the prescribed action script. The history also retains the emitted trajectory for path-dependent readouts. The two branches differ in the first action recorded in that script. Afterward the scripts contain the same actions. Time can be included in the state; we write time-indexed kernels for clarity.

The environment operation P_t executes the script's current action in the simulator, emits its observation and reward, and updates the model state with that observation. For TWM, this update appends the observation tokens to the history. The model operation $\hat{P}_t$ uses the available history and internal state to generate the next output and update that state. It does not consume the simulator component. That unused component may be retained unchanged during model generation, since each hybrid rollout

switches from environment to model dynamics only once. Both operations append their emitted outputs to the same trajectory history. The endpoint readout selects these outputs; the environment reference is the native observation.

For a query and branch, let p_0^b be the law after the model has processed the common initial history and the branch's action sequence has been recorded. Define $p_t^b = p_0^b P_{0:t-1}$ and

$$\hat{p}_H^{b,L} = p_0^b P_{0:H-L-1} \hat{P}_{H-L:H-1}.$$

An empty product is the identity. Adjacent suffixes share the environment prefix through $s = H - L$, including its model state, and the model kernels after s. Subtracting their endpoint laws gives

$$\hat{p}_H^{b,L} - \hat{p}_H^{b,L-1} = p_s^b (\hat{P}_s - P_s) \hat{P}_{s+1:H-1}.$$

Subtracting branches and applying T_ϕ yields Equation 3. The local difference therefore compares the whole environment operation, including its history update, with the whole model operation on this common space. This attributes error along the specified sequence of substitutions; other paths can give different contributions.

A return readout selects rewards from this emitted history, using environment rewards in an environment prefix and predicted rewards in a generated suffix. For substitutions between two learned generators, the common state instead retains the emitted history and the internal states required by both generators. Each generator reconstructs its own conditioning state from the shared emitted history. Replacing the environment kernel by one learned generator and the model kernel by the other gives the same adjacent-substitution identity.

C Models, Evaluation Sets, and Estimation

Models and comparisons. We use ten independently trained TWM systems at checkpoint 1,004,544 with the published Craftax-Classic M5 block-teacher-forcing configuration, tokenizer, and actor (Dedieu et al., 2025; Matthews et al., 2024). System numbers identify the ten independent training seeds. We restore each environment state to sample repeated native rollouts, while models receive only the observation–action history. The action-schedule and learning comparisons use common native histories collected by all ten actors, encoded by each system's own tokenizer. The return-validation set instead samples 256 episodes per system's actor and weights the ten systems equally. Table 1 summarizes the evaluation sets, horizons, and sampling budgets.

Table 1: **Evaluation sets, task horizons, and rollout sampling budgets.** State counts denote selected episodes, with one state per episode in each set. Environment/model columns give trajectory draws per state and condition. Comparisons across model versions share evaluation states, which form the resampling unit in state-bootstrap analyses. Each experiment reports its readout-validation criteria and valid-state counts.

Comparison	States and systems	Horizon	Env./model
Action schedules and updates	64 shared states per event; 10 systems, 3 events	4	64/128
Ordinary prediction	128 shared states; 42 model versions	4	32/32
Return/observation validation	256 states per system; 10 systems	20	64/128
Selected-state deep sampling	10 states; one per system	20	4096/8192
Delayed dependency	128 shared states; 3 systems, 3 training sets	4–20	64/128
Illustrative substitutions	Selected action pairs	20	8/32
Actor evaluation	1024 shared episodes per system; 63 continuations	Full episode	1/—

Sampling and uncertainty. Independent episode streams separate training, development, and held-out evaluation. Shared environment random streams reduce variance across action conditions (Buesing et al., 2019). Each evaluation records every selected state, including its termination status and any readout-validation failure. The action-schedule analysis weights the ten producing actors

equally and states equally within each producer. The downstream dependency analysis weights states equally. Paired state-bootstrap intervals quantify variation across evaluation states conditional on the trained checkpoints and saved rollout pools, with all conditions and model versions resampled together.

Squared effect error. Squaring a sample-mean error includes its sampling variance. For the downstream comparison, independent halves estimate the environment effect as d_A, d_B and a model effect as $\hat{d}_A, \hat{d}_B$. With $e_A = \hat{d}_A - d_A$ and $e_B = \hat{d}_B - d_B$, conditional independence gives

$$\mathbb{E}[e_A e_B \mid q] = \mathbb{E}[e_A \mid q]\mathbb{E}[e_B \mid q] = (\hat{d}(q) - d(q))^2.$$

We average this product over states and subtract paired model risks to assess improvement. Individual finite-sample products can be negative and remain in the estimate. For bounded semantic readouts, shared terms are cancelled before evaluating the product at the remaining interval endpoints. Appendix H gives the related categorical estimator used for observation validation.

Returns and termination. The environment return is the $\gamma = 0.925$ discounted sum of the binary reward targets used by TWM, $R_t^b = \mathbf{1}\{r_t^b \geq 0.5\}$. Model returns sum reward-head positive-class probabilities $\hat{R}_t^b$ with the same discount. For a hybrid beginning model generation at s, the return is

$$\hat{J}_L^b = \mathbb{E}\left[\sum_{t=0}^{s-1} \gamma^t R_t^b + \sum_{t=s}^{H-1} \gamma^t \hat{R}_t^b\right], \qquad s = H - L.$$

Displayed step 1 has discount exponent zero. The joint action and downstream tests evaluate events on the terminal transition, then absorb termination. The first-visible-tool analysis excludes terminal observations from event detection (Appendix G); both protocols retain terminated trajectories. Return margins subtract branch 0 from branch 1. Actor evaluation uses native undiscounted task return, a separate quantity described in Appendix J.

D Action Execution and Timing

Common state distribution. We collect independent Craftax-Classic episodes using all ten pretrained actors and retain the first state satisfying each event's environment prerequisites with a complete 20-frame history. Resource acquisition requires a stone pickaxe, iron ore immediately ahead, and iron inventory below its cap. The other tests require the native prerequisites for crafting a stone pickaxe or placing a workbench. Each event has 64 held-out states, allocated across the ten producing actors; every receiving model evaluates these same native histories. We select states solely from environment prerequisites. These 192 states are distinct from the return-validation episodes and the training sets.

Schedules and event definitions. The four-step sequences are $(\text{execute}, N, N, N)$, $(N, \text{execute}, N, N)$, and (N, N, N, N), where N denotes no-op and execution is the event's native action. Every condition starts from the same complete observed history and then retains its generated history. We use 64 environment and 128 model draws per condition. For acquisition, V_t indicates an iron-inventory increase and D_t depletion of the designated ore block on transition t. The joint event $V_t D_t$ requires both changes on the same transition. Stone-pickaxe crafting requires a tool-inventory increase and decreases of exactly one wood and one stone on the same transition. Workbench placement requires a decrease of exactly two wood and appearance of the workbench at the designated location. We retain the individual inventory and world-change components as well as their conjunction. Environment labels come directly from simulator states; model labels come from emitted observations.

Semantic measurement. Inventory token categories map to compatible counts, and renderer-derived terrain categories map to material labels. Ambiguous categories retain all compatible values. We track possible camera displacements using consecutive decoded world patches, fitting an RGB gain and bias on background cells outside the avatar and interaction region. The match uses at least 24 overlapping cells, patch-error clipping at 0.01, and two robust reweighting steps. Thresholds are calibrated separately for each tokenizer on development data and remain unchanged for evaluation.

We propagate all compatible positions; off-screen targets, occlusion, missing correspondence, or more than 4,096 candidate paths produce an unknown label. For placement, we measure the target only while every prefix displacement is uniquely supported as zero; any other correspondence makes subsequent target mappings unknown. This stricter rule contributes to the wider placement bounds. Alignment uses the observed prefix and generated frames. Table 3 gives the per-tokenizer matching parameters.

Taking the minimum and maximum over compatible labels bounds ambiguity in the observation readout. Calibration also reveals definite terrain errors: for example, one system mislabels 21 of 171,360 clear iron cells and 208 of 59,432 occluded iron cells. We therefore validate the readout on each selected state's environment trajectories before interpreting model outputs. Four system–state pairs fail this check in the held-out acquisition test: one for System 2 and three for System 9. The analysis retains their failure status without replacement and uses 63 and 61 valid states, respectively; the remaining systems and the other two events use all 64. Across repeated model versions this accounts for 16 invalid evaluations out of 8,064 attempted state–version–event evaluations.

Timing and effect estimates. Let T be the first joint-event step. We measure normal onset $P(T = 1)$, delayed onset $P(T = 2)$, premature occurrence $P(T \leq 1)$ under delay, and cancellation occurrence $P(T \leq 4)$. A definite onset requires a definite event at that step and definite absence earlier; a possible onset requires a possible event and no definite earlier event. For a model probability interval $[l, u]$ and native probability p, absolute-error bounds are

$$[\max(l - p, p - u, 0),\ \max(|l - p|, |u - p|)].$$

The timing loss averages the normal- and delayed-onset absolute errors. Execution and delay effects subtract the corresponding branch probabilities as in Equation 1. Table 2 reports all ten original systems on the common acquisition states.

We weight producing actors equally, then valid states equally within each producer. Ten thousand bootstrap samples resample the originally selected number of episodes within each producer; all model versions share the resampling counts. A resample missing every valid state from a producer is omitted, leaving 9,999 usable resamples for some acquisition comparisons. The 95% envelope runs from the 2.5th percentile of the lower semantic endpoint to the 97.5th percentile of the upper endpoint. Table 2 summarizes mean semantic bounds. Training comparisons report paired state-bootstrap envelopes.

Table 2: **Action dependence on acquisition states across all ten systems.** Cells show semantic mean bounds (%). Normal and at-delay columns measure first onset at steps 1 and 2; before-delay measures occurrence at step 1; cancel measures occurrence by step 4. n is the valid count out of 64 selected states. Environment onset is 100% for normal and 98.6% for delayed execution, and zero otherwise.

System	n	Normal	Cancel	Before delay	At delay
0	64	[95.0, 98.4]	[1.5, 3.8]	[1.0, 2.3]	[88.2, 92.0]
1	64	[90.5, 95.0]	[43.6, 45.0]	[43.6, 44.9]	[44.6, 49.0]
2	63	[93.8, 99.8]	[93.5, 99.4]	[90.1, 95.9]	[3.6, 4.0]
3	64	[89.7, 99.9]	[1.6, 2.1]	[1.4, 1.6]	[83.0, 97.1]
4	64	[85.6, 97.1]	[5.1, 10.3]	[4.2, 7.1]	[66.1, 78.7]
5	64	[76.4, 85.0]	[0.0, 0.0]	[0.0, 0.0]	[69.1, 77.1]
6	64	[87.4, 99.5]	[82.7, 93.6]	[73.3, 83.1]	[12.8, 18.4]
7	64	[85.3, 99.2]	[0.3, 0.4]	[0.2, 0.3]	[4.9, 7.4]
8	64	[85.3, 98.9]	[61.1, 75.2]	[60.0, 71.8]	[18.5, 25.0]
9	61	[95.6, 99.0]	[94.8, 98.6]	[94.7, 98.5]	[0.4, 0.4]

Low joint error can conceal a component error. For example, System 0 predicts an inventory increase in 47.8% of cancelled rollouts but joint acquisition in only 1.5–3.8%. System 7 also shows a mismatch between components: it predicts premature target depletion under delay in 81.2–94.5% of rollouts but premature joint acquisition in only 0.2–0.3%. It predicts target depletion in 85.9–99.8% of cancelled rollouts while predicting an inventory increase in fewer than 0.5%. The joint event definition tests

Table 3: **Per-tokenizer correspondence parameters.** RGB gain and bias intervals, maximum matching error, and tolerance above the best displacement. All parameters are set from development data; the same values apply to original, replay, and coverage models within a system.

System	Gain interval	Bias interval	Error threshold	Candidate tolerance
0	$[0.363524, 1]$	$[0, 0.137099]$	0.00333724	0.00053475
1	$[0.432192, 1]$	$[0, 0.12136]$	0.00358412	0.00053814
2	$[0.775899, 1.02236]$	$[-0.00439602, 0.0521976]$	0.00172166	0.00000010
3	$[0.797543, 1]$	$[0, 0.0577611]$	0.00195473	0.00000010
4	$[0.844534, 1.00059]$	$[0, 0.0501599]$	0.00113401	0.00000010
5	$[0.868761, 1]$	$[-0.0026364, 0.0266669]$	0.00155583	0.00000010
6	$[0.847816, 1]$	$[-0.00227281, 0.0299596]$	0.00151280	0.00000010
7	$[0.820546, 1.00012]$	$[-0.00215368, 0.0687728]$	0.00136819	0.00000010
8	$[0.83516, 1.00021]$	$[-0.00439262, 0.0424327]$	0.00102304	0.00000010
9	$[0.782014, 1.00014]$	$[0, 0.0815297]$	0.00384800	0.00000010

whether the physical changes occur together, while the component measurements identify errors masked by the joint probability.

E Learning from Immediate Action Alternatives

Matched supervision. We compare two updates starting from each pretrained checkpoint. Replay uses two copies of the successor for the producer's recorded action; action coverage uses one execution successor and one no-op successor at the same history and environment draw. Thus the arms match histories, target counts, batch order, and optimization budget while changing which immediate actions receive supervision. In three histories in each of training sets 0 and 2, the producer recorded an action other than execution. Each training set contains 64 acquisition opportunities, separate from the held-out episodes. Set 0 trains all ten systems; sets 1 and 2 repeat the comparison for Systems 0–2 with new episodes and optimization random streams. These three systems were carried forward from diagnostic development before collecting the new test rollouts. Repetitions vary training data and optimization randomness within each pretrained system.

We run 750 Adam updates with batch size 16, learning rate 3×10^{-5}, 32-update warmup, and gradient clipping at 32. The loss supervises the next observation's 81 tokens and termination. We leave the tokenizer, actor, critic, and reward-head parameters unchanged, although updating the shared world model can still change predicted rewards. Neither delayed sequences nor downstream crafting outcomes are training targets. The comparison produces 32 updated models; together with the ten original models, 42 versions undergo evaluation.

Generalization and preservation. We evaluate every version on acquisition, crafting, and placement using the common states and schedules in Appendix D. The ten-system aggregate averages the set-0 contrasts with equal weight per system. Tables 4 and 5 combine all training sets while retaining both replay and original-model references. For normal execution, preservation within five percentage points requires the upper 95% bound on the coverage-minus-reference *absolute-error difference* to be at most 0.05 against both references. Failure to satisfy this criterion does not by itself establish a five-point degradation. No acquisition comparison establishes this bound; crafting does so for Systems 0–2 and placement for System 4 in training set 0.

Ordinary prediction. A separate set of 128 episodes is collected without filtering for an event opportunity. All 42 versions predict the recorded four-action continuation, with 32 environment and 32 model draws per state. We measure observation negative log-likelihood and reward/termination likelihoods and classification errors on the first native successor, conditioned on the observed history; classification error compares the argmax prediction with the native label. Generated-token mismatch instead compares all four rollout steps. Likelihood comparisons are made within each tokenizer before averaging system contrasts. Table 6 reports the ten-system set-0 means. These ordinary prediction metrics show no corresponding average improvement over replay: observation-likelihood and token-mismatch differences remain uncertain, while their differences from the original model point in opposite directions. Together, these comparisons show improved timing for the trained event

Table 4: **Acquisition timing and normal-execution preservation.** C, R, and O denote action coverage, replay, and the original model. Cells show 95% paired state-bootstrap envelopes for coverage-minus-reference error differences in percentage points. Normal execution is preserved within a five-point tolerance when both upper bounds are at most $+5$. Set 0 covers all ten systems, with sets 1–2 repeating training for Systems 0–2.

System	Set	Timing C−R	Timing C−O	Normal C−R	Normal C−O
0	0	$[-8.0, 7.2]$	$[-7.4, 7.9]$	$[-6.3, 7.9]$	$[-7.8, 6.0]$
1	0	$[-26.9, -5.6]$	$[-25.0, -2.8]$	$[-3.5, 13.9]$	$[-6.6, 11.4]$
2	0	$[-48.9, -27.9]$	$[-49.9, -29.3]$	$[-9.8, 12.9]$	$[-9.8, 12.9]$
3	0	$[-20.2, 16.9]$	$[-18.3, 18.9]$	$[-16.2, 16.4]$	$[-16.3, 16.5]$
4	0	$[-25.8, 15.4]$	$[-24.6, 16.1]$	$[-18.5, 20.8]$	$[-19.4, 18.8]$
5	0	$[-39.4, -9.6]$	$[-29.7, 2.2]$	$[-15.5, 16.1]$	$[-28.0, 5.1]$
6	0	$[-48.8, -12.8]$	$[-49.7, -15.3]$	$[-16.1, 20.6]$	$[-16.5, 20.3]$
7	0	$[-50.4, -11.0]$	$[-53.9, -17.4]$	$[-22.2, 24.4]$	$[-21.1, 24.1]$
8	0	$[-50.8, -14.0]$	$[-47.5, -7.9]$	$[-19.6, 21.4]$	$[-20.9, 20.6]$
9	0	$[-48.8, -37.1]$	$[-49.3, -37.4]$	$[-5.0, 9.5]$	$[-6.1, 8.7]$
0	1	$[-7.6, 6.0]$	$[-4.9, 8.6]$	$[-3.4, 8.5]$	$[-5.1, 6.9]$
0	2	$[-8.1, 7.3]$	$[-6.0, 8.8]$	$[-4.9, 8.2]$	$[-6.8, 6.5]$
1	1	$[-25.1, -4.5]$	$[-24.0, -2.2]$	$[-1.7, 14.7]$	$[-5.6, 12.6]$
1	2	$[-20.7, 1.7]$	$[-20.1, 3.5]$	$[-1.4, 16.8]$	$[-2.8, 16.4]$
2	1	$[-48.8, -28.7]$	$[-48.9, -28.6]$	$[-9.8, 13.2]$	$[-9.9, 13.2]$
2	2	$[-48.9, -28.6]$	$[-49.7, -30.1]$	$[-10.5, 12.8]$	$[-10.5, 12.8]$

Table 5: **Transfer to crafting and placement.** Timing columns show 95% paired state-bootstrap envelopes for coverage-minus-replay (C−R) and coverage-minus-original (C−O) error differences, in percentage points. Normal U summarizes preservation as the maximum of the corresponding upper 95% bounds on normal-execution error differences, with $U \leq 5$ meeting the five-point tolerance against both references.

		Tool crafting			Workbench placement		
System	Set	Timing C−R	Timing C−O	Normal U	Timing C−R	Timing C−O	Normal U
0	0	$[-8.6, -3.9]$	$[-2.7, 2.9]$	0.5	$[-18.0, 18.1]$	$[-18.1, 17.8]$	17.7
1	0	$[-4.4, 6.7]$	$[-3.4, 7.3]$	4.9	$[-18.2, 18.1]$	$[-17.9, 18.3]$	17.8
2	0	$[-7.6, 2.7]$	$[-7.3, 3.1]$	5.0	$[-4.5, 6.6]$	$[-3.9, 7.3]$	5.1
3	0	$[-6.5, 4.0]$	$[-10.9, 0.4]$	5.1	$[-5.6, 5.4]$	$[-5.5, 5.6]$	5.5
4	0	$[-2.1, 6.3]$	$[-2.9, 5.5]$	6.3	$[-4.4, 5.1]$	$[-4.1, 5.1]$	5.0
5	0	$[-3.4, 6.9]$	$[-4.0, 6.3]$	5.2	$[-6.4, 5.8]$	$[-5.7, 5.8]$	5.4
6	0	$[-9.4, 0.8]$	$[-2.8, 7.3]$	7.6	$[-5.7, 5.8]$	$[-5.4, 5.7]$	5.5
7	0	$[-4.2, 6.0]$	$[-3.7, 6.6]$	5.4	$[-5.2, 5.9]$	$[-5.0, 6.0]$	5.7
8	0	$[-2.3, 8.0]$	$[-2.5, 7.9]$	6.5	$[-7.8, 8.5]$	$[-8.0, 8.4]$	8.2
9	0	$[-4.9, 5.2]$	$[-5.1, 5.1]$	5.2	$[-7.7, 7.5]$	$[-7.6, 7.6]$	7.2
0	1	$[-4.2, 0.6]$	$[3.0, 8.9]$	6.8	$[-18.0, 17.6]$	$[-17.9, 17.8]$	17.2
0	2	$[-1.3, 3.4]$	$[-0.1, 5.7]$	2.9	$[-18.1, 18.1]$	$[-18.4, 17.5]$	17.6
1	1	$[-1.3, 9.3]$	$[-1.3, 9.4]$	4.9	$[-18.2, 17.9]$	$[-18.1, 18.0]$	17.6
1	2	$[-2.4, 7.8]$	$[1.2, 12.3]$	5.4	$[-18.5, 18.0]$	$[-18.2, 18.2]$	17.7
2	1	$[-6.0, 4.1]$	$[-6.2, 4.1]$	4.9	$[-4.8, 6.5]$	$[-4.0, 7.2]$	5.2
2	2	$[-6.3, 3.8]$	$[-6.5, 3.7]$	4.8	$[-5.0, 6.1]$	$[-3.4, 8.0]$	5.2

without a corresponding average gain in the other events or ordinary prediction scores relative to replay.

Table 6: **Ordinary prediction after model updates.** Cells show mean error differences [95% paired state-bootstrap intervals] over ten set-0 systems on the same 128 ordinary states, in the units indicated by each row. NLL and classification errors use the first native successor, while token mismatch uses four generated steps. Likelihood differences are computed within each tokenizer before averaging.

Metric	Coverage − replay	Coverage − original
Observation NLL (10^{-3})	-0.539 $[-1.285, 0.204]$	0.874 $[0.623, 1.174]$
Token mismatch (10^{-3})	0.016 $[-0.483, 0.552]$	-0.788 $[-0.964, -0.611]$
Reward NLL (10^{-3})	-1.917 $[-4.890, 0.429]$	1.796 $[-0.524, 3.871]$
Reward error (10^{-3})	-1.506 $[-3.686, 0.000]$	0.577 $[0.000, 1.731]$
Termination NLL (10^{-8})	1.63 $[0.78, 2.67]$	-17.78 $[-42.36, -2.81]$
Termination error	0	0

F Downstream Transition Substitutions

Dependent event sequences. We evaluate whether changing the initial transition alters a later event that requires its outcome. In Craftax, acquiring iron supplies a prerequisite for a subsequent iron-pickaxe crafting action. The initial state has a stone pickaxe, no iron, and no iron pickaxe. The native continuation reaches the required facilities with the remaining recipe materials and contains no opportunity to acquire replacement iron. We retain the first 128 episodes in the fixed episode-then-producer order, from 411 eligible opportunities among 4,000 attempted episodes, without filtering model outputs or return accuracy. The ten producing actors contribute 7, 10, 17, 14, 6, 17, 12, 20, 14, and 11 states. Routes last 4–20 transitions, with no added no-op padding. Execution and cancellation differ only in the first acquisition action and then share this route and final crafting action.

The endpoint event requires an iron-pickaxe inventory increase and an iron-inventory decrease of exactly one on the final crafting transition. Other recipe resources and survival are recorded separately. The terminal transition counts if it satisfies the event; later absorbing states cannot produce it. All model predictions begin from the complete observed history and retain their full generated continuation. We use 64 native and 128 model draws per branch and path.

Separating the changed transition from its continuation. For each of Systems 0–2 and each of its three training sets, let R denote the replay-updated generator and C the action-coverage generator. We evaluate four paths: RR, CR, RC, and CC, where the first letter chooses the initial transition and the second chooses all later transitions. For example, CR changes the first transition while retaining the replay model downstream. Pairs sharing a first generator reuse its sampled prefix. The later generator processes the complete emitted history using its own parameters, and we sample the continuation with independent randomness. One original-model path per system provides a further reference.

Let $\mathcal{R}_{ab}$ be the squared environment–model effect error for path ab, estimated with the independent-half product in Appendix C. The comparisons

$$G_{\text{first}} = \mathcal{R}_{RR} - \mathcal{R}_{CR}, \quad G_{\text{later}} = \mathcal{R}_{RR} - \mathcal{R}_{RC}, \quad G_{\text{full}} = \mathcal{R}_{RR} - \mathcal{R}_{CC}$$

measure replacing the first transition, the continuation, and both. Positive differences indicate lower effect error. Bounds cancel reused random variables before evaluating the remaining 16 endpoint combinations.

Coverage and interpretation. Four selected episodes fail at least one model-specific environment readout check, producing five invalid system–state pairs. The paired valid counts are 127 for System 0 and 126 for Systems 1 and 2 in each training set; 15 of 1,152 system–set–state evaluations are invalid and not replaced. We weight valid states equally and use 10,000 paired episode-bootstrap resamples of the originally selected 128 states, applying each system's valid subset afterward. The intervals quantify variation across states conditional on these models, readout rules, and sampled rollouts.

Table 7 reports the three substitution gains, and Table 8 gives the absolute event probabilities behind them. First-transition substitution reduces effect error across all three training repetitions of System 2, primarily by suppressing false events after cancellation. Normal crafting remains rare in every path, including the original generator. Systems 0 and 1 predict no joint crafting event in these sampled rollouts, leaving large effect errors despite the absence of cancellation hallucinations. The attributed effect belongs to the whole generated transition, including any altered fields and their subsequent use by the continuation.

Table 7: **First-transition changes reduce downstream effect error in System 2.** R and C denote replay and coverage, with the two letters specifying the first and later generators. Gains measure reductions in squared probability-effect error relative to RR. RR risk is bounded by semantic ambiguity, while gain columns show 95% paired state-bootstrap envelopes. Systems 0–1 have identical zero gains across training sets because their sampled rollouts contain no joint crafting events (Table 8).

System	Set	RR risk	G_{first}	G_{later}	G_{full}
0	0–2	0.7247	0	0	0
1	0–2	0.7303	0	0	0
2	0	$[0.7293, 0.7296]$	$[0.0209, 0.0762]$	$[-0.0031, 0.0029]$	$[0.0185, 0.0694]$
2	1	$[0.7316, 0.7317]$	$[0.0177, 0.0695]$	$[-0.0009, 0.0064]$	$[0.0199, 0.0731]$
2	2	$[0.7270, 0.7271]$	$[0.0142, 0.0611]$	$[-0.0039, 0.0016]$	$[0.0133, 0.0587]$

Table 8: **Downstream crafting under execution and cancellation.** Cells show mean semantic bounds in percent for the joint event of tool increase and iron expenditure. Native cancellation has zero probability, making the native normal probability equal to the native effect. Rows marked All combine identical zero outcomes across three training sets and four generator paths. Model crafting probabilities under normal execution remain far below the native reference.

System	Set	Path	n	Native normal	Model normal	Model cancel	Model effect
0	All	All	127	82.38	$[0.000, 0.000]$	$[0.000, 0.000]$	$[0.000, 0.000]$
0	—	Original	127	82.38	$[0.000, 0.000]$	$[0.000, 0.000]$	$[0.000, 0.000]$
1	All	All	126	82.94	$[0.000, 0.000]$	$[0.000, 0.000]$	$[0.000, 0.000]$
1	—	Original	126	82.94	$[0.000, 0.000]$	$[0.000, 0.000]$	$[0.000, 0.000]$
2	0	RR	126	82.47	$[3.137, 3.156]$	$[3.224, 3.237]$	$[-0.099, -0.068]$
2	0	CR	126	82.47	$[3.100, 3.113]$	$[0.062, 0.062]$	$[3.038, 3.051]$
2	0	RC	126	82.47	$[2.741, 2.753]$	$[2.840, 2.846]$	$[-0.105, -0.087]$
2	0	CC	126	82.47	$[2.648, 2.660]$	$[0.043, 0.043]$	$[2.604, 2.617]$
2	1	RR	126	82.47	$[2.505, 2.505]$	$[2.722, 2.728]$	$[-0.223, -0.217]$
2	1	CR	126	82.47	$[2.449, 2.449]$	$[0.031, 0.031]$	$[2.418, 2.418]$
2	1	RC	126	82.47	$[2.734, 2.734]$	$[2.827, 2.834]$	$[-0.099, -0.093]$
2	1	CC	126	82.47	$[2.666, 2.666]$	$[0.019, 0.019]$	$[2.648, 2.648]$
2	2	RR	126	82.47	$[2.393, 2.393]$	$[2.356, 2.362]$	$[0.031, 0.037]$
2	2	CR	126	82.47	$[2.338, 2.344]$	$[0.012, 0.012]$	$[2.325, 2.331]$
2	2	RC	126	82.47	$[2.319, 2.319]$	$[2.282, 2.288]$	$[0.031, 0.037]$
2	2	CC	126	82.47	$[2.245, 2.251]$	$[0.006, 0.006]$	$[2.238, 2.245]$
2	—	Original	126	82.47	$[3.881, 3.937]$	$[3.776, 3.801]$	$[0.081, 0.161]$

G Return and Event Validation

G.1 Event Readouts and Transition Substitutions

Task-event readout. The transition-substitution examples evaluate the first visible acquisition of a tool absent from the initial inventory. Environment inventory records establish its first appearance and corresponding achievement increment. Task rewards use the TWM binary target $\mathbf{1}\{r_t \geq 0.5\}$.

For each model's tokenizer, we match target-cell categories to zero and one tool using native frames with known inventory counts. All generated target categories in the selected cases belong to these unambiguous sets. Let τ be the first time a sampled nonterminal observation contains the target tool. The semantic return is

$$U_{\text{tool}} = \sum_{t=1}^{H} \gamma^{t-1} \mathbf{1}\{\tau = t\}, \qquad \gamma = 0.925.$$

A tool's first appearance remains counted if the tool later disappears or the branch terminates; a later reappearance is not counted again. Terminal observations use the absorbing readout, so tokens at or after the boundary cannot create a new visible event. No first raw tool appearance coincides with a terminal boundary in these cases. We record disappearance, reappearance, first appearance on an unrelated action, and unmapped categories separately. Unmapped nonterminal categories receive unknown labels. The model readout measures the first visible tool event in its emitted observations.

Return and semantic contributions. Task returns use the native-prefix/model-suffix construction in Appendix C; U_{tool} is evaluated on that same complete hybrid trajectory. All observed-prefix tokens and termination boundaries match their assigned environment references.

For task return and the tool score, the margin is branch 1 minus branch 0; adjacent-suffix differences use Equation 3 with the corresponding scalar readout. Increasing L replaces the environment transition producing observation $H - L + 1$ with its model counterpart. In the retained environment comparisons, non-target rewards have zero branch difference, so the task-return and target-event margins agree even though either branch may contain other rewards.

Case sampling. Each case contains eight saved environment references and four paired model draws per reference. We preserve draw identities and reference assignments across suffix lengths. Retrospectively selected from an earlier analysis set, these cases illustrate how transition substitutions alter delayed tool events and return margins.

G.2 Tool Effects on Held-Out Episodes

Episode sampling. Beyond the selected cases above, the validation study evaluates the ten trained Craftax systems on held-out actor episodes. An eligible query has a full 20-block input history and 20-transition continuation, with a tool initially absent and its crafting action among the shared actions at offsets $1, \ldots, 19$. Targets are wooden, stone, and iron pickaxes and swords. We select one eligible query uniformly per episode and retain all its eligible tools. Episodes therefore receive equal weight regardless of their numbers of opportunities. Inclusion uses inventories and recorded actions, without effect-energy or model-output filtering; failed crafting attempts, terminated branches, and zero effects remain.

The completed sample contains 256 eligible episodes per training seed: 2,560 independent episodes and 2,969 query–tool pairs. Collection encountered 2,616 episodes; the 56 without an eligible query are retained in the collection records. The target distribution is an actor episode conditioned on having an eligible query, followed by a uniform query within that episode. At each query, the actor's two highest-probability actions form the initial alternatives; both branches then follow the same 19 recorded actions. Each query has 64 paired environment references and 128 paired model trajectories. Model generation starts at the query history and covers all 20 transitions. Task returns use $\gamma = 0.925$, native targets $\mathbf{1}\{r_t \geq 0.5\}$, and the model reward predictor's positive-class probabilities. The terminal transition retains its reward; subsequent rewards are zero.

Event semantics. We apply the first-visible-tool return U_{tool} defined above to each eligible tool. Native inventory counts define presence. Model tokens use a presence map for each training seed and tool, based on native counts, token categories, and rendered frames. Count labels establish absence or presence; positive counts also supply icon templates from the 33 pixels outside the central 4×4 count overlay of each 7×7 HUD patch. Codebook matches use maximum channel error $1/1024$; an all-zero patch denotes absence. Conflicting and unmatched categories remain unknown. Mapping covers the entire codebook without reference to generated-category frequencies.

For an unknown category, the lower event return uses the first definite appearance, or zero if none occurs; the upper return uses the first possible appearance. Tokens at or after termination cannot

initiate an event, while an earlier event remains counted. Branch averages and differences propagate these intervals to bounds on event-margin error. All 12,176 unknown active tool-token readings, spanning 224 episodes, remain represented. These are semantic bounds induced by ambiguous token decoding. This event readout measures tool appearance; the task return separately aggregates all binary reward targets.

Independent filtering and measurement. Return filtering and event measurement use independent samples to avoid reusing the same sampling noise in both quantities. We split the 128 model pairs into two disjoint halves of 64 and the 64 environment pairs into two disjoint halves of 32. For each direction, one model/environment half filters by return accuracy and the other measures event-margin error. We exchange the halves and average the two assessments. The return criterion requires the estimated return error of *each* branch to be at most 0.05. A large event error means that the estimated model and environment first-tool-event margins differ by at least 0.10. All margins subtract branch 0 from branch 1, and all tolerances are in discounted return units, without division by the environment margin. We chose these tolerances before collecting the evaluation data.

We report the fraction meeting the return criterion (coverage), the joint fraction meeting that criterion with a large event error, and the conditional fraction obtained by dividing the joint fraction by coverage. Tools and the two assessment directions are averaged within episode; episodes are averaged within training seed; the ten systems then receive equal weight. Conditional fractions are ratios of these aggregate averages, not averages of within-seed ratios. We compute both ends of the joint and conditional fractions using the semantic bounds. The two directions are correlated measurements from the same episode.

The native and model random streams are independent. Coverage and joint averages are unbiased for this finite-draw procedure; their ratio need not be. Filtering therefore applies to estimated returns at the stated sampling budget; Appendix G.3 tests accuracy in expectation separately.

Table 9: **Tool-effect errors remain after per-branch return filtering in all ten systems.** Each trained system is evaluated on 256 episodes. Coverage is the fraction satisfying the both-branch return criterion; conditional event error is the fraction of retained assessments with first-tool-event margin error at least 0.10. Ranges quantify ambiguity in token decoding. The aggregate divides the equal-system joint fraction by equal-system coverage. The bottom row measures tool appearance by the horizon, irrespective of onset time.

Training seed	Coverage (%)	Conditional event error (%)
0	36.52	10.43–12.03
1	44.14	16.59
2	33.01	9.76
3	43.16	3.85
4	45.70	9.83
5	39.45	3.47
6	34.57	11.30
7	24.80	46.46
8	43.36	18.17–19.07
9	25.59	11.45
Aggregate	37.03	12.99–13.26
Timing-independent readout, all systems		
Both returns, ever present	37.03	17.66–18.00

Uncertainty and system variation. All ten systems contribute missed tool-effect errors, with conditional frequencies from 3.47% to 46.46% (Table 9); the aggregate 13.0–13.3% averages over these substantially different systems. The interval $[11.08, 15.39]\%$ describes episode variation for the aggregate over these ten systems.

We resample episodes within each training seed in 2,000 bootstrap replicates, retaining their tools and both assessment directions. Each replicate recomputes the complete ratio. Reported envelopes combine the lower semantic bound's 2.5th percentile and the upper bound's 97.5th percentile.

A secondary readout records whether the tool ever appears before termination within 20 steps, ignoring its timing. Its conditional effect-error fraction is $[17.66, 18.00]\%$, showing that the mismatch includes tool presence, beyond the timing of its first appearance. This readout differs from tool presence at the final step.

Development and sampling precision. Development used four independent episodes per training seed to choose 64 environment and 128 model pairs per query. We then collected the 256 evaluation episodes per system, excluding the development states. The two independent halves disagree on the both-branch return criterion for 6.48% of states, illustrating the sampling variability of this finite-draw validation procedure.

G.3 Expected Returns and Event Effects at Selected States

The evaluation above measures what a finite-sample return check misses. To determine whether this gap persists beyond sampling noise, we test whether accurate expected returns coexist with incorrect expected event effects at the same state. We selected ten states, one per training seed, and their twelve tool readouts before collecting additional samples. Within each seed, the first entry in the earlier case list determines the state; all its listed tools are retained. Selection uses earlier results, whereas all samples below are new and independent of those results.

Each state receives $4{,}096$ environment and $8{,}192$ model pairs, with the original actions, $H = 20$, $\gamma = 0.925$, binary reward target, and discounted first-tool-event readout unchanged. One environment pair and two independent model pairs form each of $4{,}096$ independent groups. Branches remain paired within each rollout pair. Unknown tool categories retain their lower and upper event scores, using the state's original environment-derived token dictionary.

Table 10: **Expected-return accuracy and event-effect error at selected states.** Return columns give empirical signed errors (model minus environment) for branches 0 and 1. Event error is the empirical magnitude, bounded where token semantics are unknown. "Yes" records a conclusion from the tests defined in this section; "–" denotes no conclusion. A row with Yes in both final columns establishes accurate expected returns despite an incorrect expected tool effect. All ten selected states and twelve tool readouts are shown.

State (seed/query)	Tool	Return errors ($b=0,\ b=1$)	Event error	Both returns $\lvert\text{error}\rvert < 0.05$	Event error > 0.10
0/2360297	Wood sword	$(+0.0279, -0.0743)$	0.7854	–	Yes
1/50071	Stone pickaxe	$(-0.0230, +0.0127)$	$[0.0375, 0.0378]$	–	–
1/50071	Stone sword	$(-0.0230, +0.0127)$	$[0.4830, 0.4885]$	–	Yes
2/960171	Stone sword	$(-0.0220, +0.0098)$	$[0.1354, 0.1359]$	–	Yes
3/460071	Wood sword	$(+0.0023, -0.0109)$	$[0.4851, 0.4854]$	Yes	Yes
4/2650336	Stone pickaxe	$(-0.0107, +0.0141)$	0.0073	Yes	–
4/2650336	Stone sword	$(-0.0107, +0.0141)$	0.5370	Yes	Yes
5/1290171	Wood sword	$(+0.0405, +0.0375)$	0.9250	–	Yes
6/500090	Stone pickaxe	$(-0.0090, +0.0046)$	$[0.2687, 0.2692]$	Yes	Yes
7/1940342	Iron sword	$(+0.0096, +0.0148)$	$[0.6627, 0.6662]$	Yes	Yes
8/1370209	Stone sword	$(-0.0017, +0.0296)$	$[0.5384, 0.5385]$	–	Yes
9/980232	Iron pickaxe	$(+0.0005, -0.0005)$	$[0.1199, 0.1236]$	Yes	Yes

At five of the ten tested states, accurate expected returns coexist with at least one incorrect expected tool effect (Table 10). Event error alone is established for ten of the twelve readouts. The tests below account for sampling uncertainty when establishing these expected discrepancies. The separate episode sample estimates how often finite-sample return validation misses them.

Tests of expected prediction errors. For one branch, let D_i be the mean model return in group i minus its environment return. Then $D_i \in [-R_{\max}, R_{\max}]$, where $R_{\max} = \sum_{t=0}^{19} 0.925^t \approx 10.52936$. To test the nulls $\mathbb{E}D_i \geq 0.05$ and $\mathbb{E}D_i \leq -0.05$, respectively, we use increments $X_i = 0.05 - D_i$ and $X_i = 0.05 + D_i$ in the fixed mixture

$$M(X;\lambda) = \frac{1}{4}\sum_{k=0}^{3}\prod_{i=1}^{4096}\left(1 + 2^{-k}\lambda X_i\right), \qquad \lambda = \frac{0.9}{R_{\max} + 0.05}.$$

Rejecting both tails for each of the two branches establishes that both expected return discrepancies lie strictly inside $(-0.05, 0.05)$.

For one tool, let D_i^- and D_i^+ bound the groupwise model-minus-environment event-margin discrepancy; both lie in $[-2, 2]$. To establish an expected discrepancy above 0.10, use $X_i = D_i^- - 0.10$; to establish one below -0.10, use $X_i = -D_i^+ - 0.10$. Both tests use the same mixture with $\lambda = 0.9/2.10$. The corresponding nulls are that the expected true discrepancy is at most 0.10 or at least -0.10. Rejecting either establishes event-effect error greater than 0.10. Each mixture uses all four predefined components with weight $1/4$, computed in log space. The weights are chosen before observing the samples.

The test family defined before sampling comprises 100 scalar tests: four return tails for each of sixteen states and two event tails for each of eighteen tool readouts. We evaluated ten states and twelve readouts, conducting 64 tests. The 36 unperformed tests retain their error allocation, so the rejection threshold remains $100/0.025 = 4{,}000$.

To see the error control, under each stated null the corresponding increment has expectation at most zero. Its factor is positive throughout the bounded support and has expectation at most one. Independence across groups makes each product have expectation at most one; their fixed convex mixture preserves this property. Markov's inequality bounds the rejection probability of a true null by $0.025/100$, and a union bound controls all 100 tests at 0.025. Semantic lower and upper bounds preserve the event null inequalities. Dependence between branches, tools, or tests does not affect this union bound. The test conclusions concern expected discrepancies at the selected states under their semantic bounds.

H Observation Checks and Retained Action Effects

We reuse all 2,560 episodes and the independent-half return/event measurements of Appendix G.2. We designed this additional analysis of the return-validation set after the return comparison and before examining the complete observation scores. Each assessment consists of a state, an eligible tool, and a choice of filtering half. The observation score uses the same half as return filtering; the other half supplies the event-effect error and environment-effect magnitude. We exchange halves and average.

Categorical observation discrepancy. For a branch, time, and token position, let x_i and y_j be independent one-hot environment and model samples, of sizes n and m. We estimate half the squared distance between their categorical distributions by

$$\ell = \frac{\sum_{i \neq i'} \langle x_i, x_{i'} \rangle}{2n(n-1)} + \frac{\sum_{j \neq j'} \langle y_j, y_{j'} \rangle}{2m(m-1)} - \frac{\sum_{i,j} \langle x_i, y_j \rangle}{nm}.$$

Off-diagonal products remove the sampling-variance contribution to squared empirical frequencies. We retain negative estimates and use absorbing categories after termination. The four scores are: mean over all 81 tokens and 20 times; mean over the 18 HUD tokens and times; maximum over time of the HUD mean; and maximum over time of the eligible tool's token discrepancy. Each score then takes the worse of the two branches. We use empirical maxima to rank assessments. The target-field score incorporates the eligible tool's identity, while the other scores aggregate over the frame or HUD.

Retention and independent measurement. Among assessments satisfying both branch-return criteria, we retain the lowest-scoring fraction of assessment weight at 5% increments from 5% to 100%. Equal-score assessments at the threshold receive the same fractional weight, so exact retention does not depend on state order or event labels. For each retained set, we report the independently measured fraction with tool-effect error at least 0.10 and the fraction of weighted absolute environment-effect magnitude retained from the return-filtered set. We preserve the original equal-system, equal-episode, within-episode tool, and half-assignment weights. Thus 50% retention corresponds to 18.52% of the original opportunity assessments.

Two thousand bootstrap resamples draw episodes within each system, retaining each episode's tools and both halves. Each resample recomputes the weighted thresholds and ratios. The 2.5th percentile

of the lower semantic estimate and 97.5th percentile of the upper estimate form pointwise envelopes conditional on the checkpoints and saved draws. Table 11 gives common retention points from the full curves in Figure 3.

Table 11: **Observation validation at common retention rates.** Percentages use the return-filtered assessment set as the retention denominator. Error ranges are semantic mean bounds. Effect retained is the fraction of weighted absolute environment-effect magnitude.

Check	Retention (%)	Event error (%)	Effect retained (%)
Whole frame	25	7.38	14.38
Whole frame	50	9.34–9.65	37.30
Whole frame	75	11.87–12.08	72.09
HUD mean	25	8.65	33.98
HUD mean	50	7.54	51.17
HUD mean	75	9.00–9.07	71.08
HUD temporal max	25	7.59	28.75
HUD temporal max	50	9.49	52.18
HUD temporal max	75	9.92–9.99	74.26
Target field	25	0.00	18.31
Target field	50	0.42	35.25
Target field	75	1.62	56.25
All checks	100	12.99–13.26	100.00

Stricter validation reduces error while retaining less environment-effect magnitude (Table 11). At 75% retention, the target-field check leaves an event-error rate of 1.62% while keeping 56.25% of the environment-effect magnitude in the return-filtered set. Among retained assessments, 21.2% have absolute environment effect at least 0.10, compared with 27.6% before observation filtering, so the retained set still includes substantial action effects. Reducing retention to 50% lowers error to 0.42%, but also lowers retained effect magnitude to 35.25%. The check therefore trades retained action-effect magnitude for lower prediction error.

I Qualitative Action and Transition Comparisons

These illustrative cases were selected retrospectively from an earlier analysis set. The first-transition example in Figure 5a connects the initial resource difference, a tool available at step 8, and the return margin at step 20. Using the environment for the first transition lets the model continuation recover the tool difference and 96.2% of the return-margin gap. This percentage is $(\hat{m}_{H-1} - \hat{m}_H)/(m - \hat{m}_H)$, using eight environment references and 32 model draws per branch.

A later substitution changes the same kind of delayed dependency. Both conditions use environment histories through step 10. Replacing the environment transition producing step 11 with a model transition reduces predicted stone-pickaxe acquisition at step 16 in branch 0 from 32/32 to 2/32; branch 1 remains at zero, against environment counts of 7/8 versus 0/8. The tool-margin increment of $+0.291$ closely matches the return-margin increment of $+0.295$. The whole-transition substitution changes both a concrete tool consequence and the return margin. We display a paired draw minimizing token-Hamming distance across conditions, branches, and times; counts and margins use all draws.

Action-schedule examples. Figure 6 complements the transition substitutions with action-schedule comparisons, showing how generated events respond when execution is delayed or cancelled (Section 4.2). Its two states come from an earlier diagnostic set and illustrate the contrasting temporal responses at the trajectory level.

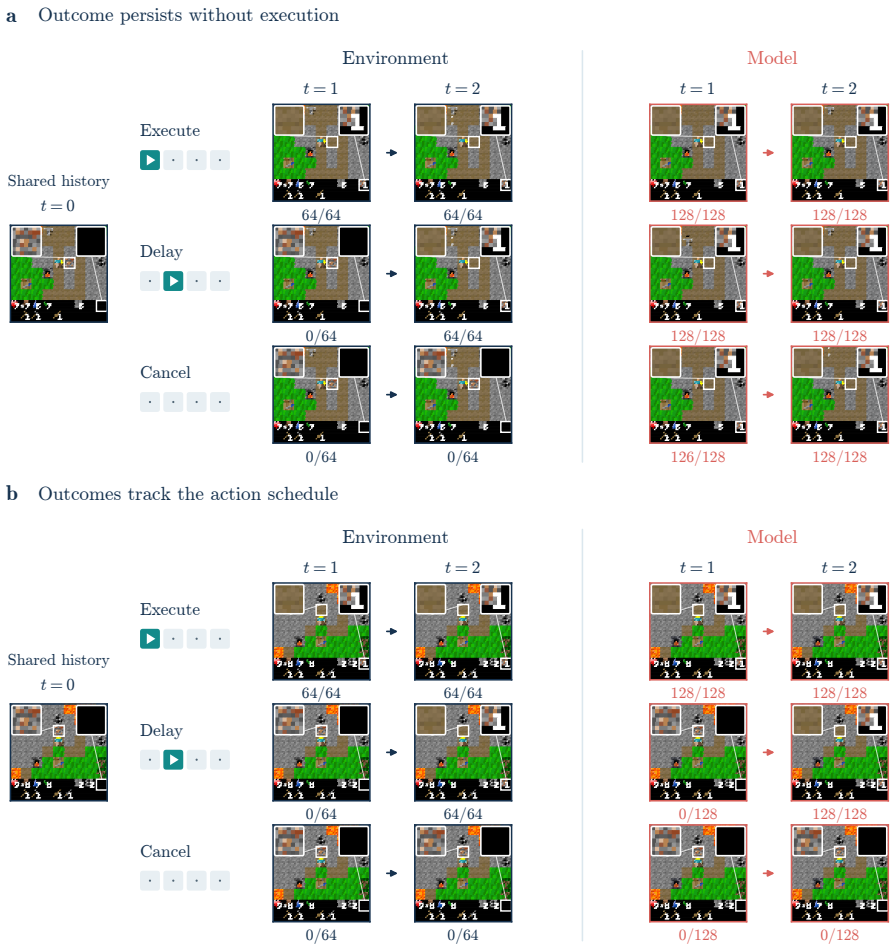


Figure 6: **Contrasting temporal responses to action execution, delay, and cancellation. (a)** Acquisition persists before delayed execution and under cancellation. **(b)** Target depletion and inventory increase follow the action schedule. Panels (a) and (b) use states from an earlier diagnostic set for Systems 2 and 1, respectively. Each set of schedules shares a 20-frame observed history; $t = 0$ shows its last frame. Triangles denote execution and dots no-ops; insets copy the target and iron-inventory regions of their own frame. Images are separate environment and model token-Hamming medoids, selected jointly across all schedules and the two displayed times. Counts below each frame give cumulative joint events across all 64 environment or 128 model draws for that state and schedule; the step-2 counts remain unchanged through step 4. The joint event requires iron increase and designated-target depletion in the same transition.

J Actor Continuation After World-Model Updates

We continue the pretrained actor and critic in Systems 0–2 using seven world models (original, replay, and coverage across three training sets) and three actor seeds per model, giving 63 runs. World-model and tokenizer parameters remain unchanged.

We restore the pretrained actor, critic, and value-normalization state and initialize fresh Adam optimizers in every condition. Training uses a static set of 4,096 ordinary episodes per system, batch size 48, imagination horizon 20, and 10,000 updates. Development evaluations at updates 0, 500, 2,000, and 5,000 use 512 episodes; the reported endpoint is always update 10,000, evaluated on 1,024 independent held-out episodes. The pretrained actor is evaluated on the same held-out episode seeds. Native return is undiscounted and includes negative rewards and the terminal reward. For the 22

achievement probabilities p_j, the Craftax score is $\exp(\frac{1}{22}\sum_j \log(1+100p_j)) - 1$, on a 0–100 scale (Matthews et al., 2024).

All 63 continuations underperform their pretrained actors in return and score, including those using the original world model (Figure 7). Coverage exceeds replay on average in Systems 0 and 1 but underperforms it in System 2 (Table 12).

Table 12: **Mean actor performance after continuation.** Native return and achievement score on 1,024 held-out episodes per run. Original averages three actor seeds; replay and coverage average three seeds within each training set, then the three sets. Figure 7 shows every run.

System	Metric	Pretrained	Original	Replay	Coverage	C−R
0	Return	13.81	1.64	3.15	4.46	1.31
0	Score	37.04	2.46	6.41	9.86	3.44
1	Return	15.08	12.54	12.97	13.06	0.09
1	Score	38.60	32.81	33.62	34.14	0.52
2	Return	15.43	7.67	10.54	7.88	-2.66
2	Score	48.02	23.08	31.99	23.26	-8.73

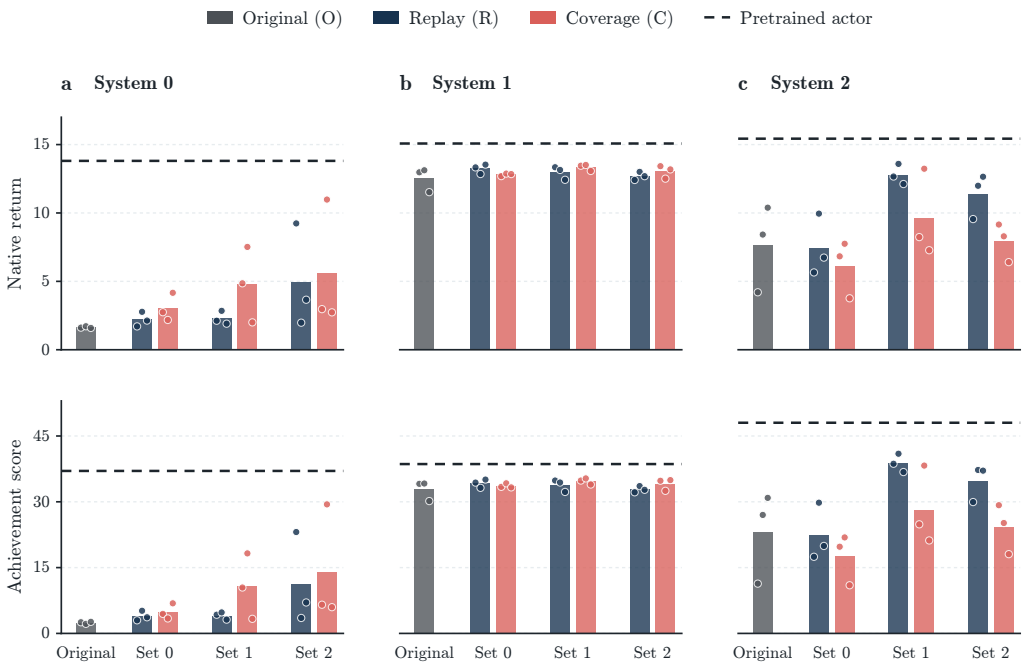


Figure 7: **Actor continuation lowers performance with original and updated world models.** Bars average three actor seeds; dots show individual results in matched seed order. Each run is evaluated on 1,024 held-out episodes after 10,000 updates. Gray, blue, and coral denote original, replay, and coverage models. Dashed lines show pretrained actors. All 63 continuations fall below their baselines in both metrics.