ROLEWORLD: KNOWLEDGE TRANSFER IN WORLD MODELING WITH FUNCTIONAL ROLE ABSTRACTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models enable large language model (LLM) agents to anticipate action effects and make informed decisions. For generalization, it is desirable for such world models to transfer efficiently to new environments with similar interaction structure and underlying dynamics. Recent approaches encode world dynamics as symbolic rules or executable programs, whose compositional structure in principle affords transfer, but still organize such knowledge around environment-specific objects. To address this limitation, we introduce *RoleWorld*, a symbolic world modeling approach which uses reusable prototypes and environment-specific parameter values as *functional role abstractions* to represent how objects behave in interactions. From interaction trajectories, a teacher LLM extracts role prototypes and executable transition rules defined over role instances, together with a skill that guides the actor LLM to use role information in decision making. In a target world, without the teacher, the actor uses this skill together with its current role assignments and parameter estimates, and actively tests a role instance hypothesis through interaction when its uncertainty affects the current decision. Feedback from each action refines role assignments and estimates parameter values online, with no LLM parameter updates and the transition rules held fixed. Experiments on MARS benchmark show that RoleWorld enables robust generalization across environments with significantly altered object behavior, suggesting that functional role abstractions can serve as an effective component for knowledge transfer in world modeling. Our code and data will be open-sourced upon formal release.

1 INTRODUCTION

Learning how the world works enables an agent to anticipate the consequences of its actions and make informed decisions (Schrittwieser et al., 2020). World models capture such knowledge by describing how actions change the environment (Ha & Schmidhuber, 2018; Hafner et al., 2025), allowing agents, such as Large Language Models (LLMs), to reason about possible outcomes before acting (Zhou et al., 2025). Recent evaluations increasingly consider agents across multiple related environments or environment variants, rather than within a single fixed world (Tang et al., 2024b;a). This setting creates a natural need for world models that can reuse previously acquired knowledge when entering a new environment, instead of learning its dynamics entirely from scratch (Taylor & Stone, 2009). Such transfer is plausible because related environments often preserve substantial interaction structure and underlying dynamics even as their particular configurations vary (Tang et al., 2024a). A central question is therefore *how to represent world knowledge* so that it can transfer efficiently across related environments.

Recent approaches use LLMs to express world dynamics as symbolic rules or executable programs (Zhou et al., 2025; Tang et al., 2024a; Piriyakulkij et al., 2025), whose compositional structure provides a natural basis for reuse (Battaglia et al., 2018). However, their rules remain tied to environment-specific objects and can fail when familiar objects serve different functions. This suggests a functional view, as in affordance models relating objects, actions, and effects (Montesano et al., 2008; Khetarpal et al., 2026). We organize such functional knowledge for transfer by separating shared interaction mechanisms from the object–function assignments and parameters that can be revised online.

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.

To address this problem, we introduce *RoleWorld*, a symbolic world modeling approach that uses *functional role abstractions* to represent how objects behave in interactions. Each functional role abstraction consists of a reusable *role prototype* and environment-specific *parameter values*. This factorization separates reusable interaction functions from environment-specific object–function assignments and parameter values. As a result, the same role-level interaction structure can be reused when the objects realizing a function change across environments, while environment-specific differences are captured through role assignments and parameters. From trajectories collected in a source world, a teacher LLM extracts role prototypes and executable *transition rules* defined over role instances, together with a reusable *skill* for role-based decision making. Each transition rule captures an interaction mechanism whose requirements and effects are instantiated through concrete role instances. This formulation is intended to support reuse across environments where different objects instantiate the same role prototypes. In a target world, the actor LLM operates without the teacher, and the role prototypes, transition rules, and skill remain fixed. Guided by the skill, the actor uses the transition rules to predict action outcomes and select actions. When uncertainty about a role assignment could affect its decision, the actor gathers evidence through an informative interaction. Action feedback is then used to refine role assignments and parameter estimates online, allowing RoleWorld to adapt the environment-specific instantiation of its world model.

We evaluate RoleWorld on MARS (Tang et al., 2024b), a challenging benchmark for evaluating generalization in world modeling. The results show that RoleWorld enables robust generalization across environments with significantly altered object behavior, suggesting that functional role abstractions can serve as an effective component for knowledge transfer in world modeling.

2 Related Work

LLM world models support look-ahead planning through prompted predictions and in-context dynamics estimation (Hao et al., 2023; Liu et al., 2024). Induction from Reflection (IFR) and WALL-E induce environment rules from interaction, with WALL-E learning executable rules to correct predictions (Tang et al., 2024b; Zhou et al., 2025). PoE-World composes programmatic experts organized by object type, while OneLife models stochastic dynamics through conditionally activated laws (Piriyakulkij et al., 2025; Khan et al., 2025).

Object-centric abstractions support reuse beyond individual object identities. OO-MDPs and Schema Networks represent interaction rules and local causal schemas (Diuk et al., 2008; Kansky et al., 2017). James et al. (2022) learn object types and lifted planning models reusable across tasks sharing those types, while Marom & Rosman (2024) learn transferable transition and reward models. Adaptation can involve revising transferred code, as in WorldCoder (Tang et al., 2024a), or inferring task-specific hidden parameters of shared dynamics, as in HiP-MDPs (Doshi-Velez & Konidaris, 2016). RoleWorld allows multiple compatible roles per object and adapts their assignments and instance parameters while holding role prototypes and transition rules fixed. We consider target-world changes expressible within this transferred library.

Affordance models provide a functional view of objects through object–action–effect relations (Montesano et al., 2008) and support partial transition modeling (Khetarpal et al., 2020). Khetarpal et al. (2026) extend affordances to multiple tasks and improve search with LLM partial models. SR-WM (Cheng et al., 2026) binds visual entities to task-specific roles such as gripper and target. Our roles describe reusable interaction functions with parameterized requirements and effects. Transferring these functions still requires inferring which objects instantiate them and with what parameter values in the new environment.

Active model learning offers a way to resolve such uncertainty by selecting interactions that test hypotheses. WorldLLM (Levy et al., 2025) combines Bayesian inference over natural-language hypotheses with curiosity-driven exploration. OPINE-World (Courtis et al., 2026) combines program synthesis, hypothesis testing, and replay verification, using a Bayesian measure of object-type adequacy to guide exploration. In RoleWorld, a fixed skill guides such testing when uncertainty about a role instance hypothesis affects the current decision. Feedback refines role assignments and instance parameters while the prototypes and transition rules remain fixed, without further teacher calls.

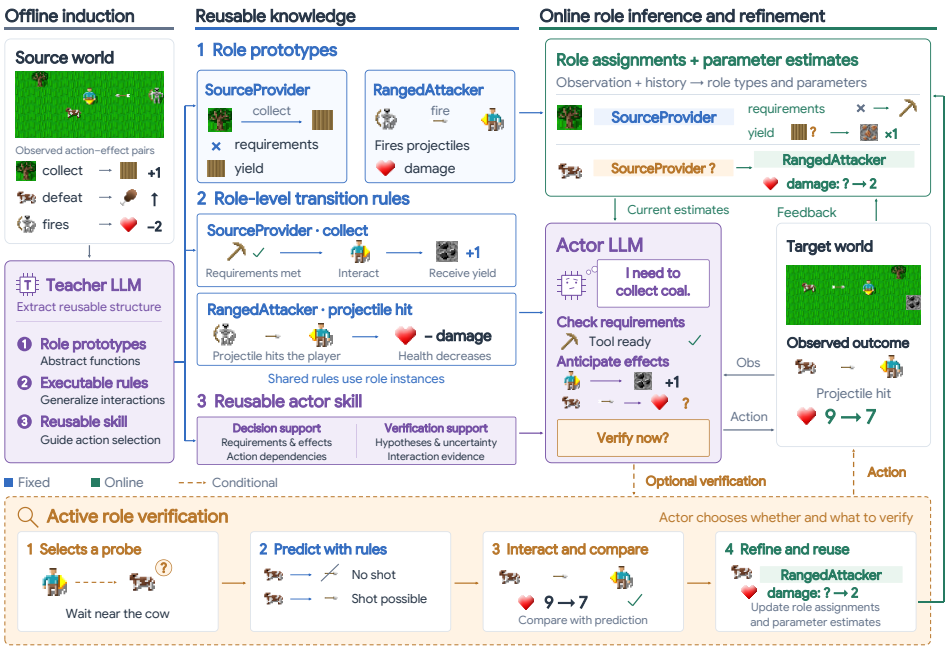

Figure 1: **Overview of RoleWorld**. A teacher LLM induces role prototypes, transition rules, and an actor skill from source trajectories. These remain fixed in the target world, where the actor refines role estimates online and tests hypotheses when uncertainty could affect its decision.

3 METHOD

RoleWorld uses functional role abstractions to separate reusable interaction structure from environment-specific instantiations. Our framework consists of three components. First, a teacher LLM constructs a symbolic world model in the source world, consisting of reusable role prototypes and executable transition rules defined over role instances. Second, in a target world, we infer and progressively refine object–role assignments and environment-specific parameter values from interaction feedback, while keeping the prototypes and transition rules fixed. Third, a reusable skill guides the actor to make decisions using current role estimates and transition rules, and to take actions to test those estimates when role uncertainty affects the current decision.

3.1 OFFLINE WORLD MODEL CONSTRUCTION

Given source-world trajectories, we use an LLM teacher to jointly construct a role-based world model $\mathcal{M} = (\mathcal{R}, \mathcal{T})$, where $\mathcal{R}$ is a library of reusable role prototypes and $\mathcal{T}$ is a set of executable transition rules over instances of these prototypes. The teacher constructs these rules from interaction evidence using predefined event types and operators in a domain-specific language (DSL).

The teacher proposes candidate functions and jointly specifies their role prototypes and transition rules, including triggering events, participants, conditions, effects, and the required parameters and state variables. It executes candidate rules to compare their predicted observable outcomes with real trajectories and obtains additional evidence through environment-interaction tools when needed. To resolve discrepancies, it revises parameter values or rule conditions and effects, adding or splitting prototypes and their associated rules when an independent function requires a separate representation. It merges candidate prototypes and their transition rules when they share triggering events, participant structure, and condition–effect structure, with differences in concrete requirements and effects captured by the parameter values of role instances. Each structural revision is checked against accumulated interaction evidence. Appendix A.1 describes the construction procedure.

Role prototypes. A role prototype is the reusable component of a functional role abstraction: it defines a behavioral function that different concrete objects can instantiate. We represent it as

$$\rho = (\mathcal{C}_\rho, \Theta_\rho, \mathcal{X}_\rho),$$

where $\mathcal{C}_\rho$ is the set of role prototypes that can be instantiated alongside ρ on the same object; Θ_ρ and $\mathcal{X}_\rho$ define the parameter and state spaces of its instances. An object may instantiate multiple compatible prototypes when it supports multiple functions.

Role instances. A role instance is the instantiation of a role prototype ρ for a specific object in an environment e, with parameter values $\theta_\rho^{(e)} \in \Theta_\rho$ and, at each interaction step t, a state $x_{\rho,t}^{(e)} \in \mathcal{X}_\rho$, where the object index is omitted for brevity. A role assignment associates an object with a prototype; the parameter values specify the resulting instance's interaction requirements and effects. These values may differ across objects and environments but remain fixed for a given object within an environment. The values of the state variables may change at each interaction step. We constrain the teacher to declare only parameters and state variables that are referenced by at least one transition condition or effect.

Role-level transition rules. Transition rules describe interaction dynamics over role instances. The teacher constructs executable transition rules in the DSL, defining their conditions and effects in terms of the instances' parameters and states, as illustrated in Figure 1. We represent a transition rule as

$$\tau = \big(\mathrm{event}, P_\tau, g_\tau, F_\tau\big),$$

where event is the triggering event, P_τ specifies participant variables and their required role prototypes, g_τ specifies the conditions under which the rule applies, and F_τ specifies the resulting state changes and interaction outcomes. At execution time, participant variables bind role instances of the required prototypes. Parameters specialize a reusable rule for a particular role instance, while states track properties that evolve through interaction. For stochastic interactions, a rule specifies a distribution over trajectory-supported outcomes, with the distribution expressed in terms of the instances' parameters and states. Because transition rules reference role prototypes and the parameters and states of their instances rather than concrete object identities, the same rule can be instantiated by different objects with different parameter values across environments.

3.2 Online Role Inference and Refinement

With role prototypes and transition rules held fixed, online role inference estimates object–role assignments and environment-specific role parameters from interaction feedback. RoleWorld initializes these role estimates from the role assignments and parameter values retained in the source world, treating them as revisable priors. For each encountered object, it maintains role instance hypotheses with associated probabilities. Each role instance hypothesis is a candidate instantiation for an object, specifying a role type, identified by its prototype, and a particular set of parameter values. For each interaction, we evaluate combinations of role instance hypotheses, considering only the prototypes and parameters referenced by the relevant transition rules. We predict the interaction effects by directly executing these DSL rules with the current states and action under each evaluated combination. We express the effects recorded in environment feedback in the same DSL operation representation and compare them with the predictions to update hypothesis probabilities and guide revisions to role assignments and parameter values.

Distinguishing parameter errors from role assignment errors. A prediction error does not necessarily imply an incorrect role type, since interaction outcomes also depend on environment-specific parameters. If the predicted and observed DSL operations have the same categories and multiplicities but differ in their concrete argument values—for example, predicting a yield of wood while observing a yield of coal (see Figure 1)—the mismatch can still be explained by incorrect parameter values. Here, an operation category identifies the operation kind and, for state changes or resets, the affected field. The model therefore first revises the parameters referenced by the responsible rules, considering both existing alternatives and new values suggested by feedback. Only when parameter revision cannot explain a category-level discrepancy does the model reconsider the role assignments. It then considers additions, removals, or substitutions of assigned prototypes within the existing library, subject to compatibility constraints, and jointly re-evaluates the parameter values in the resulting role instance hypotheses.

Updating role instance hypothesis probabilities from interaction feedback. After identifying whether a prediction discrepancy calls for a parameter or role assignment revision, RoleWorld evaluates the resulting combinations of role instance hypotheses against the observed interaction. Let H denote a combination of compatible role instance hypotheses for the objects participating in the interaction. We compare each predicted outcome o with the observed outcome o_t^{obs} as multisets of DSL operations. Before comparison, repeated state and quantity changes are consolidated into their net effects. An operation matches only when its kind and all argument values agree. Let $\mathcal{O}(H)$ denote the possible outcomes predicted under H; deterministic rules predict a single outcome with probability one. We define the feedback score as

$$\mathcal{L}_t(H) = \sum_{o \in \mathcal{O}(H)} P(o \mid H)\, \mathbf{1}[o = o_t^{\text{obs}}],$$

where the indicator requires an exact match of the operation multisets. The resulting feedback score reweights the probability of each combination as

$$P_t(H) \propto P_{t-1}(H)\, \mathcal{L}_t(H), \tag{1}$$

with normalization over the evaluated combinations. If no evaluated hypothesis combination assigns positive probability to the observed outcome, we treat the observation as unexplained and trigger hypothesis expansion or revision rather than performing the probability update.

The initial probabilities of hypothesis combinations are approximated from those of their constituent role instance hypotheses. To bound inference, each update evaluates only a finite set of high-probability combinations rather than the full Cartesian product of hypothesis sets. Marginal confidence in an individual role instance hypothesis is obtained by summing the updated probabilities of the combinations that contain it.

Validating revisions against accumulated evidence. After updating hypothesis probabilities, we determine whether to accept a proposed revision to role assignments or parameter values by checking it against the complete interaction history. A revision from the current hypothesis combination H to a proposed combination H' is accepted only if it improves the cumulative log feedback score:

$$\sum_{i=1}^{t} \log_2 \mathcal{L}_i(H') > \sum_{i=1}^{t} \log_2 \mathcal{L}_i(H). \tag{2}$$

Each revision is checked against the trajectory collected so far within the current episode. Thus, a new observation can suggest a revision, but accumulated evidence determines whether it replaces the current explanation. Role instance hypotheses that remain observationally indistinguishable are retained until subsequent interactions provide discriminating evidence.

3.3 Role-Guided Decision Making

Effective use of inferred role information involves both applying current role knowledge to decisions and checking uncertain assumptions through interaction. We use a reusable skill to guide the actor in both activities.

Organizing role information for the actor. The teacher examines the actor's interaction trajectories from the source world and develops a program that organizes role information around action requirements, predicted effects, and dependencies between actions. Together with predefined instructions for decision making and active verification, this program forms the skill.

At each step, the skill uses the current role instance hypotheses and their probabilities, observations, and interaction history to present relevant action candidates. The shared transition rules provide the requirements and predicted effects for these candidates under the current role instance hypotheses. Each interaction description draws its requirements and effects from the same combination of compatible role instance hypotheses. The skill also relates predicted effects to the prerequisites of subsequent actions and highlights uncertainty in role assignments and parameter estimates. The skill guides the actor to base judgments about object behavior on role instance hypotheses and interaction evidence, even when familiar names suggest different functions. The actor selects the final action from the environment's available actions, considering this information alongside its observations, task priorities, and immediate needs.

Active verification guided by the skill. A role instance hypothesis used for prediction may still lack interaction evidence, and support for its prototype assignment does not imply that all its parameter values or interactions have been verified. The skill encourages the actor to seek interaction evidence for unverified functions. Using the supplied role instance hypotheses, the actor assesses whether verification is useful for its decision and chooses which interaction, if any, to perform.

Feedback from the chosen action is used to refine role assignments and parameter estimates through the inference mechanism described above. The skill incorporates these updates and the interaction history into its guidance at the next step. In a target world, the teacher is absent, and both the actor's LLM parameters and the skill remain fixed.

4 EXPERIMENT

We evaluate RoleWorld in MARS (Tang et al., 2024b), an open-world survival benchmark built on Crafter (Hafner, 2022). Agents pursue 22 achievements in a partially observable world using text observations of nearby objects and their coordinates, the agent's status, and its inventory. MARS includes a *Default* world and seven variants that alter terrain, survival settings, and task dependencies, individually or in combination, while retaining Crafter's objects and achievements. Changes such as hostile cows and stone yielding diamonds make object names unreliable indicators of behavior, testing adaptation to changed object functions. We compare RoleWorld with the MARS implementation of ReAct (Yao et al., 2023; Tang et al., 2024b), which reasons before each action; Induction from Reflection (IFR) (Tang et al., 2024b), which induces environment rules from trajectory reflections; and three world-model-based approaches: PoE-World (Piriyakulkij et al., 2025), WALL-E (Zhou et al., 2025), and WorldCoder (Tang et al., 2024a).

Evaluation protocol. Table 1 distinguishes three stages: *Source learning* acquires knowledge in Default before transfer; *Target learning* acquires knowledge in each target world before evaluation; and *Online refinement* updates knowledge using feedback within each evaluation episode. We use two complementary comparisons to evaluate transfer. First, IFR, PoE-World, and WALL-E learn separately in each target world for five episodes before evaluation. This comparison tests whether RoleWorld can transfer knowledge from Default to achieve better task performance than baselines that learn in the target world. Second, WorldCoder and RoleWorld autonomously learn through interaction in Default, each with a source-learning budget of at most five episodes. Both enter target-world evaluation without target learning, providing a direct comparison of transfer performance. ReAct uses neither pre-evaluation learning stage.

All LLM calls during source learning use GPT-5.6 Sol (OpenAI, 2026). WorldCoder constructs an executable world model. For RoleWorld, the teacher constructs role prototypes and transition rules, infers initial role assignments and parameter values, and develops the program that organizes role information for the actor. During target-world evaluation, WorldCoder refines its model code online, whereas RoleWorld refines only role assignments and environment-specific parameter estimates, keeping the prototypes, transition rules, and skill fixed. IFR continues to induce rules, PoE-World refines its experts and weights, and WALL-E refines symbolic knowledge.

We use Qwen3-8B (Yang et al., 2025) and GPT-5.6 Luna (OpenAI, 2026) as target-world backbones. In each configuration, the designated backbone handles all LLM calls during target learning and evaluation, including action selection and knowledge refinement. Episodes in all three stages are limited to 1,000 environment steps. Learned knowledge is retained across episodes within source learning and target learning. Each evaluation episode starts from the same knowledge state obtained before evaluation for that method and world; interaction history and refinements made during one evaluation episode are not carried into another.

We follow MARS (Tang et al., 2024b) in computing episode reward and score (%). For each target world, we evaluate with nine random seeds and report the mean and standard deviation. The *All* entries report the arithmetic mean of the seven per-world means for each metric.

As shown in Table 2, RoleWorld achieves the highest average reward across the seven variants with both backbones, demonstrating its effectiveness in supporting agent decision making across environments with altered dynamics. The advantage over WorldCoder under the same transfer protocol

Table 1: Evaluation stages (✓: yes; ×: no). Source learning: at most five episodes in Default; target learning: five per target world before evaluation; online refinement: within each evaluation episode.

Stage	ReAct	IFR	PoE-World	WALL-E	WorldCoder	RoleWorld
Source learning	×	×	×	×	✓	✓
Target learning	×	✓	✓	✓	×	×
Online refinement	×	✓	✓	✓	✓	✓

Table 2: Reward and score (%) on the seven MARS variants under the protocols in Table 1. Column groups indicate the backbone used for all target-world LLM calls.

Metrics	Mod. Type	Qwen3-8B						GPT-5.6 Luna					
		ReAct	IFR	PoE-World	WALL-E	World Coder	Ours	ReAct	IFR	PoE-World	WALL-E	World Coder	Ours
Reward	Terrain	2.8 ± 1.6	2.0 ± 0.9	0.4 ± 1.2	2.0 ± 2.2	5.3 ± 1.7	$\mathbf{9.4 \pm 3.7}$	4.0 ± 2.5	10.8 ± 1.2	0.8 ± 0.7	8.2 ± 3.1	5.8 ± 1.7	$\mathbf{11.4 \pm 1.1}$
	Survival	2.0 ± 1.2	3.0 ± 2.0	0.0 ± 1.1	1.3 ± 0.8	2.3 ± 1.4	$\mathbf{8.0 \pm 2.6}$	4.9 ± 2.5	9.3 ± 1.5	1.7 ± 2.4	10.2 ± 4.0	3.1 ± 1.7	$\mathbf{10.8 \pm 2.4}$
	Task Dep.	1.9 ± 1.7	2.2 ± 0.7	1.5 ± 1.6	2.2 ± 1.4	0.8 ± 0.8	$\mathbf{6.8 \pm 0.8}$	1.7 ± 1.1	7.7 ± 1.0	1.4 ± 0.8	4.8 ± 2.5	1.2 ± 0.6	$\mathbf{8.5 \pm 1.5}$
	Terr. Surv.	1.4 ± 1.0	1.7 ± 0.8	0.1 ± 0.7	1.7 ± 3.2	4.4 ± 1.6	$\mathbf{8.1 \pm 3.3}$	4.1 ± 1.5	7.8 ± 0.7	2.1 ± 0.9	5.7 ± 2.4	4.3 ± 1.4	$\mathbf{11.0 \pm 1.7}$
	Terr. Task.	2.1 ± 1.2	3.0 ± 0.7	1.4 ± 1.1	1.7 ± 1.4	3.7 ± 1.5	$\mathbf{6.2 \pm 2.1}$	3.8 ± 1.9	6.4 ± 0.8	3.8 ± 0.8	3.8 ± 1.3	4.2 ± 1.4	$\mathbf{7.0 \pm 1.0}$
	Surv. Task.	1.4 ± 2.0	3.2 ± 1.5	-0.6 ± 0.5	2.0 ± 1.2	1.3 ± 0.6	$\mathbf{4.3 \pm 0.8}$	1.8 ± 1.6	3.3 ± 1.6	0.5 ± 1.7	4.3 ± 2.2	0.7 ± 0.8	$\mathbf{4.7 \pm 0.8}$
	All Three	-0.2 ± 0.6	-0.1 ± 0.6	-0.5 ± 0.5	-0.2 ± 0.8	-0.0 ± 0.6	$\mathbf{3.0 \pm 1.5}$	0.6 ± 0.9	2.5 ± 1.5	0.3 ± 1.1	0.9 ± 1.5	-0.4 ± 0.5	$\mathbf{3.3 \pm 1.5}$
	All	1.6	2.1	0.4	1.5	2.5	**6.5**	3.0	6.8	1.5	5.4	2.7	**8.1**
Score (%)	Terrain	1.6 ± 0.5	1.5 ± 1.1	0.6 ± 0.4	1.7 ± 1.2	4.3 ± 1.2	$\mathbf{16.6 \pm 9.4}$	3.3 ± 1.0	13.1 ± 2.3	0.6 ± 0.1	12.0 ± 1.2	5.5 ± 0.5	$\mathbf{19.2 \pm 5.1}$
	Survival	1.5 ± 0.2	2.6 ± 1.8	0.5 ± 0.4	1.0 ± 0.2	1.5 ± 0.3	$\mathbf{10.6 \pm 5.6}$	4.6 ± 0.7	14.7 ± 3.5	1.7 ± 1.3	16.3 ± 0.8	2.3 ± 0.4	$\mathbf{19.1 \pm 7.4}$
	Task Dep.	1.5 ± 0.6	1.5 ± 0.4	1.2 ± 0.6	1.6 ± 0.4	0.6 ± 0.2	$\mathbf{4.9 \pm 1.3}$	1.0 ± 0.4	7.1 ± 2.0	0.8 ± 0.3	4.0 ± 1.3	0.7 ± 0.1	$\mathbf{9.9 \pm 4.8}$
	Terr. Surv.	0.8 ± 0.2	1.1 ± 0.6	0.5 ± 0.2	1.0 ± 0.2	3.4 ± 1.0	$\mathbf{11.0 \pm 4.8}$	2.3 ± 0.5	6.8 ± 0.8	1.4 ± 0.3	5.2 ± 0.3	2.8 ± 1.1	$\mathbf{17.6 \pm 7.6}$
	Terr. Task.	1.1 ± 0.4	1.8 ± 0.6	1.0 ± 0.2	1.2 ± 0.3	2.5 ± 0.1	$\mathbf{5.8 \pm 2.9}$	2.6 ± 1.0	5.0 ± 0.8	2.3 ± 0.4	3.1 ± 0.8	2.7 ± 0.3	$\mathbf{5.6 \pm 1.6}$
	Surv. Task.	1.4 ± 0.7	2.8 ± 1.7	0.2 ± 0.0	1.5 ± 0.3	1.0 ± 0.3	$\mathbf{4.4 \pm 1.7}$	1.5 ± 0.4	1.6 ± 0.5	0.9 ± 0.7	3.3 ± 0.5	0.6 ± 0.1	$\mathbf{4.1 \pm 2.5}$
	All Three	0.3 ± 0.1	0.4 ± 0.4	0.2 ± 0.0	0.4 ± 0.0	0.3 ± 0.1	$\mathbf{2.2 \pm 0.9}$	0.8 ± 0.4	1.6 ± 1.0	0.6 ± 0.2	1.1 ± 0.5	0.2 ± 0.1	$\mathbf{2.4 \pm 0.5}$
	All	1.2	1.7	0.6	1.2	1.9	**7.9**	2.3	7.1	1.2	6.4	2.1	**11.1**

suggests that our role-based symbolic world model transfers more effectively across environments with changed object behavior.

4.1 Ablation

We evaluate four ablations with Qwen3-8B under the same transfer protocol: *w/o Skill* retains online adaptation but removes the skill's information organization and decision/verification guidance; *w/o Adaptation* freezes initial role assignments and parameter estimates; *Fixed Role Assignments* permits parameter refinement but freezes object–role assignments; and *w/o Active Verification* removes explicit verification guidance while retaining other decision support and online adaptation. Without the skill, the actor receives action requirements, predicted effects, and uncertainty from the highest-probability compatible combination of role instance hypotheses. We also compare *Direct LLM*, which acts without a separate reasoning step or role guidance, and *RoleWorld (Symbolic Planner)*, which replaces the actor with a hand-designed symbolic planner while retaining the same role representation and adaptation procedure. Implementation details appear in Appendices A.3 and A.4.

As shown in Table 3, the transferred role knowledge and skill provide a useful starting point for target-world decisions: *w/o Adaptation* achieves an average reward of 4.9, compared with 1.0 for *Direct LLM*. Online adaptation further increases average reward to 6.5, although freezing the source role estimates remains competitive in several variants. Thus, adaptation builds on a useful transferred prior, with benefits that vary across environments.

Although *w/o Adaptation* and *Fixed Role Assignments* both average 4.9 reward, per-world results suggest that parameter refinement and role reassignment are complementary, with neither consistently more important. On Terr. Surv., parameter refinement alone raises reward from 6.2 to 7.8, close to the full model's 8.1. On Task Dep., it raises reward only from 0.9 to 1.4, versus 6.8 with role reassignment. On Terr. Task., reward rises from 4.3 to 5.4 with parameter refinement and further to 6.2 with role reassignment. These patterns are consistent with mismatches in parameter values, functional bindings, or both. On All Three, parameter refinement accounts for most of the gain (0.2 to 2.5, versus 3.0 for the full model), suggesting that role reassignment's relative benefit does not necessarily increase as more environment factors change.

Realizing these benefits also requires guidance for using and testing role knowledge. Removing the skill reduces average reward to 1.2, close to *Direct LLM*. Retaining its decision support with-

Table 3: Ablation results and decision-policy comparison on the seven MARS variants. LLM-based variants use Qwen3-8B. All is the arithmetic mean of the per-world means for each metric. Bold indicates the highest mean in each column within each metric.

Metrics	Method	Terrain	Survival	Task Dep.	Terr. Surv.	Terr. Task.	Surv. Task.	All Three	All
Reward	Direct LLM	0.2 ± 1.0	2.0 ± 1.7	0.9 ± 1.4	1.3 ± 1.2	2.0 ± 1.1	0.8 ± 1.4	0.0 ± 1.1	1.0
	w/o Skill	0.9 ± 0.8	2.2 ± 1.6	1.3 ± 1.0	2.3 ± 1.5	1.1 ± 0.9	1.2 ± 0.7	-0.3 ± 0.5	1.2
	w/o Adaptation	9.7 ± 1.8	8.4 ± 3.1	0.9 ± 0.4	6.2 ± 1.9	4.3 ± 2.4	4.8 ± 2.6	0.2 ± 1.4	4.9
	Fixed Role Assignments	6.8 ± 3.1	6.5 ± 3.3	1.4 ± 1.3	7.8 ± 4.1	5.4 ± 1.4	3.7 ± 3.9	2.5 ± 1.7	4.9
	w/o Active Verification	8.0 ± 1.9	6.0 ± 4.3	1.2 ± 0.7	7.7 ± 3.1	1.7 ± 1.5	0.9 ± 0.8	-0.5 ± 0.7	3.6
	RoleWorld (full)	9.4 ± 3.7	8.0 ± 2.6	6.8 ± 0.8	8.1 ± 3.3	6.2 ± 2.1	4.3 ± 0.8	3.0 ± 1.5	6.5
	RoleWorld (Symbolic Planner)	$\mathbf{13.5 \pm 0.6}$	$\mathbf{14.9 \pm 1.5}$	$\mathbf{8.3 \pm 1.8}$	$\mathbf{12.4 \pm 1.6}$	$\mathbf{14.2 \pm 1.7}$	$\mathbf{15.8 \pm 1.9}$	$\mathbf{6.5 \pm 2.6}$	**12.2**
Score (%)	Direct LLM	0.5 ± 0.3	1.9 ± 0.7	1.0 ± 0.4	1.0 ± 0.5	1.5 ± 0.0	1.0 ± 0.3	0.5 ± 0.4	1.0
	w/o Skill	0.8 ± 0.5	1.4 ± 0.6	0.9 ± 0.5	2.1 ± 1.7	1.0 ± 0.4	1.0 ± 0.2	0.2 ± 0.2	1.0
	w/o Adaptation	12.6 ± 3.8	12.1 ± 5.9	0.8 ± 0.3	6.2 ± 3.0	2.8 ± 2.1	5.4 ± 4.9	0.6 ± 0.6	5.8
	Fixed Role Assignments	7.7 ± 1.7	8.4 ± 6.0	1.2 ± 0.3	12.3 ± 3.7	3.6 ± 0.4	5.3 ± 5.1	1.8 ± 0.8	5.7
	w/o Active Verification	9.8 ± 3.8	10.0 ± 4.4	0.9 ± 0.2	9.8 ± 0.9	1.3 ± 0.3	0.7 ± 0.2	0.2 ± 0.1	4.7
	RoleWorld (full)	16.6 ± 9.4	10.6 ± 5.6	4.9 ± 1.3	11.0 ± 4.8	5.8 ± 2.9	4.4 ± 1.7	2.2 ± 0.9	7.9
	RoleWorld (Symbolic Planner)	$\mathbf{21.3 \pm 0.7}$	$\mathbf{39.5 \pm 15.2}$	$\mathbf{7.7 \pm 1.4}$	$\mathbf{20.6 \pm 6.6}$	$\mathbf{37.0 \pm 12.6}$	$\mathbf{46.5 \pm 17.9}$	$\mathbf{8.3 \pm 5.4}$	**25.8**

out active verification raises reward to 3.6, but this remains below *w/o Adaptation*. Together with the full model's 6.5, these results support the joint use of online adaptation and active verification. Verification provides larger reward gains in variants with task-dependency changes, consistent with uncertain assumptions about role instances and interaction requirements affecting downstream decisions through prerequisite chains.

The symbolic planner further illustrates the importance of how role knowledge is used. With the same role representation and adaptation procedure, it increases average reward from 6.5 to 12.2 and score from 7.9% to 25.8%. Together with the skill ablation, this suggests substantial room to improve the decision policy, whose actions also determine the feedback available for adaptation.

4.2 Controlled Validation of Role Factorization

We test whether interaction feedback can correct corrupted role estimates while keeping the environment, role prototypes, and transition rules fixed.

In *Default*, we use a reference prior validated against environment action effects in controlled tests (Appendix A.6). It specifies 30 object–role assignments covering traversal, harvesting, crafting, and placement, with 110 role parameter fields. We corrupt the two separately: *role parameter corruption* replaces 30%, 50%, 70%, or 90% of the fields with incorrect values drawn from their declared domains while retaining all role assignments; *role assignment corruption* replaces the role types in the same proportions of the 30 assignments with randomly sampled incorrect types from the prototype library. We evaluate each condition after the first 100 and 300 active interactions from actor-generated trajectories, reporting means over six random seeds.

Role-type accuracy is the percentage of the 30 reference object–role assignments whose inferred role types match the reference, irrespective of their parameter values. *Role parameter accuracy* is the percentage of the 110 fields matching the reference values, counting missing fields and those associated with incorrect role assignments as incorrect. *One-step prediction discrepancy* is the total variation distance between predicted and ground-truth distributions of direct action effects, averaged over 213 fixed, independent state–action pairs and reported as a percentage. The pairs cover satisfied and unsatisfied preconditions. Ground-truth distributions are obtained by executing the environment's action logic and enumerating its stochastic branches, excluding passive survival and NPC updates. If a required role assignment or prediction is unavailable, the discrepancy is set to 100%. All conditions share this evaluation set, and evaluation does not update the model.

Under role assignment corruption, role-type accuracy reaches 100% by 100 active interactions and remains at 100% at 300 interactions. Thus, role types are recovered within this interaction budget, while recovering the associated environment-specific parameters requires additional evidence. Figure 2 therefore focuses on role parameter accuracy and prediction discrepancy to characterize the remaining errors in role estimates and their predictive consequences.

Figure 2 shows greater recovery of role estimates under role assignment corruption than under parameter corruption at both interaction budgets and across all tested corruption levels. After 300

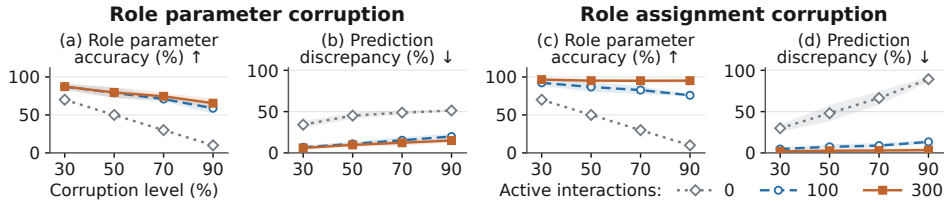

Figure 2: Controlled validation of role factorization. Role assignments and role parameter values are corrupted separately, while the environment, role prototypes, and transition rules remain fixed. Shaded bands indicate one standard deviation; see Appendix A.5 for details.

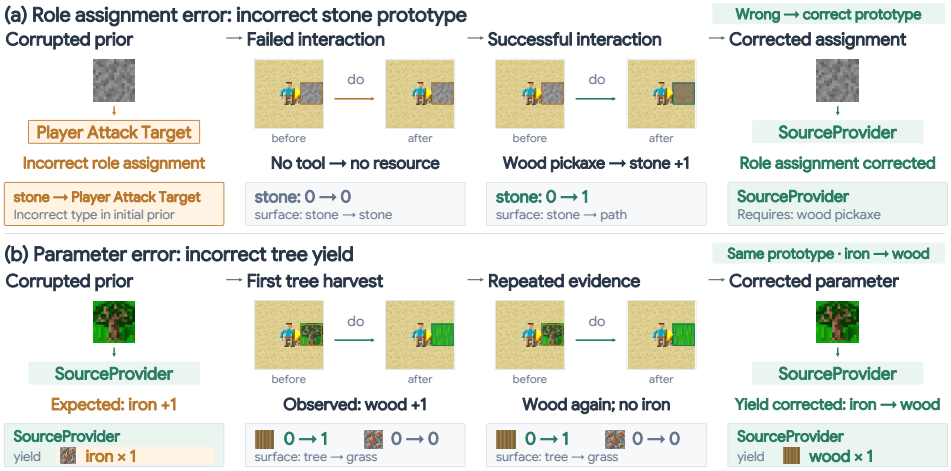

Figure 3: Correction of role estimates with fixed role prototypes and transition rules: (a) correcting a role assignment; (b) correcting a role parameter.

interactions, assignment corruption yields approximately 95–97% role parameter accuracy and prediction discrepancy below 4% across all corruption levels, suggesting limited sensitivity to initial role assignment errors within this controlled setting. A plausible explanation is that incorrect role types can produce qualitatively discriminative prediction mismatches, whereas identifying individual parameter values may require more specific interaction conditions.

Under 90% parameter corruption, 300 interactions raise parameter accuracy from 10% to approximately 65.5%, while reducing prediction discrepancy from approximately 51% to 15%. Thus, exact recovery of role assignments and parameters is not a prerequisite for recovering useful predictions on the evaluated interactions. This suggests that adaptation can target role estimates that predict the relevant interaction dynamics without fully reconstructing the environment's symbolic model.

Figure 3 qualitatively illustrates role estimate correction. For stone in Figure 3(a), failure without a tool followed by successful harvesting with a wooden pickaxe supports replacing the illustrative *Player Attack Target* assignment with *SourceProvider*. For tree in (b), repeated observations of wood rather than iron correct the resource-yield parameter while preserving its *SourceProvider* assignment.

5 Conclusion

We introduced RoleWorld, which transfers world knowledge through functional role abstractions, reusing role prototypes and transition rules while refining environment-specific role assignments and parameter estimates through interaction feedback. Across both backbones in MARS, transferring from Default yields higher average reward than the evaluated baselines under the reported configurations. Controlled interventions further show that useful predictions can be recovered by correcting role estimates while keeping shared prototypes and rules fixed.

AI USE STATEMENT

Generative AI tools were used to assist with the writing and editing of this manuscript and for limited code-related assistance, such as debugging. All AI-assisted content and code were reviewed and verified by the authors, who take full responsibility for the final work.

ETHICS STATEMENT

Our work does not involve human subjects, personally identifiable information, or the collection of sensitive data. We use only publicly available datasets in accordance with their intended research purposes. We do not identify any specific ethical concerns beyond those generally associated with machine learning research. Nevertheless, as with other machine learning methods, care should be taken when applying the proposed approach to sensitive or high-stakes real-world applications.

REPRODUCIBILITY STATEMENT

We provide the methodological details and experimental setup necessary to reproduce our results in the main paper and the appendix, including implementation details, training configurations, and evaluation protocols. Additional experimental details and analyses are also provided in the appendix. Our code and data will be open-sourced upon formal release to further facilitate reproducibility.

REFERENCES

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

Juntao Cheng, Jingkai Wang, Yijun Shen, Xiansheng Chen, and Zhiwei Yu. Beyond instance slots: Semantically rich world models for physical interaction planning. *CoRR*, abs/2608.22294, 2026. doi: 10.48550/ARXIV.2608.22294. URL https://doi.org/10.48550/arXiv.2608.22294.

David Courtis, Wenhao Li, and Scott Sanner. Opine-world: Programmatic world modeling with ontology-error-prioritized interactive exploration for ARC-AGI-3. *CoRR*, abs/2607.01531, 2026. doi: 10.48550/ARXIV.2607.01531. URL https://doi.org/10.48550/arXiv.2607.01531.

Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis (eds.), *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pp. 240–247. ACM, 2008. doi: 10.1145/1390156.1390187. URL https://doi.org/10.1145/1390156.1390187.

Finale Doshi-Velez and George Dimitri Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In Subbarao Kambhampati (ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 1432–1440. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/206.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse control tasks through world models. *Nat.*, 640(8059):647–653, 2025. doi: 10.1038/S41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 8154–8173. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.507. URL https://doi.org/10.18653/v1/2023.emnlp-main.507.

Steven James, Benjamin Rosman, and George Konidaris. Autonomous learning of object-centric abstractions for high-level planning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=rrWeE9ZDw_.

Ken Kansky, Tom Silver, David A. Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, D. Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1809–1818. PMLR, 2017. URL http://proceedings.mlr.press/v70/kansky17a.html.

Zaid Khan, Archiki Prasad, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. One life to learn: Inferring symbolic world models for stochastic environments from unguided exploration. *CoRR*, abs/2510.12088, 2025. doi: 10.48550/ARXIV.2510.12088. URL https://doi.org/10.48550/arXiv.2510.12088.

Khimya Khetarpal, Zafarali Ahmed, Gheorghe Comanici, David Abel, and Doina Precup. What can I do here? A theory of affordances in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5243–5253. PMLR, 2020. URL http://proceedings.mlr.press/v119/khetarpal20a.html.

Khimya Khetarpal, Gheorghe Comanici, Jonathan Richens, Jeremy Shar, Fei Xia, Laurent Orseau, Aleksandra Faust, and Doina Precup. Affordances enable partial world modeling with llms. *CoRR*, abs/2602.10390, 2026. doi: 10.48550/ARXIV.2602.10390. URL https://doi.org/10.48550/arXiv.2602.10390.

Guillaume Levy, Cédric Colas, Pierre-Yves Oudeyer, Thomas Carta, and Clément Romac. Worldllm: Improving llms' world modeling using curiosity-driven theory-making. *CoRR*, abs/2506.06725, 2025. doi: 10.48550/ARXIV.2506.06725. URL https://doi.org/10.48550/arXiv.2506.06725.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled architecture for autonomous LLM agents. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 31186–31261. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/liu24ab.html.

Ofir Marom and Benjamin Rosman. Transferable dynamics models for efficient object-oriented reinforcement learning. *Artif. Intell.*, 329:104079, 2024. doi: 10.1016/J.ARTINT.2024.104079. URL https://doi.org/10.1016/j.artint.2024.104079.

Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Trans. Robotics*, 24(1):15–26, 2008. doi: 10.1109/TRO.2007.914848. URL https://doi.org/10.1109/TRO.2007.914848.

OpenAI. GPT-5.6 System Card, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Top Piriyakulkij, Yichao Liang, Hao Tang, Adrian Weller, Marta Kryven, and Kevin Ellis. Poe-world: Compositional world modeling with products of programmatic experts. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/262dd62fd1bbb30d6a6b4d578f5e65ff-Abstract-Conference.html.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609, 2020. doi: 10.1038/S41586-020-03051-4. URL https://doi.org/10.1038/s41586-020-03051-4.

Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/820c61a0cd419163ccbd2c33b268816e-Abstract-Conference.html.

Xiaojuan Tang, Jiaqi Li, Yitao Liang, Song-Chun Zhu, Muhan Zhang, and Zilong Zheng. Mars: Situated inductive reasoning in an open-world environment. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/1fb6d0b52f5e41b11392841a66dbfe7d-Abstract-Datasets_and_Benchmarks_Track.html.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, 2009. doi: 10.5555/1577069.1755839. URL https://dl.acm.org/doi/10.5555/1577069.1755839.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-E: world alignment by neurosymbolic learning improves world model-based

LLM agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/5e772a13ccba5255331240dcd99aa38b-Abstract-Conference.html.

A Appendix

A.1 Offline World Model Construction Details

Inputs and predefined representation. The teacher constructs role prototypes and transition rules from source-world trajectories and additional interaction evidence. The role representation, DSL event vocabulary, condition and effect operators, and interpreter are provided in advance. This includes the execution semantics and event scheduling used to compose rules. Within this representation, the teacher selects the functional decomposition, declares parameters and state variables, and composes conditions and effects. These structural choices are made by the teacher; the interpreter executes the resulting rules. Environment-interaction tools allow the teacher to restore states, set controlled conditions, and execute actions. Evidence for induction consists of public observations before and after each action, together with the recorded intervention conditions.

Preparing interaction evidence. The teacher compares observations before and after each action to identify candidate effects, retaining unsuccessful attempts and interactions with no observable change. For controlled interactions, the observation before the action is recorded after setting the experimental conditions, so changes made by the tools are excluded from the action's effects. Comparisons with control conditions and no-operation actions help distinguish action effects from background changes. An observed difference alone does not establish which mechanism caused it. Likewise, an unchanged observation may reflect unmet conditions or a stochastic outcome, so it does not by itself eliminate a candidate function.

Jointly proposing prototypes and transition rules. For each candidate mechanism, the teacher selects a triggering event event from the predefined vocabulary and specifies participant variables and their required role prototypes in P_τ. Successful and unsuccessful interactions constrain the applicability condition g_τ. The effects F_τ specify the resulting state changes and interaction outcomes, including dependencies among effects and conditions under which they occur. Effects that share a stochastic outcome are represented jointly, preserving which changes occur together. Outcome probabilities are estimated from repeated interactions under the corresponding conditions; observations of only successful outcomes support a deterministic modeling assumption, rather than proving that the underlying probability is one.

The teacher synchronizes these rules with the associated prototypes. Parameters describe interaction requirements and effects that are fixed for an object within an environment, with values in Θ_ρ; state variables describe properties that evolve through interaction, with values in $\mathcal{X}_\rho$. Candidate parameter values are supported by the interaction evidence. Each declared parameter or state variable must be referenced by at least one transition condition or effect. Thus, constructing a prototype includes determining the variables required by its rules, and revising a rule may require corresponding changes to the prototype.

Choosing the scope of a revision. The teacher first checks whether a discrepancy can be explained by parameter values within the current prototype and transition structure. Assigning an existing prototype to another object also requires no new functional definition. If the same function instead needs an additional applicability condition or a revised outcome structure, the teacher modifies its transition rules and any required parameter or state declarations. A new or split prototype is introduced when an independent function requires a separate description of its events, participants, conditions, or effects; the associated transition rules are constructed at the same time. A failed parameter revision therefore does not automatically imply a new prototype. Functions that coexist independently on the same object remain separate, with $\mathcal{C}_\rho$ specifying which prototypes may be instantiated together.

Parameterizing and merging candidates. The teacher merges candidates when their triggering events, participant structure, and condition–effect structure can be shared, including the temporal relationships and dependencies among outcomes. Object-specific constants become parameters, supported values enter the parameter domains, and each object's complete parameter combination is retained in its source-world role instance. The associated transition rules are merged at the same time: prototype references, participant requirements, and parameter or state references are updated,

and superseded rules are removed to avoid executing duplicate effects. Similar outputs or coexistence on the same object are insufficient grounds for a merge. Distinct conditions or temporal relationships must remain explicit as conditional branches or separate functions.

Executing rules and checking revisions. The interpreter executes candidate rules using the observed state before the action and the parameter values of the corresponding role instances, composing applicable rules according to the predefined DSL semantics. Predictions are compared with the complete observable effects covered by the candidate model, including quantities or fields expected to remain unchanged. Deterministic predictions are checked for agreement with these effects. For stochastic predictions, checking whether an observation lies in the predicted support is distinct from comparing predicted probabilities with empirical outcome frequencies. Missing effects, extra effects, incorrect conditions, and incorrect dependencies among outcomes provide counterexamples for revising the model.

After a structural revision, the teacher checks the affected rules against accumulated successful and unsuccessful interactions. Parameterization and merging are also checked for preservation of the predicted outcome distributions on the recorded inputs. This check evaluates whether the revised representation preserves the candidate mechanisms on those inputs; it supplies no independent evidence for their correctness on unobserved inputs. Additional controlled interactions can distinguish competing explanations. The model is frozen before checking it on held-out interactions, which are not used to select its structure or estimate its parameters.

Retained model and transfer. The retained prototypes define reusable functions and their parameter and state spaces; the transition rules give their executable behavior. Source-world role assignments and parameter values are retained separately as the initial role prior, using the formats in Appendix A.2. In the target world, prototypes and transition rules remain fixed while role assignments and parameter estimates are refined online. The construction establishes an evidence-supported representation within the observed conditions and the supplied DSL; finite trajectories need not uniquely determine the functional decomposition.

A.2 Role Prototype, Transition, and Prior Formats

We provide example formats for role prototypes, transition rules, and initial role priors below. The schemas below show their field names and nesting, with angle-bracketed placeholders describing the corresponding contents. Each array may contain multiple entries of the displayed form.

Role prototypes. Each role prototype defines a reusable function, the compatible prototypes that may be instantiated on the same object, and the parameter and state spaces of its instances. Parameters describe properties that are fixed for a given object within an environment; state variables track quantities that can change during interaction.

```
{
  "role_prototypes": [
    {
      "prototype": {
        "id": "<Unique prototype identifier>",
        "name": "<Human-readable prototype name>",
        "definition": "<Reusable behavioral function>",
        "compatible_prototype": ["<Compatible prototype ID>"],
        "decision_invariants": ["<Criteria for role inference>"],
        "variables": {
          "parameters": {
            "<parameter_name>": {
              "type": "<Parameter data type>",
              "domain": <Allowed values or numeric interval>
            }
          },
          "state": {
```

```
            "<state_name>": {
              "type": "<State variable data type>"
            }
          }
        }
      }
    }
  ]
}
```

A parameter's `domain` is specified by an explicit list of admissible `values` or an `interval` of numeric bounds. The `parameters` and `state` mappings may be empty when the prototype declares no variables of that kind.

Transition rules. Each rule specifies a triggering event, participant aliases with their required role prototypes, an applicability condition, and effects. Participant aliases bind concrete role instances when evaluating the DSL condition and effects.

```
{
  "transitions": [
    {
      "id": "<Unique transition identifier>",
      "role": "<Prototype ID associated with this rule>",
      "event": "<Event triggering rule evaluation>",
      "participants": {
        "<participant_alias>": "<Required prototype ID>"
      },
      "when": <DSL condition for applying the rule>,
      "effects": [<DSL operations for state changes and outcomes>]
    }
  ]
}
```

Each effect operation specifies `op` (the operation), `observable` (whether it contributes observable evidence), and `evidence_owner` (the participant to which the evidence is attributed), together with operation-specific arguments.

Initial role priors. The prior groups initial object–role assignments by prototype identifier and supplies environment-specific parameter values for each binding. These values instantiate the parameter declarations in the corresponding prototype.

```
{
  "role_bindings": [
    {
      "role_id": "<Assigned prototype identifier>",
      "instance_bindings": [
        {
          "observed_type": "<Object type assigned this prototype>",
          "parameters": {
            "<parameter_name>": <Initial parameter value>
          }
        }
      ]
    }
```

```
    ]
}
```

The prototypes and transition rules remain fixed in the target world. The prior initializes role assignments and parameter estimates, which are revised online from interaction feedback; current state values are supplied separately during interaction.

A.3 Ablation Details

Shared settings. The four component ablations in Table 3 use Qwen3-8B in the seven non-Default MARS worlds, with the same source-world prior, role prototypes, and executable transition rules as the full method. Each episode resets role beliefs and interaction history. The prototypes, transition rules, and LLM parameters remain fixed throughout evaluation. All variants use the same observation format, up to ten recent action–observation turns, and the same action set and decoding settings. Episodes are limited to 1,000 environment steps, including an initial `noop`; an episode reaching this limit therefore contains at most 999 actor decisions.

w/o Skill. We retain online role inference and refinement and query the same transition model for available actions, including hypothetical `do` interactions after approaching and facing other visible object types. For each query, a fixed filter selects the compatible combination of role instance hypotheses with the highest probability mass. It presents preconditions, effect quantities and conditional probabilities, current outcome predictions, verification status, and unknowns. Hypothesis probability mass and action-effect probabilities are displayed separately. Rules with mismatched action conditions or no effects are omitted, together with internal parameter tables and repeated evidence. We remove both the program-generated decision guidance and the fixed textual skill instructions. The remaining presentation does not rank information by task or history, resolve prerequisite chains, or provide navigation and retry suggestions. The actor still receives its observation and action-feedback history, and its chosen actions continue to supply evidence for online adaptation.

w/o Adaptation. At every decision, we construct the role-query interface from a fresh copy of the initial source-world prior and populate it with the current observed state. Interaction evidence does not revise role assignments, parameter hypotheses, their weights, or verification status. Prior assignments retain their initial unverified status. The full skill, including its active-verification guidance, remains available. Current observations, inventory, status, observed movement, and action history are still supplied, allowing the skill to respond to changed conditions and previous failures. Thus, the frozen component is the persistent role knowledge; the decision input continues to reflect the ongoing episode.

Fixed Role Assignments. We fix each object's assigned prototype set to its initial assignments and disable the addition, removal, and replacement of assigned prototypes. Interaction feedback still refines parameter values and hypothesis weights within the assigned prototypes. For example, the model can revise the yield or requirements of an assigned resource-provider role instance. In this implementation, role verification status and event-support flags also remain at their initial values. The full skill, current observations, action history, and verification-related information are retained. This condition therefore permits parameter refinement within the prior assignments while keeping the functional assignments and their verification status fixed.

w/o Active Verification. We retain the skill's ordinary decision support and full online adaptation but remove its explicit support for active verification. In the full method, the skill provides role instance hypotheses and their probabilities, verification status, interaction evidence, and action requirements and predicted effects. Using this information alongside its observations and task priorities, the actor decides whether verification is useful and which interaction, if any, to perform. Ordinary action descriptions in the ablation use the highest-probability compatible candidate. Prerequisite planning, exploration to find missing materials or stations, risk indications, and failure history are retained. The actor remains free to choose any available action, including an uncertain interaction, and feedback from its actions continues to update role assignments and parameter esti-

mates. This ablation evaluates the contribution of the skill's support for the actor's own verification decisions.

A.4 Symbolic Planner Settings

For RoleWorld (Symbolic Planner), we replace the LLM actor with a hand-designed symbolic planner that operates on inferred role instances. The role prototypes, executable transition rules, and online role inference and refinement procedure are retained. We evaluate this variant on the same seven MARS variants, with reward and score reported in Table 3.

Role-based action representation. At each step, the planner associates observed objects and remembered terrain with their current role assignments. For ordinary planning, each assigned prototype is instantiated with its highest-probability parameter hypothesis. The shared transition rules then provide action descriptions consisting of an action, its preconditions, and its predicted effects. These include resource requirements, proximity to workstations, inventory and status changes, object creation, and achievement progress. The planner interprets the DSL condition and effect vocabulary; prototype identifiers select the corresponding inferred role instances for each object. Thus, concrete object functions and interaction requirements enter planning through the current role estimates.

Goal selection and prerequisite planning. A fixed goal scheduler prioritizes survival and recovery, followed by unfinished achievements. The planner matches a desired outcome to a role-instantiated transition effect and recursively resolves unmet prerequisites. For example, to obtain a resource, it finds an interaction predicted to increase that resource, then plans how to acquire any required tools or establish access to a required workstation. Each dependency is read from the transition conditions and linked to another action through its predicted effects. Resource and achievement plans are compared using action cost and the predicted probability of the desired effect, while a feasible ongoing achievement target is retained. Navigation searches over position and facing on a map accumulated from observations, using role-derived traversability and risk estimates from inferred role instances and interaction history.

Active verification and closed-loop execution. When ordinary goal plans are blocked, the planner can test goal-relevant role instance hypotheses or unverified interaction prerequisites. It compares candidate predictions of observable outcomes and ranks informative tests by expected information gain per action cost, subject to survival constraints. When no goal or verification plan is available, it explores the frontier of the observed map. The planner executes only the first action of each plan, records its prediction before execution, and compares it with the resulting feedback. The same interaction feedback updates role assignments and parameter estimates through the online inference procedure. The next decision uses the refreshed role information to compute a new plan. Throughout this process, the role prototypes and transition rules remain fixed, while revised role estimates can change both the selected objects and the prerequisite chains used to reach a goal.

A.5 Controlled Validation Details

Reference and corruption settings. The experiment covers 30 object–role assignments in Default: 13 for traversal, seven for harvesting, six for crafting, and four for placement. Together they contain 110 parameter fields. The reference prior is validated in Appendix A.6; the environment, role prototypes, and transition rules remain fixed during adaptation.

For parameter corruption, the four levels alter 33, 55, 77, or 99 fields while retaining all bindings. Fields are sampled without replacement from the 104 fields that admit an incorrect value in their declared domains, including action-valued fields; the six fields with singleton domains remain unchanged. Each selected field receives a uniformly sampled incorrect value. The role assignment corruption protocol selects 9, 15, 21, or 27 assignments. In each selected assignment, the correct role type is replaced by a randomly sampled incorrect type from the prototype library, and the replacement prototype's parameters are sampled independently from their respective declared domains. Unselected assignments retain their reference role types and parameter values. For each of six seeds, a random ordering of eligible fields or assignments defines nested corruption sets, so a

higher level includes the changes at lower levels. Each condition starts from its own corrupted copy of the reference prior.

Active-interaction budget and evidence. We count active interactions by the agent's verification attempts. Each *active interaction* consists of an action selected by the agent to test an uncertain role assignment, parameter value, or interaction prerequisite, together with its observed outcome. Actions used only for navigation or routine resource acquisition do not contribute to this count. A movement action selected to verify traversability is counted, and both successful and unsuccessful verification attempts are retained.

We let the agent continue interacting until the required number of active verification attempts has been reached, and extract the observation–action–observation records from the first 100 or 300 attempts in chronological order. These records are processed sequentially through the same online inference and refinement procedure to evaluate how role assignments and parameter estimates change as verification feedback accumulates. The role prototypes and transition rules remain fixed.

Recovery metrics. Role parameter accuracy uses all 110 reference fields as a fixed denominator, with equal weight per field. Missing fields and all reference fields associated with a missing or incorrect role assignment count as incorrect. A list-valued parameter is evaluated as one field; order is ignored for resource requirements, resource costs, workstation requirements, and allowed surfaces. When several parameter hypotheses share the highest probability, a field counts as correct only if it matches the reference in every tied hypothesis. Role-type accuracy is evaluated over all 30 reference object–role assignments and checks whether the inferred role types match the reference, independently of parameter accuracy.

Independent prediction evaluation. All frozen models are evaluated on the same set $\mathcal{Q}$ of 213 state–action pairs: 38 traversal, 57 harvesting, 42 crafting, and 76 placement cases. These are constructed separately from the adaptation corpus and cover tool combinations, material thresholds, workstation availability, surface types, and occupancy constraints. Each case evaluates its corresponding binding using the highest-probability parameter hypothesis; ties select the first stored candidate. Auxiliary quantity constraints are held at their reference values. The compared effects comprise inventory and status increments, achievement increments, player displacement, and changes to the action's target cell. The environment distribution is obtained by executing its action logic and enumerating stochastic branches, excluding passive survival and NPC updates while retaining quantity limits and lethal-entry effects.

For a test pair q, let $P_t(y \mid q)$ denote the frozen model's distribution over direct effects after t active interactions, and let $P_{\mathrm{env}}(y \mid q)$ denote the environment distribution. We report

$$E_{\mathrm{pred}}(t) = \frac{100}{213} \sum_{q \in \mathcal{Q}} \frac{1}{2} \sum_{y} \left| P_t(y \mid q) - P_{\mathrm{env}}(y \mid q) \right| . \tag{3}$$

An unavailable binding or incomplete prediction receives a per-case error of one and remains in the denominator. All cases receive equal weight, and evaluation never updates the model. This metric measures direct effects of individual bindings under the specified test conditions.

Aggregation and reported checkpoints. Table 4 reports means and sample standard deviations across the six seeds, with standard deviations computed using a denominator of five. In Figure 2, shaded bands show the mean plus or minus one standard deviation. Prediction discrepancy is evaluated at 0, 100, and 300 active interactions. The plotted initial parameter accuracy is 100% minus the corruption percentage: this is exact for parameter corruption and a nominal reference for role assignment corruption, since reference assignments contain different numbers of fields. All post-adaptation accuracies use the actual field counts.

A.6 Environment-Transition Validation of the Reference Prior

Scope and prediction targets. We independently evaluate the uncorrupted reference prior used to construct the controlled corruption experiments. The evaluation covers 30 object–role assignments in Default: 13 traversal assignments (R0), seven harvesting assignments (R1), six crafting assignments (R2), and four placement assignments (R3), together with the auxiliary quantity constraints

Table 4: Controlled validation results (mean ± sample standard deviation across six random seeds). Column subheadings give the active-interaction budget; both metrics are reported as percentages.

Corruption type	Level (%)	Role parameter accuracy		Prediction discrepancy		
		100	300	0	100	300
Role parameters	30	87.27 ± 3.77	87.27 ± 4.23	34.19 ± 4.22	6.81 ± 2.09	6.00 ± 1.93
	50	79.09 ± 6.88	79.55 ± 7.01	45.15 ± 3.84	10.95 ± 2.83	9.68 ± 2.80
	70	70.91 ± 3.50	74.55 ± 4.78	48.79 ± 2.59	15.33 ± 3.84	12.30 ± 3.56
	90	58.79 ± 7.10	65.45 ± 5.75	51.29 ± 1.25	19.85 ± 3.53	14.98 ± 2.81
Role assignments	30	92.27 ± 3.93	96.52 ± 2.11	29.89 ± 4.26	4.69 ± 2.18	1.88 ± 1.92
	50	86.82 ± 6.50	95.15 ± 2.74	48.04 ± 9.91	7.36 ± 3.12	2.59 ± 2.02
	70	82.73 ± 5.17	95.00 ± 2.80	66.43 ± 8.21	8.84 ± 3.00	2.85 ± 2.53
	90	75.91 ± 1.38	95.15 ± 2.48	89.28 ± 3.48	13.34 ± 2.27	3.39 ± 2.69

(R12) used in their predictions. Resources and tools have a maximum quantity of nine. The R0–R3 reference parameters are constructed from the environment configuration, so agreement with those fields alone is not independent evidence of predictive performance. We instead compare predictions with effects obtained by executing the environment. The prior and transition rules remain frozen throughout evaluation, with no learning or LLM-based revision.

For each state–action pair, the predictor uses only information visible before the action. We compare the complete vector of inventory and vital-statistic increments, achievement-count increments, player displacement, changes to the action's target cell, and the player's death status, including fields expected to remain unchanged. Occupation of the target cell by the player after movement is not counted as a terrain change. Changes to other cells, full NPC states, rewards, and hidden timers are outside the prediction target.

Controlled scenarios. We evaluate 585 state–action pairs, retaining all 213 inputs from the main corruption experiment and adding condition boundaries and combinations. Test scenarios are generated from environment definitions without reading the prior under evaluation. They cover traversal directions, health levels, occupied cells and boundaries; harvesting with different tool combinations and inventory quantities; crafting with insufficient, exact, or surplus materials and different workstation combinations and distances; placement across terrain types, material quantities, and occupancy conditions; and repeated triggering of previously unlocked achievements.

Each case restores a complete snapshot from normal environment initialization, then sets the required terrain, inventory, workstations, and occupancy. NPCs are removed; occupancy tests use immature plants. Vital-statistic timers retain their initial values and the environment remains at step zero. We execute a full environment step without replacing action, background-update, or random-number functions. Background updates still run, but under these initial conditions they do not change the compared vital statistics during that step. Thus, these are controlled interventions rather than samples from the natural state distribution. Deterministic cases run once. Each stochastic case runs 1,000 times from the same physical state with different environment random seeds. The evaluation comprises 566 deterministic cases and 19 stochastic cases, requiring 19,566 environment steps. The deterministic cases comprise 155 traversal, 115 harvesting, 72 crafting, and 224 placement cases.

Metrics. For deterministic cases, a match requires agreement on the entire visible effect. For a state–action pair q, stochastic predictions are compared with empirical outcome frequencies using total variation (TV):

$$\widehat{E}(q) = \frac{1}{2} \sum_{y} \left| P_{\mathrm{ref}}(y \mid q) - \widehat{P}_{\mathrm{env}}(y \mid q) \right|. \tag{4}$$

We compute two-sided 95% Wilson intervals for each outcome frequency. These are pointwise intervals, not a joint acceptance criterion; finite-sample empirical TV need not be zero even for a matching distribution. We report both the mean over equally weighted state–action pairs and the mean obtained by averaging within each assignment and then equally across the 30 assignments.

Repeated stochastic trials improve probability estimates without increasing a case's aggregation weight. Cases without a prediction remain in the denominator and receive TV of one.

Results and sensitivity to injected errors. The reference prior produces complete predictions for all 585 controlled cases and matches all 566 deterministic cases, including harvesting and crafting at inventory capacity. Its case-averaged and assignment-averaged empirical TV are 0.0410% and 0.0421%, respectively (Table 5).

All 19 stochastic cases involve grass harvesting with a predicted drop probability of 50%. A drop event is identified by an increment in the `collect_sapling` achievement count; at full inventory, the event can occur without increasing the sapling quantity. Across 19,000 trials, 9,602 drop events occur, giving a pooled frequency of 50.5368% with a 95% Wilson interval of $[49.8259\%, 51.2476\%]$. Per-case frequencies range from 47.8% to 52.9%, and the predicted probability lies within the corresponding pointwise 95% interval in all 19 cases. Mean empirical TV over these cases is 1.2632%. These results support using the prior as an environment-consistent reference within the tested states, conditions, and visible effects; they do not establish correctness for other role assignments, all reachable states, or complete world-state predictions.

To check whether the evaluation can reveal errors, we independently introduce seven single changes to the reference prior: remove the tree/R1 assignment; require a wooden pickaxe for harvesting trees; change the wood cost of a wooden pickaxe to two; remove the workstation requirement for crafting an iron pickaxe; allow planting on sand; change the grass drop probability from 0.5 to 0.1; or change the drink quantity limit to eight. Every model is evaluated against the same environment outcomes. The deterministic changes produce counterexamples or missing predictions. The drop-probability change raises mean TV over the 19 stochastic cases from 1.2632% to 40.5368%. All seven error types are detected through their corresponding scenarios.

Table 5: Environment-transition validation of the frozen reference prior and independent single-error controls. Deterministic mismatches are counted over 566 cases and include missing predictions. Both TV averages cover all 585 cases and are reported as percentages.

Prior	Deterministic mismatches	Missing predictions	Case mean TV	Assignment mean TV
Reference	0	0	0.0410	0.0421
Remove tree/R1	19	19	3.2889	3.3754
Require tool for tree	12	0	2.0923	2.1474
Change wood cost	7	0	1.2376	2.1633
Remove workstation requirement	4	0	0.7248	1.0677
Allow planting on sand	4	0	0.7248	0.2802
Change drop probability	0	0	1.3166	1.3512
Change drink quantity limit	3	0	0.5538	0.5421

Natural-trajectory diagnostic. We additionally collect six trajectories from normal spawn states using independent environment seeds and uniformly random actions, with a limit of 300 steps per trajectory and termination on death. Resources are not prepared, NPCs remain present, and the prior neither selects actions nor learns from feedback. The trajectories contain 1,140 transitions and visit 15 of the 30 assignments: R0–R3 predictions are available for 787 transitions, and 113 of these have observed effects outside the predicted support. The remaining 353 transitions comprise 222 steps in sleeping states, 122 actions outside the 30 assignments, and nine object interactions. We retain all transitions, including background changes and cases without predictions. The mean multiclass Brier score is 0.825439 when unavailable predictions receive the maximum penalty of two. This diagnostic reflects both prediction quality and coverage of R0–R3; its discrepancies cannot be interpreted directly as parameter errors in these assignments or as a measure of the entire prior's predictive accuracy, since joint prediction with other role instances is not implemented here.