# What Is Worth Representing? Representational Empowerment for Continual Model Construction

**Fei Dai**[*]
University of California, Berkeley
f_d@berkeley.edu

**Hanqi Zhou**[*]
University of Tübingen, TU Darmstadt
hanqi.zhou@uni-tuebingen.de

**Alison Gopnik**
University of California, Berkeley
gopnik@berkeley.edu

**Charley M. Wu**
TU Darmstadt
charley.wu@tu-darmstadt.de

## Abstract

The first problem of modeling the world is not just estimating the right parameters or causal structure, but deciding *what should be represented at all*. We frame this problem as *continual model construction*: an agent maintains an environment-specific model $M$ of an inaccessible world $W$ and curates a persistent library $\mathcal{L}$ of reusable representational elements across environments. We propose *Representational Empowerment* ($\mathrm{RepEmp}$) to score candidate elements by how much they expand the agent's future capacity to model and plan, complementing the classic definition of empowerment, but redefined as control over internal representations instead of external states. We realize the framework as a hierarchical Curator-Actor architecture and test it across three experiments. In a closed-vocabulary causal-learning task, human participants construct causal models at varying abstraction granularities to maximize goal reachability rather than fidelity to the world—a signature better predicted by $\mathrm{RepEmp}$ over information-gain alternatives. Matched simulations reveal $\mathrm{RepEmp}$-guided construction contributes more than exploration to sufficient structure recovery and cross-task transfer. Finally, in an open-vocabulary planning domain, an LLM-augmented Curator builds more compact symbolic libraries, which also generalize better than baselines. Ablating $\mathrm{RepEmp}$ eliminates these benefits. Together, these results identify $\mathrm{RepEmp}$ as a key principle for continual model construction: deciding what to build, retain, and reuse under bounded resources.

## 1 Introduction

There does not exist a complete, ready-made vocabulary for modeling our observations in the world. Concepts such as force, mass, and acceleration in physics were *constructed* to explain, predict, and intervene in the world. The resulting vocabulary (e.g., Newton's laws), is not the world itself, but a compressed representation sufficient for prediction and control. That vocabulary is also *curated* over centuries [19]: Aristotelian categories gave way to Newtonian mechanics, which gave way to relativity. Each transition was driven by phenomena the old vocabulary could not support, while abstractions that continued to support reasoning in new domains persisted.

Human problem-solving operates on similar principles at the smaller developmental scale. A child learning to cook does not build a complete chemical model of food; she constructs just enough (e.g., "bigger food needs to cook longer") to achieve her goals, retaining what transferred successfully

[*]These authors contributed equally to this work.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

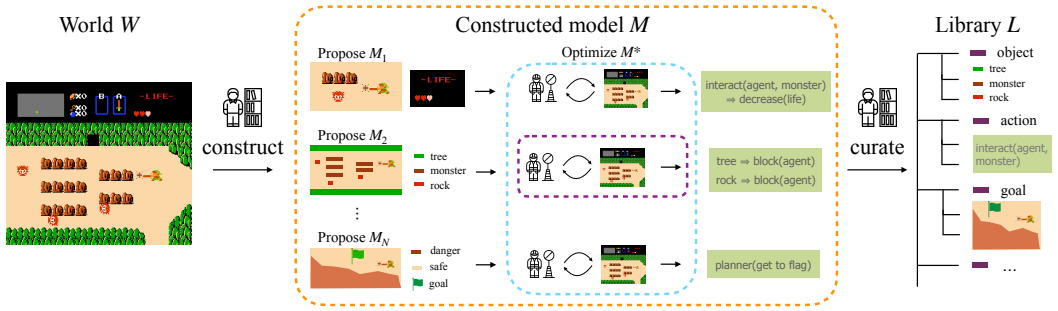


Figure 1: **Continual model construction as a $W \to M \to \mathcal{L}$ pipeline.** The agent is unable to recover the full world model $W$; instead it constructs candidate models $M_1, \ldots, M_N$, optimizing each against goals and curating reusable elements into a persistent library $\mathcal{L}$. Existing paradigms operate inside this loop without choosing $M$: world-model learning optimizes a single monolithic $M$ to all observation data, and the continual and meta-learning reuses $M$ across environments. Our framework treats the choice of $M$ and the curation of $\mathcal{L}$ as the primary objects of optimization.

and discarding what was specific to last Tuesday's dinner. This is a loop that cognitive science has long described: people plan with simplified, task-specific models rather than reconstructing the full complexity of the world [29, 10, 44, 59], consistent with resource-rational accounts of bounded agents [37, 20, 41, 62], and with children entertaining simplified causal hypotheses rather than full reconstructions [21]. Yet, both examples raise a question that precedes parameter estimation or structure learning: *which representational elements should an agent build, retain, and reuse?*

Existing work addresses important parts of this problem without making model construction itself the main object of optimization. Continual learning seeks to preserve knowledge across tasks [31, 52, 49]; world-model and causal learning estimate structure within a supplied state or variable space [26, 25, 7, 30]; and LLM-based agents leverage symbolic knowledge with little pressure to select what is kept [56, 58]. These approaches motivate but do not yet provide a general criterion for deciding which candidate representational elements should be retained under resource limitations (Sec. A).

Answering this question requires separating three objects that formal treatments often conflate (Fig. 1): an environment's inaccessible generative process $W$, the task model $M$ constructed from finite interaction, and a persistent library $\mathcal{L}$ of reusable elements. The resulting $W \to M \to \mathcal{L}$ pipeline makes two decisions explicit: what should enter the current representation, and what should survive beyond it. Within an environment, *construction* changes $M$ so that the agent can model and plan for its current goals; across environments, *curation* determines which elements remain available in $\mathcal{L}$ for future tasks. To guide both decisions, we introduce *representational empowerment* (RepEmp) [61] as an information-theoretic measure of how new representations expand the agent's future capacity to model and plan. Sec. 2 formalizes this criterion, and Sec. 3 shows how a Curator-Actor architecture evaluates them directly in enumerable settings and approximates them when representations must be invented online.

**Contributions.** (1) We formulate continual model construction as a $W \to M \to \mathcal{L}$ pipeline where agents construct environment-specific models and curate reusable elements across tasks (Sec. 2). (2) We introduce *representational empowerment* as a model construction criterion, and realize it in a Curator-Actor architecture that proposes, scores, and curates elements through planning and execution (Sec. 3). (3) In closed-vocabulary settings, where the candidate elements form a fixed set of representation granularities (Sec. 4), human choices are best explained by empowerment; matched simulations then show that RepEmp-based representation selection causally improves cross-task transfer (Sec. 4). (4) In an open vocabulary, an LLM-augmented Curator invents predicates and operators online and uses execution-grounded evidence to build compact Planning Domain Definition Language (PDDL) [4] libraries. Against PPO, Motif [32], and WorldCoder [56], the curator demonstrates better cross-task transfer and library compactness.

## 2 Representational Empowerment for Model Construction

We frame continual model construction as decisions about what enters a task model and what persists in a library. This section motivates our framework: why *three* modeling objects should be considered

rather than one, why *empowerment* is the right criterion for what those objects should contain, and why empowerment must be measured over *internal* representations. Sec. 3 then turns this framework into a concrete hierarchical agent. Formal properties and proofs are deferred to Sec. B.

**Three objects in the $W \to M \to \mathcal{L}$ pipeline.** World-model learning typically treats the environment $W$ as the object to be learned, seeking a model that captures its dynamics from experience [26]. For a resource-bounded agent, however, representing all of $W$ is neither feasible nor necessary. It must construct a limited task model $M$ and decide which elements to carry forward in a reusable library $\mathcal{L}$ [59, 44]. Concretely, the agent encounters environments $e_1, e_2, \ldots \sim \mathcal{E}$, each generated by an inaccessible process $W_e$. It assembles a task model $M_e$ from candidate elements $\sigma \in \Sigma$, where an element may be an observation granularity, causal relation, predicate, or operator schema. A persistent library $\mathcal{L}$ carries selected elements across environments. Thus, *construction* modifies $M_e$ within an environment, whereas *curation* modifies $\mathcal{L}$ across environments.

**What to represent?** Once $M_e$ and $\mathcal{L}$ are themselves objects of choice, the central problem is how to evaluate a candidate representational change: which changes expand the agent's modeling capacity and which merely add irrelevant detail? *Information gain* [38, 39, 45] rewards uncertainty reduction about $W_e$, whether or not the resolved distinction changes what the agent can represent or use. *Environmental empowerment* [33, 50, 42, 36] instead measures controllable future diversity of states within $e$: it treats the environment as an information channel from action sequences to future states, whose capacity $\mathrm{EnvEmp}_T(s; e) = \sup_{p(a_{1:T})} I(A_{1:T}; S_T \mid S_0 = s, e)$ is high when many states are reachable from $s$ and the agent can reliably steer to a chosen target. This provides the right structural idea about controllable future diversity, but is mapped to external states, leaving the agent's representation fixed (see Sec. B.7 for a detailed contrast).

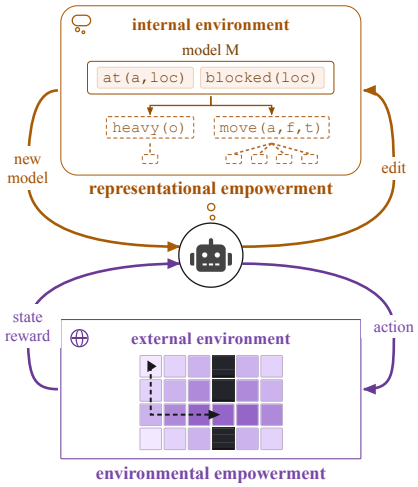


Figure 2: **Two forms of empowerment.** Representational empowerment (RepEmp) describes how an agent edits its own model $M$, encouraging representations that support reaching future models (dashed boxes). Environmental empowerment (EnvEmp) is the traditional definition, describing how an agent acts on the external environment and encourages reaching future states (shading).

*Representational empowerment* instead makes representational change the channel itself: edits to the agent's representation are the inputs and future representations are the outputs [61]. Let $Z_t = (M_t, \mathcal{L}_t)$ denote the agent's representational state and $\Omega$ a finite alphabet of edits to that state (e.g., ADD, REMOVE, or REFINE; Sec. B.2). Over a horizon $T$, an edit sequence $U \in \Omega^T$ induces through $K_T$ a distribution over future representational states $Z_T$. The capacity of this channel defines representational empowerment:

$$\mathrm{RepEmp} = \mathcal{R}_T(z) \;=\; \sup_{\rho \in \Delta(\Omega^T)} I_{\rho K_T}\big(U;\, Z_T \mid Z_0 = z\big). \tag{1}$$

Thus, RepEmp is high when edits reliably reach diverse future representations. Eq. 1 defines the exact criterion, though evaluating it empirically requires an observable way to compare the internal representations it distinguishes; we address this next by projecting the channel onto goal achievement.

**Grounding the criterion in the environment.** Exact representational empowerment distinguishes future internal states $Z_T$, including representations that differ syntactically but support the same behavior. For empirical evaluation, we observe a task model through its goal-achievement signature $\Gamma_e(M) = (\,\mathbb{I}[\,g \text{ is achieved by planning with } M \text{ and executing in } e\,]\,)_{g \in \mathcal{G}_e}$. Applying $\Gamma_e$ to the channel's outputs grounds it in goal outcomes in $e$:

$$\mathcal{R}^{\Gamma}_T(z; e) = \sup_{\rho \in \Delta(\Omega^T)} I_{\rho K_T}\big(U; \Gamma_e(Z_T) \mid Z_0 = z\big). \tag{2}$$

This projection is a measurement of representational empowerment rather than an alternative definition: it preserves exactly those distinctions in $Z_T$ that are visible through goal achievement. When the projected channel is enumerable, we evaluate it directly; otherwise, Sec. 3 approximates its relevant

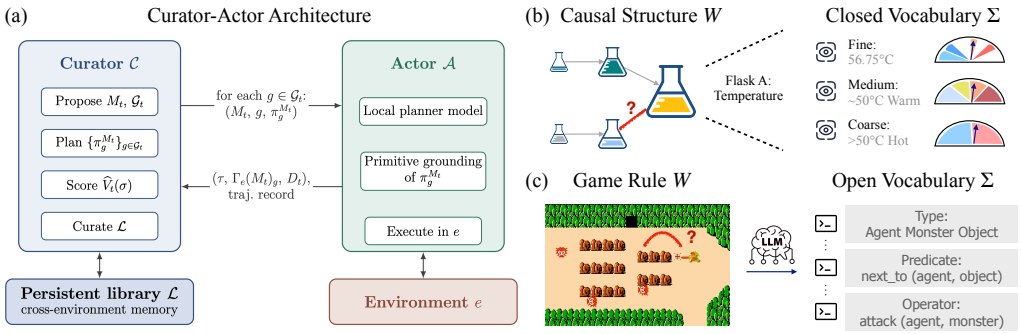


Figure 3: **Curator-Actor architecture across experiments. (a)** The Curator proposes and scores model edits; the Actor grounds and executes their plans, returning execution evidence that determines what is retained in the persistent library. **(b)** Closed vocabulary (Exp. 1-2): $\Sigma$ is a fixed set of model granularities/variables/properties. **(c)** Open vocabulary (Exp. 3): $\Sigma$ is an LLM-proposed space of PDDL types, predicates, and operators; curated elements enter $\mathcal{L}$ and seed $M$ on the next task.

consequences using finite probes and execution feedback. Its formal relation to exact representational empowerment is given in Sec. B.

## 3 The Curator-Actor Architecture

Continual model construction operates on two timescales: representational commitments persist across environments, whereas interactions with the environment unfold within a single task. We realize the agent with a hierarchical architecture (Fig. 3a): a slow, high-level *Curator* $\mathcal{C}$ decides what to build and retain (i.e., editing the task model $M_t$ and the library $\mathcal{L}$), while a fast, low-level *Actor* $\mathcal{A}$ plans and acts under the model supplied by the Curator. Each level of the architecture only depends on what it needs. The Actor depends only on the current model, not the history of edits. The Curator depends on execution outcomes, not the actions that achieved them. We present the architecture at the level of detail the experiments require, and give its full specification in Secs. B and F with the loop itself in Algorithm 1.

**Propose and probe (Curator).** At iteration $t$, the Curator receives the library $\mathcal{L}$, observations from environment $e$, and a trajectory buffer $\mathcal{D}_t$. It assembles $M_t$ from relevant elements of $\mathcal{L}$ and new candidate elements from $\Sigma$, and constructs a probe-goal set $\mathcal{G}_t$ that puts those candidates to the test: *positive probes* test goals a candidate could newly enable (gained capabilities), whereas *regression probes* test goals the previous model already supported (preserved capabilities). The Curator then plans for each probe goal under $M_t$ and hands the plans to the Actor.

**Execute (Actor).** For each probe goal $g$, the Actor receives the model, the goal, and its plan $(M_t, g, \pi_g^{M_t})$. Because the plan is expressed in the abstract operators of $M_t$, the Actor grounds it into the primitive actions available in $e$ and executes it. It reports back the realized trajectory, whether the goal was achieved ($\Gamma_e(M_t)_g$), and any operator whose predicted effects failed to hold. Across $\mathcal{G}_t$ these outcomes give the goal-achievement signature $\Gamma_e(M_t)$, the execution evidence that separates a merely plausible symbolic proposal from a representation that supports successful behavior.

**Score and promote (Curator).** With the probe outcomes in hand, the Curator evaluates each candidate by its effect on the grounded channel. When the candidates and goal outcomes can be enumerated (Exps. 1-2), the score is computed exactly. In the open-vocabulary setting (Exp. 3), neither the candidate space nor the space of relevant goals can be enumerated. Only there, we use the finite-sample approximation

$$\widehat{V}_t(\sigma) = q_t(\sigma) \sum_{g \in W_t(\sigma)} [\widehat{p}_t(g \mid M_t \cup \{\sigma\}) - \widehat{p}_t(g \mid M_t)] - \lambda c_\sigma, \tag{3}$$

where $q_t(\sigma)$ estimates the validity of candidate $\sigma$, the witness set $W_t(\sigma)$ contains sampled goals that are currently blocked but could be unlocked by $\sigma$, $\widehat{p}_t$ is the empirical achievement rate returned by the Actor, and $\lambda > 0$ weights the representational cost $c_\sigma$. The Curator accepts $\sigma$ into $M_t$ when $\widehat{V}_t(\sigma) > 0$ and its probe successes are reliable, and otherwise retains the previous model. This

score approximates Eq. (2), whose capacity comes entirely from edits that change goal outcomes; $\widehat{V}_t$ measures those changes directly on sampled probes.

The three steps above adapt $M_t$ within a single environment, while curation operates at the slower timescale of the environment sequence $e_1, e_2, \ldots$ (Sec. 2). Before moving on to the next environment, the Curator re-evaluates each retained element conditional on the rest of the library and the accumulated evidence. Removing elements that are unreliable, redundant, or narrowly task-specific compresses the library, and the resulting $\mathcal{L}$ seeds model construction in the next environment.

**Experimental regimes.** The framework makes three claims that require different kinds of evidence: (i) $\mathrm{RepEmp}$ *describes* how natural agents construct models, (ii) optimizing $\mathrm{RepEmp}$ *improves* performance, and (iii) $\mathrm{RepEmp}$ *scales* to settings where the vocabulary itself must be invented. We therefore instantiate the Curator-Actor architecture across two regimes (Fig. 3b-c). In the *closed-vocabulary* regime, the vocabulary is a fixed menu of representation granularities, so every criterion for selecting representational elements ($\mathrm{RepEmp}$, $\mathrm{EnvEmp}$, information gain) can be scored exactly on the same alternatives. Exp. 1 provides correlational evidence by fitting these criteria to human choices, while Exp. 2 provides causal evidence by implementing them in matched simulated agents. In the *open-vocabulary* regime, an LLM proposes PDDL predicates and operators online, so models must be invented rather than chosen from a menu and $\mathrm{RepEmp}$ is only available through the finite-sample score of Eq. (3). Exp. 3 tests whether empowerment-guided curation survives such recursive dynamics and whether it builds a compact, transferable library. The experiments thus progress from human alignment, to causal mechanism, to open-vocabulary realization.

## 4 Closed-Vocabulary Causal Learning

With the framework and architecture in place, we test two implications: whether $\mathrm{RepEmp}$ predicts human representation choices, and whether optimizing $\mathrm{RepEmp}$ improves structure learning and generalization. In a closed-vocabulary setting, the candidate representations are finite and enumerable, so we can compute $\mathrm{RepEmp}$, $\mathrm{EnvEmp}$, information gain exactly to compare them. Specifically, we use a causal-discovery environment in which $\Sigma$ is a fixed set of representation granularities. Exp. 1 (Sec. 4.1) tests the framework's predictions against human representation and intervention choices. Exp. 2 (Sec. 4.2) implements each criterion (representation choice and intervention choice) with simulated agents and tests whether $\mathrm{RepEmp}$ captures optimal granularity under different conditions.

**Causal-discovery environment.** Both experiments use a multivariable causal-discovery environment adapted from Ke et al. [30], presented as an alchemy task. The world $W$ is a structural equation model (SEM) whose latent variables are flasks, each carrying three observable properties (temperature, color, and weight). Learners never observe $W$ directly. Instead, they complete a sequence of tasks, each exposing a small subset of the flasks. Within a task, the learner explores by intervening to discover the causal rules among the visible flasks. Each step comprises two choices. First, the learner selects an observation granularity $\ell \in \{\texttt{coarse}, \texttt{medium}, \texttt{fine}\}$. Second, the learner intervenes on a variable-property pair $(v, r)$, such as setting flask A's temperature to a chosen value. The intervention propagates through $W$, and its consequences are reported at the selected granularity. At the end of each task, the learner reports a causal graph and uses it in goal-directed tests, where each goal asks for a specific property to be brought to a target value (e.g., "set flask B's weight to level 3"). Here $M_t$ is the causal model constructed from observations at $\ell$ (different granularities), and $\mathcal{L}$ contains structure retained across task windows. Finer observations distinguish more values but are noisier, so the granularity that best identifies goal-relevant structure need not be the finest (Sec. C).

### 4.1 Experiment 1: Human causal learning

Participants ($n$=70; see Sec. D) completed four such tasks, choosing an observation granularity $\ell$ and a flask-property intervention $(v, r)$ at each step. We ask whether (A) participant behavior is consistent with resource-rational model construction, (B) whether this pays off in goal achievement, and (C) if $\mathrm{RepEmp}$ provides the best account of human choices about the granularity of observations.

**(A) Human choices are consistent with resource-rational model construction.** Participants chose `medium` granularity on $43\%$ of decisions, compared with $38\%$ for `fine` and $19\%$ for `coarse` (Fig. 4a). $\mathrm{RepEmp}$ also prefers `medium` granularity under this setting, since it provides sufficient resolution for achieving the probe goals without adding too much noise (see Exp. 2 for settings

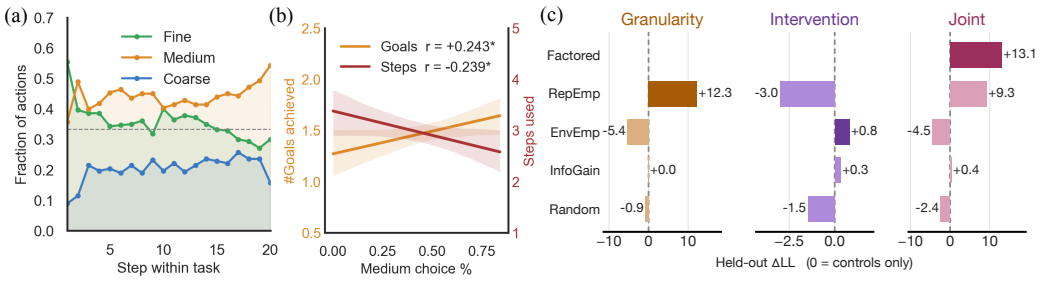


Figure 4: **Exp. 1 results. (a)** Fraction of actions at each observation granularity as a function of step within a task, averaged across participants. **(b)** Across participants, greater use of `medium` was associated with more goals achieved and fewer steps to achieving a solution in the generalization environments. **(c)** Held-out $\Delta$LL of each selection criterion (including random effects and control variables) for granularity choice, intervention choice, and both jointly. Zero is the control-only model and positive values indicate better performance.

where RepEmp predicts other granularities). Across tasks, choices converged on `medium` from both extremes, with `medium` rising from 35% to 46% ($t(69)$=2.83, $p$=.006) and `coarse` *falling* from 25% to 17% ($t(69)$=−2.86, $p$=.006). This rules out a cost-only account, which predicts participants simply minimize representational cost (see Sec. D.2 for within-task dynamics). Cost minimization would consistently favor `coarse`, whereas participants moved away from it instead, progressively favoring the task-sufficient `medium` granularity (Proposition 2). A second alternative is that `medium` is preferred because it is easier to process, with goal sufficiency playing no role. An effort account would predict that granularities differing in processing demand would differ in reaction time (RT), but median RTs were largely equivalent at 7.6/7.6/7.3 s for `coarse`/`medium`/`fine`, respectively, with the `fine`-`medium` contrast on log-RT returning a null result ($\beta$=−0.003, $p$=.94; Sec. D.2).

**(B) Goal-sufficient compression covaries with performance.** Greater `medium` use predicted more goals achieved ($r$=.24, $t(68)$=2.07, $p$=.042) and in fewer steps ($r$=−.24, $p$=.046; Fig. 4b), while greater `fine` use predicted the opposite ($r$=−.29, $p$=.015; $r$=.31, $p$=.010), and `coarse` use predicted neither ($p$>.26). A logistic mixed model over the 560 binary goal outcomes, with a participant random intercept, confirmed that participants' overall `medium` use (between-participant component) was positively associated with goal achievement ($\beta$=1.29, $z$=2.07, $p$=.039). The task-specific deviation from each participant's own average (the within-participant component) was positive but unreliable ($\beta$=.59, $p$=.31). We therefore interpret this as an association rather than a causal effect; Exp. 2 provides the corresponding manipulation (Sec. D.3).

**(C)** RepEmp **best explains granularity choices.** To compare various choice criteria, we reconstructed each participant's belief state online and scored all actions on the same observed trajectory. Utilities were evaluated using leave-one-task-out predictions in a hierarchical regression controlling for population and participant random effects, and with previous-choice stickiness, and intervention familiarity as control variables (Sec. D.3). Fig. 4c reports $\Delta$LL as model improvement over a baseline containing only the control variables. RepEmp provided the best predictions of granularity, while EnvEmp provided the best predictions of intervention source. However, combining both RepEmp and EnvEmp yields the largest joint increment across all pairings, with almost all of the factorized improvement in predictions coming from RepEmp.

**Exp. 1 summary.** Across tasks, participants progressively compress from initially fine observations toward the goal-sufficient `medium` granularity. Greater `medium` use was associated with better goal performance. Controlling for individual preferences and stickiness, the factored RepEmp-granularity $\times$ EnvEmp-intervention criteria achieved the highest joint fit, with RepEmp playing the largest role. Next, Exp. 2 uses simulations to directly test how these criteria influence learning and generalization.

### 4.2 Experiment 2: Mechanistic evaluation with agent-based simulations

We now use simulations to reproduce the causal-discovery problem with two changes the human experiment cannot make. Bayesian agents let us assign the selection criteria directly rather than inferring them from choices, while varying observation noise across granularities changes which granularity is optimal. Each of 30 seeds sampled a ten-variable DAG ($p_e$=.25) and generated a curriculum of five training and five held-out tasks. Each task exposed 4-7 of the ten variables and

Table 1: **Exp. 2: Graph recovery when `medium` is optimal.** Absolute F1 (mean, SEM; 150 seed-task instances per split).

| Condition | Train | Held-out |
|---|---|---|
| Factored | **.720 (.021)** | **.703 (.021)** |
| RepEmp | .607 (.022) | .592 (.020) |
| InfoGain | .304 (.013) | .383 (.016) |
| Random | .307 (.014) | .365 (.013) |
| EnvEmp | .195 (.015) | .120 (.014) |

Table 2: **Exp. 2: One-factor-at-a-time ablations.** Held-out F1 (mean, SEM) when one decision is replaced and the other is held at the specified criterion.

| Replaced | Criterion | Held-out F1 | Δ |
|---|---|---|---|
| Granularity | RepEmp | **.703 (.021)** | — |
| | InfoGain | .425 (.014) | −.278 |
| | Random | .376 (.015) | −.327 |
| Intervention | EnvEmp | **.703 (.021)** | — |
| | InfoGain | .596 (.020) | −.107 |
| | Random | .507 (.021) | −.196 |

retained about half of the preceding task's variables, so consecutive tasks shared structure and the beliefs formed in one task remained useful in the next. Agents were given 1,000 interventions per task and carried their beliefs across task boundaries. Each condition paired one granularity criterion with one intervention criterion (RepEmp, EnvEmp, information gain, and random) (Sec. E).

**(A) Representation selection drives performance.** Tab. 1 reports F1 score for different conditions. The factored condition pairs RepEmp granularity selection with EnvEmp intervention selection, and achieves the best performance on both train and held-out splits. Comparisons across criteria use the same task order for a fair comparison. Tab. 2 clarifies the contributions of RepEmp and EnvEmp by iteratively substituting each with the other criteria, using held-out F1 as a performance metric. Replacing RepEmp as the granularity criterion reduces performance close to the random floor for each alternative, whereas replacing EnvEmp as the intervention criterion is less costly, even when interventions are chosen uniformly at random. Granularity selection therefore carries roughly three times the weight of intervention selection.

**(B) Adaptation to different optimal regimes.** To test whether RepEmp can adapt to different optimal granularities, we manipulate observation noise separately for each granularity. RepEmp selects `medium` on 63% of decisions when `medium` is the most reliable, and `fine` on 97% of decisions when `fine` is the most reliable. In contrast, EnvEmp prefers `fine` in every regime, because the state-diversity score grows with the number of bins, while InfoGain scores the three granularities roughly the same. Adapting to the granularity that is most reliable is therefore what separates RepEmp from the alternative criteria, and from fixing the granularity in advance (Sec. E.2).

In the regime where `coarse` is optimal, RepEmp nevertheless selects `fine` on 91% of decisions. With 1,000 interventions, repeated `fine` observations can be averaged to reduce noise while preserving eight distinguishable value bins, compared with only two under `coarse`. When the budget falls to 20, there are too few observations to average out this noise, and RepEmp instead selects `coarse` on 59% of decisions (Sec. E.2).

**Exp. 2 summary.** Exp. 2 establishes that representation selection causally drives graph recovery. Manipulating the granularity criterion changes held-out F1, which ablations attribute to granularity rather than intervention selection, while RepEmp adapts its granularity selection to different reliability contexts.

## 5 Open-Vocabulary Symbolic Construction

The closed-vocabulary experiments held the candidate set $\Sigma$ fixed (Sec. 4). Here, we remove that assumption by allowing candidates to be generated online from interactions, expanding the question to whether the same selection criterion can build a transferable library when the vocabulary itself must be chosen. This open-vocabulary regime allows us to test the $M \to \mathcal{L}$ step of the pipeline under an unbounded candidate space. We instantiate the Curator-Actor architecture with LLM-augmented components in two types of grid-worlds, where $M$ is a PDDL domain (i.e., typed predicates, operator schemas, or a type hierarchy) with grounding, and $\mathcal{L}$ carries promoted elements across environments. When generating representations, we semantically define each candidate $\sigma$ as a typed predicate or operator schema rather than a representation granularity [55]. Since candidates can be invalid, the validity term $q_t$ in Eq. (3) estimates whether the grounding code can fail on real observations. We also now use the cost term $\lambda c_\sigma$ to penalize retention, whereas it was previously inactive in the closed-vocabulary setting.

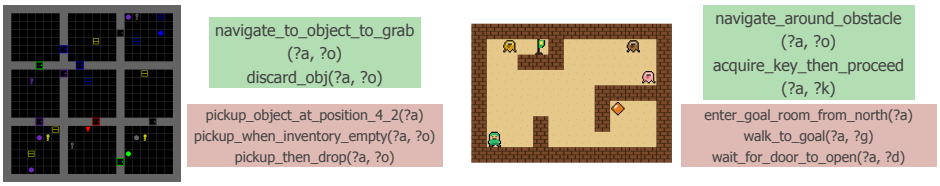


Figure 5: **Promoted vs. discarded operators in BabyAI (left) and Zelda (right).** Green = operators retained in $\mathcal{L}$ after passing the RepEmp threshold; red = candidates discarded despite being reliable on individual probes, because their witness sets are too narrow to earn persistence.

Table 3: **Open-vocabulary symbolic construction results. (a)** Transfer success across tasks within each environment (mean $\pm$ std over 3 seeds). **(b)** Training costs (total LLM tokens in millions) and library compactness (source code in KB averaged across the curriculum). See Sec. G for environment-specific results.

(a) Transfer success

| | BabyAI | | | Zelda | | |
|---|---|---|---|---|---|---|
| Method | Fwd. | Bwd. | H.O. | Fwd. | Bwd. | H.O. |
| PPO | .29±.13 | .33±.15 | .01±.00 | .03±.01 | .08±.05 | .03±.02 |
| Motif | .04±.03 | .23±.10 | .12±.05 | .19±.03 | .40±.05 | .18±.02 |
| WorldCoder | .32±.11 | .40±.10 | .35±.10 | **.35**±.02 | .36±.02 | .38±.02 |
| Hoarder | .20±.11 | .24±.10 | .03±.01 | .07±.04 | .12±.02 | .06±.00 |
| **Curator** | **.51**±.19 | **.61**±.07 | **.47**±.13 | .24±.07 | **.51**±.06 | **.48**±.01 |

(b) Training cost & footprint

| Method | Tokens (M) | Library (KB) |
|---|---|---|
| BabyAI | | |
| WorldCoder | 6.50 | 18.3 |
| **Curator** | **5.14** | **8.0** |
| Zelda | | |
| WorldCoder | 4.13 | 7.1 |
| **Curator** | **2.94** | **4.4** |

We report three main findings. RepEmp-scored curation (*Curator* model) outperforms baselines on backward transfer and held-out generalization in both environments. Removing the empowerment criterion (*Hoarder* model) collapses these gains, thus isolating it as a result of curation rather than proposal quality. As a result, the curator model requires substantially fewer LLM tokens than WorldCoder.

### 5.1 Setup

**Environments.** We use two types of grid worlds. In *BabyAI* [11], training proceeds along a curriculum ordered by how many operators a solution requires:`GoToLocal` (1 operator), `Pickup` (3), `OpenRedDoor` (3), `UnlockLocal` (6+). Each environment introduces new operator requirements that compose those from earlier environments. A held-out split tests compositional transfer to configurations not seen during training. In *Zelda* we use the layout from EMPA [57], with the first three environments for training and the last two held out for testing. The environments naturally become more difficult as new kinds of monsters are introduced. Full details are in Sec. F.

**Evaluation.** After training, $\mathcal{L}$ is frozen. We report *forward transfer* ($\mathcal{L}$ frozen after environment $i$, evaluated on $i{+}1$), *backward transfer* (the final $\mathcal{L}$ re-evaluated on each earlier training environment; Sec. G.1), and *held-out generalization* on the held-out environments. We compare four baselines. PPO [51] learns a primitive-action policy from rewards. Motif [32] adds an LLM-shaped intrinsic reward but remains policy-level. WorldCoder [56] writes executable world-model code from traces and accumulates it without selective curation. The *Hoarder* ablation uses the Curator's proposer, budgets, and reliability checks, but "hoards" each reliable proposal without being selective. Interaction budgets are matched where applicable (Sec. F.5): at matched LLM budgets, symbolic methods use about two orders of magnitude fewer environment steps than the policy baselines (Sec. F.3).

### 5.2 Results

**(A) Curated libraries transfer backward and generalize to held-out environments.** Table 3a reports success rates, where the Curator obtains the best backward and held-out transfer in both environments (see Sec. G for consistency across seeds). WorldCoder is competitive and exceeds the Curator on Zelda forward transfer, but failed on the backward and held-out columns. Thus, curated abstractions retroactively improve earlier environments and generalize to unseen ones, whereas typically accumulated code does not. In comparison, PPO and Motif transfer poorly because their policies are tied to the environment they were trained in. Motif's preference labels are at chance ($50.2\%$) without access to the explicit task specification, so its learned reward carries no signal (Sec. F.5). Fig. 5 shows which representational candidates were selected (green) or discarded (red) by RepEmp.

General schemas such as `navigate_to_object_to_grab` and `acquire_key_then_proceed` are retained, while position- and context-specific candidates (`pickup_object_at_position_4_2`, `enter_goal_room_from_north`) are discarded even though they succeed on their own probes.

**(B) Curation lowers LLM cost and keeps the library compact.** Table 3b reports total LLM tokens consumed during training and the size of the final library. WorldCoder's reward model grows monotonically with the number of environments (on BabyAI it triples from env. 0 to env. 3) because each new environment requires a new model, and prior branches cannot be deleted without breaking earlier goals. In contrast, the Curator selects sparingly and preserves reusable elements by carrying over typed predicates and operator schemas, such that new environments rebind the same handful of proven symbols to new objects. Its final library is $2.3\times$ (BabyAI) and $1.6\times$ (Zelda) smaller than WorldCoder's, and remains approximately constant in size across the curriculum (Sec. G.3).

**Failure case.** On one BabyAI held-out environment, every method including the Curator was below $15\%$ success (Fig. 12). The environment requires a conjunction of binary object-object relations ("move the A next to the B and the C next to the D"), but every training goal the LLM proposed involved a single object or the agent's own state (holding, unlocked), never a relation between two objects. Even when the proposer generates a candidate `next-to` predicate, the witness set $W_t$ contains no compositional goal on which it could score. Although $\mathrm{RepEmp}$ behaves correctly on the probes it was given, they did not cover the required goal structure. Eq. (3) is in this sense conditional on the quality of probes in how well they capture future tasks (Sec. B.4). Increasing the diversity of goal proposals [15, 47] is thus an important future direction.

**Exp. 3 summary.** In both environments the vocabulary space is unbounded, such that the uncurated retention of acquired representations hinders performance, whether representations are symbolic proposals (Hoarder) or one synthesized reward function per task (WorldCoder). Instead, $\mathrm{RepEmp}$ as a curation criteria achieves better transfer in each domain, with lower LLM tokens and smaller libraries. And although absolute success rates remain modest, we read these results as preliminary, directional evidence for $\mathrm{RepEmp}$ as a model construction criterion rather than as a benchmark claim. More environments are needed to test the generality and probe the boundary cases.

## 6 Discussion

We asked which representational elements an agent should build, keep, and reuse. We propose *representational empowerment* $\mathrm{RepEmp}$ as the answer, which quantifies the value of a representational element by the extent to which it expands the plannable goal space per unit of representational cost. $\mathrm{RepEmp}$ provides an extension of the classic *environmental empowerment* $\mathrm{EnvEmp}$, by turning inwards to focus on the agent's choice of representations rather than control over the external environment. We showed that human learners selectively chose the same granularity of representations as predicted by $\mathrm{RepEmp}$ in a causal learning task, leading to better performance. Simulations revealed representation selection via $\mathrm{RepEmp}$ as the main normative factor, which more flexibly adapts between different optimal granularities than alterative models. Lastly, in the open-vocabulary regime, an LLM Curator scored the representations it had itself proposed by the same criterion, retaining a compact library that transferred better at lower token costs.

**Limitations.** The human experiment used a fixed set of interventions and representational granularities, with evidence for $\mathrm{RepEmp}$ provided only via behavioral and model-based analyses, rather than process-level evidence. A mechanistic understanding of the normative benefits of $\mathrm{RepEmp}$ thus only comes from agent-based simulations. Additionally, the open-vocabulary experiments used only two types of grid-worlds and should be read as a directional rather than a benchmarking result. The LLM curator can also only score what the proposer generates, so a narrow proposal distribution limits the effectiveness of $\mathrm{RepEmp}$. This is captured in a failure case, where $\mathrm{RepEmp}$ behaves correctly given the probes it saw, but the probes themselves did not sufficiently cover the relevant goal structure.

**Conclusion.** Continual learning, world-model learning, and causal discovery typically operate within a predefined and fixed representational vocabulary. Yet, the question of what representational vocabulary to build and maintain comes prior to such problems. Our results offer a candidate solution: agents should represent what expands their achievable goal space, then compress until only the minimum sufficient structure remains. Whether *Representational Empowerment* generalizes more broadly is still an open question, but here, we provide a first account of representational dynamics.

## Acknowledgments and Disclosure of Funding

This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme ($C^4$: 101164709), the Hessian research funding programme LOEWE/4b//519/05/01.002(0022)/119, the Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under Germany's Excellence Strategy (EXC 3066/1 "The Adaptive Mind", Project No. 533717223), the Excellence Cluster "Reasonable AI" by the Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under Germany's Excellence Strategy – EXC-3057, the DOD ONR MURI N00014-22-1-2773 Co-PI Self-Learning perception Through Real World Interaction grant, the CIFAR Catalyst Award "Causally guided exploration in children and AI", and the Co-PI NSF AI Research Institute on Interaction for AI Assistants (ARIA) (Grant no. 2433429. Subcontract from Brown University). .

## References

[1] D. Abel, D. Hershkowitz, and M. Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

[2] D. Abel, D. Arumugam, L. Lehnert, and M. Littman. State abstractions for lifelong reinforcement learning. In *International conference on machine learning*, pages 10–19. PMLR, 2018.

[3] D. Abel, M. Bowling, A. Barreto, W. Dabney, S. Dong, S. Hansen, A. Harutyunyan, K. Khetarpal, C. Lyle, R. Pascanu, et al. Plasticity as the mirror of empowerment. *arXiv preprint arXiv:2505.10361*, 2025.

[4] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson, et al. Pddl—the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[5] A. Ahmetoglu, S. James, C. Allen, S. Lobel, D. Abel, and G. Konidaris. Skill-driven neurosymbolic state abstractions. *Advances in Neural Information Processing Systems*, 38:10750–10785, 2026.

[6] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International conference on machine learning*, pages 501–510. PMLR, 2018.

[7] N. R. Bramley, P. Dayan, T. L. Griffiths, and D. A. Lagnado. Formalizing neurath's ship: Approximate algorithms for online causal learning. *Psychological review*, 124(3):301, 2017.

[8] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[9] P. S. Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[10] T. Chen, S. Cheyette, K. Allen, J. Tenenbaum, and K. Smith. " just in time" world modeling supports human planning and reasoning. *arXiv preprint arXiv:2601.14514*, 2026.

[11] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

[12] R. Chitnis. *Learning State and Action Abstractions for Effective and Efficient Planning*. PhD thesis, Massachusetts Institute of Technology, 2022.

[13] C. Colas, T. Karch, O. Sigaud, and P.-Y. Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

[14] S. N. Cresswell, T. L. McCluskey, and M. M. West. Acquiring planning domain models using locm. *The Knowledge Engineering Review*, 28(2):195–213, 2013.

[15] G. Davidson, T. M. Gureckis, and B. Lake. Creativity, compositionality, and common sense in human goal generation. In *Proceedings of the annual meeting of the cognitive science society*, volume 44, 2022.

[16] K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, and J. B. Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pages 835–850, 2021.

[17] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[18] N. Ferns, P. Panangaden, and D. Precup. Metrics for finite markov decision processes. In *Uai*, volume 4, pages 162–169, 2004.

[19] L. Fogarty, N. Creanza, and M. W. Feldman. Cultural evolutionary perspectives on creativity and human innovation. *Trends in ecology & evolution*, 30(12):736–754, 2015.

[20] S. J. Gershman, E. J. Horvitz, and J. B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015.

[21] A. Gopnik and H. M. Wellman. Reconstructing constructivism: causal models, bayesian learning mechanisms, and the theory theory. *Psychological bulletin*, 138(6):1085, 2012.

[22] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[23] C. Grimm, A. Barreto, S. Singh, and D. Silver. The value equivalence principle for model-based reinforcement learning. *Advances in neural information processing systems*, 33:5541–5552, 2020.

[24] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

[25] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[26] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[27] P. Hansen-Estruch, A. Zhang, A. Nair, P. Yin, and S. Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pages 8407–8426. PMLR, 2022.

[28] M. Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[29] M. K. Ho, D. Abel, C. G. Correa, M. L. Littman, J. D. Cohen, and T. L. Griffiths. People construct simplified mental representations to plan. *Nature*, 606(7912):129–136, 2022.

[30] N. R. Ke, A. Didolkar, S. Mittal, A. Goyal, G. Lajoie, S. Bauer, D. Rezende, Y. Bengio, M. Mozer, and C. Pal. Systematic evaluation of causal discovery in visual model based reinforcement learning. *arXiv preprint arXiv:2107.00848*, 2021.

[31] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[32] M. Klissarov, P. D'Oro, S. Sodhani, R. Raileanu, P.-L. Bacon, P. Vincent, A. Zhang, and M. Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

[33] A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pages 128–135. IEEE, 2005.

[34] L. Lehnert and M. L. Littman. Successor features combine elements of model-free and model-based reinforcement learning. *Journal of Machine Learning Research*, 21(196):1–53, 2020.

[35] L. Li, T. J. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

[36] A. Lidayan, Y. Du, E. Kosoy, M. Rufova, P. Abbeel, and A. Gopnik. Intrinsically-motivated humans and agents in open-world exploration. *arXiv preprint arXiv:2503.23631*, 2025.

[37] F. Lieder and T. L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1, 2020.

[38] D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[39] G. Loewenstein. The psychology of curiosity: A review and reinterpretation. *Psychological bulletin*, 116(1):75, 1994.

[40] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International conference on machine learning*, pages 2295–2304. PMLR, 2017.

[41] F. Mantiuk, H. Zhou, and C. M. Wu. From curiosity to competence: How world models interact with the dynamics of exploration. In A. Ruggeri, D. Barner, C. Walker, and N. Bramley, editors, *Proceedings of the 47th Annual Conference of the Cognitive Science Society*, San Francisco, CA, 2025. Cognitive Science Society. doi: 10.48550/arXiv.2507.08210.

[42] A. Modirshanechi, P. Dayan, and E. Schulz. Testing an integrative framework for the sense of control. In *8th Annual Conference on Cognitive Computational Neuroscience (CCN 2025)*, 2025.

[43] S. Mohamed and D. Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

[44] D. G. Nagy, T. Shen, H. Zhou, C. M. Wu, and P. Dayan. Analogy making as amortised model construction. *arXiv preprint arXiv:2507.16511*, 2025.

[45] J. D. Nelson. Finding useful questions: on bayesian diagnosticity, probability, impact, and information gain. *Psychological review*, 112(4):979, 2005.

[46] S. Park, T. Kreiman, and S. Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

[47] J. Perez, C. Colas, G. Molinaro, P.-Y. Oudeyer, M. Derex, and C. Moulin-Frier. The cultural evolution of human goals. *Topics in Cognitive Science*, page e70078, 2026.

[48] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International conference on machine learning*, pages 7750–7761. PMLR, 2020.

[49] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[50] C. Salge, C. Glackin, and D. Polani. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.

[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[52] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[53] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. *arXiv preprint arXiv:2206.10680*, 2022.

[54] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*, 2022.

[55] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12120–12129, 2023.

[56] H. Tang, D. Key, and K. Ellis. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. *Advances in Neural Information Processing Systems*, 37:70148–70212, 2024.

[57] P. A. Tsividis, J. Loula, J. Burga, N. Foss, A. Campero, T. Pouncy, S. J. Gershman, and J. B. Tenenbaum. Human-level reinforcement learning through theory-based modeling, exploration, and planning. *arXiv preprint arXiv:2107.12544*, 2021.

[58] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[59] L. Wong, K. M. Collins, L. Ying, C. E. Zhang, A. Weller, T. Gerstenberg, T. O'Donnell, A. K. Lew, J. D. Andreas, J. B. Tenenbaum, et al. Modeling open-world cognition as on-demand synthesis of probabilistic models. *arXiv preprint arXiv:2507.12547*, 2025.

[60] Q. Yang, K. Wu, and Y. Jiang. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2-3):107–143, 2007.

[61] H. Zhou, F. Mantiuk, D. G. Nagy, and C. M. Wu. Agent-centric learning: from external reward maximization to internal knowledge curation. *arXiv preprint arXiv:2507.22255*, 2025.

[62] H. Zhou, D. G. Nagy, P. Dayan, and C. M. Wu. Path-dependent program induction under resource constraints explains human sequence learning. *ArXiv*, 2026. doi: 10.48550/arXiv.2606.20623.

# A Related Work

The related works differ in which component of modeling they allow to change and which criterion governs that change. State-abstraction methods select task representations; symbolic and program-learning methods construct reusable vocabularies; empowerment and autotelic methods expand behavioral competence. Our framework connects these problems by treating both the environment-specific model $M$ and the persistent library $\mathcal{L}$ as objects of selection.

**Task-relative models and abstractions.** State abstraction formalizes which distinctions a decision-making agent can discard. Bisimulation and approximate bisimulation preserve reward and transition structure [18, 9]; $\phi$-relevance and $Q^*$-irrelevance preserve policy or value information [35, 1]; and PAC abstractions extend such guarantees across a distribution of tasks [2]. Related objectives include value equivalence for specified Bellman computations [23], successor features for transfer across reward functions [6, 34], goal-conditioned bisimulation [27], and skill-compatible neurosymbolic abstractions [5]. These methods generally optimize an abstraction of a supplied observation or state space under a specified equivalence or sufficiency criterion. Our setting extends representation selection to heterogeneous elements—including observation granularities, causal relations, predicates, and operator schemas—that can be added to or removed from $M$ and $\mathcal{L}$. Their value is determined by the goal-achievement distinctions they support and the cost of retaining them.

**Constructing and reusing symbolic vocabularies.** Action-model induction and neuro-symbolic planning learn predicates and operator schemas that support planning [60, 14, 55, 53, 12]. LLM-based agents similarly construct PDDL domains, executable world models, or reusable skills from interaction [54, 24, 56, 58]. Program-induction systems such as EC and DreamCoder jointly solve tasks and learn abstractions that compress programs and improve subsequent search [16, 62]. These approaches establish that representational vocabularies can be learned. Their objectives usually emphasize planning performance, sample efficiency, or compression over a task corpus. We focus on retention where an element persists when execution shows that it improves goal achievement given the current library and its representational cost.

**Behavioral coverage and persistent competence.** Classical empowerment measures controllable diversity through the mutual information between actions and successor states [50, 43]. Skill discovery and option construction seek repertoires that span the controllable state space [22, 17, 40, 46], while goal-conditioned and autotelic agents organize exploration around achieved-goal coverage [48, 13]. These objectives select policies, skills, or goals within an existing representation. Representational empowerment instead evaluates changes to the vocabulary used to model and plan. Parameter-level continual-learning methods preserve knowledge through regularization, architectural isolation, or replay [31, 52, 49]; such mechanisms can stabilize components beneath a symbolic library, while library curation determines which reusable elements the system retains.

**Cognitive foundations.** Our proposal draws on resource-rational accounts of representation choice [37], constructivist models of causal learning [7], and the "child as scientist" tradition [21]. It is also complementary to model-synthesis accounts in which distributed knowledge is assembled into a task-specific probabilistic model [59]. Representational empowerment supplies an operational hypothesis for valuing the candidate elements used in that assembly and deciding which should persist across tasks.

# B Representational Empowerment

This section distinguishes exact representational empowerment, its goal-projected channel, and the cost-regularized coverage proxy used in the open-vocabulary experiment. We show that projection retains only distinctions visible through the selected goal-achievement signature and that, under stated conditions, the proxy selects a minimum-cost goal-sufficient model. The proxy is an empirical decision rule, not a numerical estimate of channel capacity.

## B.1 Problem setup

The agent encounters environments $e_1, e_2, \ldots \sim \mathcal{E}$, each generated by an inaccessible process $W_e$. A candidate space $\Sigma$ contains representational elements such as observation granularities, causal edges, predicates, and operator schemas. In environment $e$, the agent constructs a task model $M_e \subseteq \Sigma$ and

carries a library $\mathcal{L} \subseteq \Sigma$ across environments. Construction edits $M_e$ within an environment; curation edits $\mathcal{L}$ across environments. We represent both as transitions of the internal state $Z = (M, \mathcal{L})$.

### B.2 Representation-state channels

Let $\mathcal{Z}$ be the space of representation states and $\Omega$ a finite set of edits, such as ADD, REMOVE, REFINE, COMPOSE, and PRUNE. An edit sequence $u = \omega_{1:T} \in \Omega^T$ induces the kernel

$$K_T(z' \mid z, u) = \Pr(Z_T = z' \mid Z_0 = z, U = u),$$

conditional on the current environment and evidence.

**Definition 1** (Exact representational empowerment)**.** *The $T$-step representational empowerment of state $z$ is*

$$\mathrm{RepEmp} = \mathcal{R}_T(z) = \sup_{\pi \in \Delta(\Omega^T)} I_{\pi K_T}(U; Z_T \mid Z_0 = z),$$

*where $U \sim \pi$ and $Z_T \sim K_T(\cdot \mid z, U)$.*

This is the internal analogue of environmental empowerment [3, 61]: edits, rather than actions, index controllable outcomes. Since

$$I(U; Z_T \mid Z_0 = z) = H(Z_T \mid Z_0 = z) - H(Z_T \mid U, Z_0 = z),$$

$\mathrm{RepEmp}$ requires future representations to be both diverse and reliably attributable to particular edits.

### B.3 Grounding by goal projection

Exact empowerment distinguishes all future representation states, including syntactic variants with identical behavioral consequences. We therefore observe a task model through its goal-achievement signature

$$\Gamma_e(M) = \big(\mathbb{I}[g \text{ is achieved by planning with } M \text{ and executing in } e]\big)_{g \in \mathcal{G}_e}.$$

**Definition 2** (Grounded representational empowerment)**.** *The grounded representational empowerment of state $z$ in environment $e$ is*

$$\mathcal{R}_T^{\Gamma}(z; e) = \sup_{\pi \in \Delta(\Omega^T)} I_{\pi K_T}(U; \Gamma_e(Z_T) \mid Z_0 = z)\,.$$

**Proposition 1** (Projection lower bound)**.** *For every deterministic projection $\Gamma_e$,*

$$\mathcal{R}_T^{\Gamma}(z; e) \le \mathcal{R}_T(z).$$

*Proof.* Conditional on $Z_0 = z$, $U \to Z_T \to \Gamma_e(Z_T)$ is a Markov chain. The result follows from data processing for each $\pi$ and then taking the supremum. □

The inequality can be strict. Suppose $n$ edits deterministically produce distinct representations $z_1, \ldots, z_n$ but all have the same signature. The exact channel has capacity $\log n$, whereas the grounded channel is constant and has capacity zero. Thus the projection excludes purely syntactic diversity. When two representations have the same signature, the cost term introduced below selects the cheaper one.

### B.4 From channel capacity to the coverage proxy

Define the coverage of model $M$ by

$$\mathcal{F}_e(M) = \sum_{g \in \mathcal{G}_e} \Gamma_e(M)_g.$$

For a modeling action $a$ that yields observation $o$ and update $U(M_t, a, o)$, the expected cost-regularized gain is

$$\widehat{V}_t(a) = \mathbb{E}_{o \sim p_t(\cdot \mid a)}[\mathcal{F}_e(U(M_t, a, o)) - \mathcal{F}_e(M_t)] - \lambda \Delta \mathcal{C}_t(a).$$

For an unvalidated candidate $\sigma$, let $q_t(\sigma)$ be its estimated validity and $W_t(\sigma)$ the goals currently blocked under $M_t$ that $\sigma$ could unlock. The open-vocabulary implementation uses

$$\widehat{V}_t(\sigma) \approx q_t(\sigma) \sum_{g \in W_t(\sigma)} [\widehat{p}_t(g \mid M_t \cup \{\sigma\}) - \widehat{p}_t(g \mid M_t)] - \lambda c_\sigma,$$

where $\widehat{p}_t$ is the Actor's empirical achievement rate. Restricting the sum to $W_t(\sigma)$ removes goals whose achievement bit cannot change.

This proxy preserves the local distinction relevant to the grounded channel. In a deterministic channel with inputs KEEP and ADD($\sigma$), capacity is zero exactly when adding $\sigma$ leaves $\Gamma_e$ unchanged; any changed achievement bit makes capacity positive. The proxy scores the magnitude and reliability of such changes rather than estimating capacity directly.

For a multi-element proposal, the exact coverage gain decomposes into conditional marginal gains under any ordering. The implementation approximates it by summing singleton gains, which is exact for independent candidates with disjoint witness sets. Redundancy can cause overestimation and complementarity can cause underestimation, so proposed bundles are verified jointly before acceptance. Higher-order representational synergy remains a limitation of the singleton approximation.

### B.5 Minimum-cost sufficiency for the coverage proxy

**Definition 3** (Goal sufficiency)**.** *A model $M$ is goal-sufficient in environment $e$ if*

$$\mathcal{F}_e(M) = \mathcal{F}_e^\star := \max_{M' \subseteq \Sigma} \mathcal{F}_e(M').$$

*It is minimum-cost goal-sufficient if no goal-sufficient model has lower cost.*

Assume additive positive costs,

$$\mathcal{C}(M) = \sum_{\sigma \in M} c_\sigma, \qquad c_\sigma > 0.$$

**Theorem 1** (Minimum-cost goal sufficiency)**.** *Assume $\Sigma$ is finite and not every $M \subseteq \Sigma$ is goal-sufficient. Let*

$$C_{\max} = \sum_{\sigma \in \Sigma} c_\sigma, \qquad \Delta_e = \min_{M:\, \mathcal{F}_e(M) < \mathcal{F}_e^\star} \big(\mathcal{F}_e^\star - \mathcal{F}_e(M)\big) > 0.$$

*If $0 < \lambda < \Delta_e / C_{\max}$, every maximizer of*

$$\mathcal{F}_e(M) - \lambda \mathcal{C}(M)$$

*is a minimum-cost goal-sufficient model.*

*Proof.* Let $M$ be suboptimal in coverage and let $\widetilde{M}$ be any goal-sufficient model, which exists because $\Sigma$ is finite. Their objective difference in favor of $\widetilde{M}$ is

$$\mathcal{F}_e^\star - \mathcal{F}_e(M) - \lambda\big(\mathcal{C}(\widetilde{M}) - \mathcal{C}(M)\big) \geq \Delta_e - \lambda C_{\max} > 0.$$

Hence no coverage-suboptimal model maximizes the objective. Among goal-sufficient models the score is constant, so maximizing the objective minimizes cost. □

The theorem is a consistency result for the finite proxy under additive costs, exact signatures, and a sufficiently small $\lambda$; it does not claim that the agent knows $\Delta_e$ or that the proxy uniquely formalizes resource-rational construction.

**Definition 4** (Goal-irrelevant element)**.** *An element $\sigma \in \Sigma$ is goal-irrelevant in $e$ if*

$$\mathcal{F}_e(M \cup \{\sigma\}) = \mathcal{F}_e(M) \qquad \text{for all } M \subseteq \Sigma \setminus \{\sigma\}.$$

**Corollary 1** (Pruning of goal-irrelevant elements)**.** *No maximizer in Theorem 1 contains a goal-irrelevant element.*

*Proof.* Removing such an element preserves coverage and strictly reduces cost. □

In the causal experiments, a variable or edge outside every intervention-to-goal path is goal-irrelevant when the score depends only on intervention-to-goal reachability. Learning it may reduce uncertainty about $W_e$, but retaining it does not change the goal-achievement signature.

### B.6 Behavioral predictions for model construction

**Proposition 2** (Refinement during discovery, compression after sufficiency)**.** *There exists a two-stage construction problem in which* `fine` *has greater expected proxy value during discovery, whereas a coarser representation has greater value after the relevant structure is learned.*

*Proof.* Let $M_f$ and $M_c$ be fine and coarse representations with $\mathcal{C}(M_f) > \mathcal{C}(M_c)$. During discovery, let a distinction available only under $M_f$ enable an expected score gain larger than its additional cost; then $M_f$ has greater proxy value. After the relevant relation is learned, let $\mathcal{F}_e(M_f) = \mathcal{F}_e(M_c) = \mathcal{F}_e^\star$; the proxy then prefers the cheaper $M_c$. □

The prediction is stage-dependent rather than a fixed preference for either resolution: fine representations can expose useful distinctions before the relevant structure is known, while the same goal coverage may later be retained more cheaply. This is the pattern tested by the within-task granularity analysis in Exp. 1.

### B.7 Relation to alternative objectives

**Information gain.** For an edit $\sigma$ producing observation $o$ about an unknown model variable $M$, information gain is

$$IG(\sigma) = H(M) - \mathbb{E}_{o\sim p(o|\sigma)}[H(M \mid o)].$$

It rewards uncertainty reduction whether or not the resolved distinction affects achievable goals. Thus two tests can have equal information gain while only one changes $\Gamma_e$; grounded $\mathrm{RepEmp}$ distinguishes them through their projected outcomes.

**Environmental empowerment.** Environmental empowerment measures the capacity of the action-to-state channel, $\sup_\rho I_\rho(A_{1:T}; S_T \mid S_0 = s)$, with the agent's representation fixed. It can therefore be large when many physical states are controllably reachable but the model lacks a relation needed for a particular goal. Representational empowerment instead makes edits the channel inputs, and its grounded form retains only their consequences for $\Gamma_e$. The objectives accordingly govern different decisions: $\mathrm{EnvEmp}$ selects how to act within a representation, whereas $\mathrm{RepEmp}$ evaluates changes to the representation itself.

## C Causal Learning Environment

Exp. 1 and Exp. 2 use the same causal-learning environment with different graphs, curricula, and observation-noise regimes. The environment is adapted from Ke et al. [30] and extends standard causal discovery by allowing the learner to choose the granularity at which intervention effects are represented.

### C.1 Shared environment

**Variables and state.** The environment instantiates a linear SEM over a set of latent variables $\mathcal{V} = \{V_1, \ldots, V_n\}$; the size and connectivity of $\mathcal{V}$ are experiment-specific (Secs. C.2 and C.3). Each variable carries a *fundamental* scalar value $x \in \{1, 2, \ldots, 16\}$ and three property *roles* (`strong`, `weak`, and `noise`). Only `strong` and `weak` have causal mechanisms, while `noise` is a distractor role so that participants and agents must discriminate causally relevant roles from irrelevant ones. In Exp. 1, the mapping from surface labels (e.g., "color," "temperature") to property roles is randomized across participants and held fixed within a participant. This prevents surface-label familiarity—such as a prior on semantic association—from biasing which properties are probed first or trusted more. In Exp. 2, role identity is fixed across seeds since the simulated agent has no priors to control for.

**Representation granularities.** Each fundamental value $x$ is observed through one of three discretization maps:

$$\phi_{\text{fine}}(x) = \lceil x/2 \rceil \in \{1, \ldots, 8\}, \quad \phi_{\text{medium}}(x) = \lceil x/4 \rceil \in \{1, \ldots, 4\}, \quad \phi_{\text{coarse}}(x) = \begin{cases} \text{Low}, & x \le 8, \\ \text{High}, & x > 8. \end{cases}$$

Table 4: Ground-truth SEM for Exp. 1. Eight latent variables, nine directed edges with weights and signs. Edges 1-8 appear in at least one task window (Tab. 5); edge 9 ($V_3 \rightarrow V_7$) is never jointly observable.

| # | Edge | Weight | Sign |
|---|---|---|---|
| 1 | $V_1$.strong $\rightarrow$ $V_2$.strong | 0.70 | + |
| 2 | $V_1$.strong $\rightarrow$ $V_3$.strong | 0.60 | + |
| 3 | $V_2$.strong $\rightarrow$ $V_4$.strong | 0.70 | + |
| 4 | $V_3$.strong $\rightarrow$ $V_4$.strong | 0.50 | + |
| 5 | $V_4$.strong $\rightarrow$ $V_5$.strong | 0.70 | + |
| 6 | $V_5$.strong $\rightarrow$ $V_6$.weak | 0.85 | + |
| 7 | $V_5$.strong $\rightarrow$ $V_7$.weak | $-0.70$ | − |
| 8 | $V_6$.strong $\rightarrow$ $V_8$.strong | 0.85 | + |
| 9 | $V_3$.strong $\rightarrow$ $V_7$.weak | 0.50 | + |

Table 5: Task curriculum for Exp. 1. Four tasks share variables across consecutive windows to enable cross-task transfer. Each task exposes a different structure (fork, merge, chain) and provides two planning goals reachable in $\leq 5$ steps.

| Task | Variables $\mathcal{V}_t$ | Visible edges | Structure |
|---|---|---|---|
| $T_1$ | $\{V_1, V_2, V_3\}$ | $V_1 \rightarrow V_2,\ V_1 \rightarrow V_3$ | Fork |
| $T_2$ | $\{V_2, V_3, V_4, V_5\}$ | $V_2 \rightarrow V_4,\ V_3 \rightarrow V_4,\ V_4 \rightarrow V_5$ | Merge + chain |
| $T_3$ | $\{V_4, V_5, V_6, V_7\}$ | $V_4 \rightarrow V_5,\ V_5 \rightarrow V_6\,(+),\ V_5 \rightarrow V_7\,(-)$ | Fork with negative edge |
| $T_4$ | $\{V_5, V_6, V_7, V_8\}$ | $V_5 \rightarrow V_6\,(+),\ V_5 \rightarrow V_7\,(-),\ V_6 \rightarrow V_8$ | Deep chain |

The granularity $\ell \in \Lambda = \{\texttt{coarse}, \texttt{medium}, \texttt{fine}\}$ is chosen by the learner on every step and determines only how effects are reported; the underlying propagation in the SEM always uses fundamental values.

**Mechanisms.** All edges in the SEM are linear with additive effect noise:

$$x_{\text{child}} \;=\; e_{\text{child}} + w\, x_{\text{parent}} + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

The exogenous component $e_{\text{child}}$ is back-solved from the task's initial state so that pre-intervention observations are consistent with the SEM.

**Action space and task model.** On every step the learner selects an observation granularity $\ell \in \Lambda$, then, conditional on $\ell$, intervenes on a variable-property pair $(v, p) \in \mathcal{V}_T \times \mathcal{P}$, where $\mathcal{V}_T \subseteq \mathcal{V}$ are the variables task $T$ makes visible and $v.p$ denotes property $p$ of variable $v$ (Tabs. 4 and 5). The intervention sets $v.p$ to a new fundamental value, the mechanism above propagates it to every variable downstream of $v.p$, and the learner observes each affected variable through $\phi_\ell$. How the learner specifies that value is the one part of the action space the two experiments do not share (Secs. C.2 and C.3). Granularity is the representational element, so the candidate space here is $\Sigma = \Lambda$, whereas the intervention acts on the environment and leaves the representation unchanged, and Sec. D.3 scores the two decisions under separate rules. The task model $M_t$ is the learner's belief graph after $t$ steps, a directed graph over visible variable-property pairs with confidence-weighted edges, reconstructed from each participant's observations in Exp. 1 (Sec. D.3) and maintained as a Bayesian posterior in Exp. 2 (Sec. E.1).

### C.2 Experiment 1 instantiation: human experiment

The human experiment uses a fixed eight-variable and nine-edge causal graph. The full edge list, including mechanism class, weight, and sign, is given in Tab. 4. The four-task curriculum with visible edges and goals is given in Tab. 5; tasks share variables across the curriculum so that overlapping windows expose the same underlying SEM through different visible subsets.

Participants never see fundamental values, so they specify an intervention in the bins of the selected granularity. Each step moves the chosen property one bin up or down from its current value at $\ell$, and the environment sets the underlying fundamental value to the midpoint of the target bin.

### C.3 Experiment 2 instantiation: simulation

Exp. 2 replaces the fixed eight-variable causal graph with a ten-variable graph randomly sampled per seed. For each of 30 seeds we draw a SEM over $n_{\text{var}}$=10 variables with edge probability

$p_e$=.25, instantiated under the same SEM mechanism (linear edges, additive Gaussian noise) and observed through the same $\phi_\ell$ maps as Exp. 1. This avoids any dependence of the findings of the granularity-intervention mechanisms on a single hand-designed graph.

Each seed's curriculum is generated procedurally: a sequence of 5 training tasks followed by 5 held-out test tasks, each task exposing an overlapping window of 4–7 variables drawn from the seed's graph. Visible edges in a task are the induced subgraph on its visible variables. Consecutive tasks share variables and edges, supporting the same cross-task transfer analyses as the human curriculum. Full per-task generation parameters are in Sec. E.

# D Experiment 1 Supplementary Results

## D.1 Participants and procedure

We recruited $n$=70 adults through Prolific (50% female; age 18-40, $M$=30.68, $SD$=6.27). Participants provided informed consent and were compensated above the local minimum wage, calibrated to the median completion time. The study was approved by the Institutional Review Board at the University of California, Berkeley. After instructions and a worked example, participants completed four tasks and a debrief. Each task comprised up to 20 exploration steps, a causal-graph report, and two goal-directed tests. At each exploration step, participants selected a granularity $\ell$ and then a variable-property intervention $(v, r)$; they observed only the resulting representation at $\ell$.

We logged granularity choices, intervention choices, observed effects, response time, reported graphs, and test performance. The analyses use 3,299 valid exploration decisions. Model comparisons retain steps on which all alternatives were admissible and every rule returned a finite score under the participant's current belief state.

## D.2 Behavioral analyses

**Choice persistence.** Three features of granularity choice motivate the control model that the criterion comparison scores against (Sec. D.3). Participants repeat the preceding granularity on 67.2% of consecutive within-task steps, compared with 46.4% under a participant-level shuffle that preserves how often each participant uses each granularity but discards the order ($p$<.001). $k$-means on participant granularity shares identifies balanced, medium-preferring, and fine-preferring policies, and split-half membership agrees for 51.4% of participants against a 39.2% shuffled baseline ($z$=2.09, $p$=.036), so participants differ stably in which granularity they favor. A multinomial logit predicting the next granularity shows that the preceding granularity dominates the latest visible intervention effect, contributing +619.8 held-out log-likelihood units ($\chi^2$=1239.7, coefficient +1.34) against +14.3 ($\chi^2$=28.6); Fig. 6 ranks every predictor in that model. Granularity choice is therefore persistent and largely insensitive to the most recent observation, even though the aggregate distribution still shifts within and across tasks, so the criterion comparison credits a rule only with what it adds beyond stickiness and participant-level preference.

**Compression within and across tasks.** Granularity use compresses within a task but converges across tasks, and the two patterns move `coarse` in opposite directions. Within a task, participants who return to a source they have already probed view it at a coarser granularity more often than a finer one (557 vs. 430, two-sided binomial test, $p$<.001); splitting each task at its median step, `coarse` use rises from 17.7% to 22.3% and `fine` use falls from 38.3% to 34.5%, while `medium` use is flat (44.0% to 43.2%). Across tasks the direction of the `coarse` trend reverses: comparing each participant's first and last task, `medium` use rises from 35.0% to 45.9% ($t(69)$=2.83, $p$=.006) and `coarse` use falls from 24.7% to 16.6% ($t(69)$=−2.86, $p$=.006), while `fine` use does not change reliably (40.3% to 37.5%, $t(69)$=−0.67, $p$=.50). This is the stage-dependence predicted by Proposition 2. Detail acquired during discovery is dropped once the local structure is known, whereas across tasks participants settle on the cheapest representation that remains goal-sufficient which `coarse` is not. Taskwise trajectories are in Fig. 7.

**Granularity use and goal performance.** The correlations reported in Sec. 4.1 are between-participant associations across the 70 participants; `coarse` use predicts neither measure ($r$=.10, $p$=.43 for achievement rate; $r$=−.14, $p$=.27 for steps per goal). To distinguish participants' overall `medium` use (the between-participant component) from each task's deviation from that participant's average (the within-participant component), we fit a Mundlak logistic mixed model over the 560

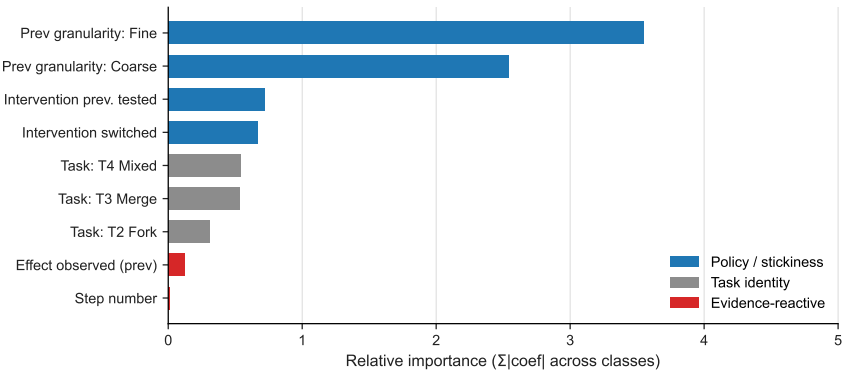


Figure 6: **Predictors of granularity choice.** Relative importance of each predictor in a multinomial logit over the three granularities, measured as the summed absolute coefficient across the three classes. The preceding granularity outweighs every other predictor by roughly fourfold, task identity contributes moderately, and the evidence available from the previous step—whether that step produced a visible effect, and how far into the task it occurred—contributes least.

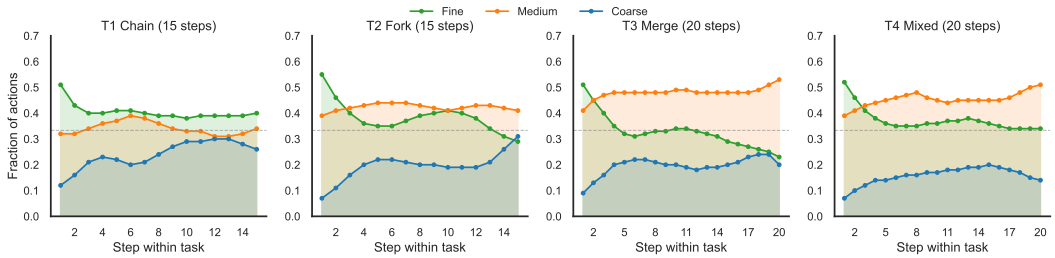


Figure 7: **Granularity use by task and step.** Across the four tasks, `medium` use increases while `coarse` use decreases. Within a task, `fine` use decreases and `coarse` use increases from the first to the second half of exploration.

binary goal outcomes, with task fixed effects and a participant random intercept. Overall `medium` use was positively associated with goal achievement ($\beta$=1.29, SE=.62, $z$=2.07, $p$=.039), whereas the task-specific deviation was positive but unreliable ($\beta$=.59, SE=.58, $z$=1.01, $p$=.31).

**Effort and response time.** Across 3,270 valid deliberation intervals, median RT is nearly identical for `coarse`, `medium`, and `fine` choices (7.6/7.6/7.3 s). The `fine-medium` contrast on log-RT, computed within participants by mean-centering each participant's log-RTs before comparison, shows no reliable difference ($\beta = -0.003$, $p$=.94). Entered into the criterion comparison of Sec. D.3, a participant-residualized RT score adds no held-out fit ($-0.16$ $[-0.66, +0.39]$) and leaves the RepEmp increment essentially unchanged when the two are fit jointly ($+12.16$). These tests do not support effort as the source of the granularity pattern.

### D.3 Criterion comparison

**Belief-graph reconstruction.** For each participant, we reconstruct the belief graph $M_t$ online from that participant's observations. The prespecified update uses detection threshold $\theta$=0, one confirmation per edge, and attributes all visible effects to the intervened variable. No scoring rule observes the ground-truth graph. We vary these assumptions in the specification curve below.

**Scoring the granularity decision.** We score the granularity and intervention decisions separately because they have different action spaces. For granularity, RepEmp is the grounded channel of Eq. (2) on the fixed probe-goal set,

$$\mathrm{RepEmp}_\ell(\ell) = I(U; \Gamma_{\mathcal{G}}(\ell) \mid M_t)\,.$$

A candidate intervention $U$ achieves a goal when its predicted outcome under $M_t$ lands within the goal's bin width; only goals expressible at $\ell$ contribute. Thus, additional resolution cannot increase

the score beyond the task's goal set. $\mathrm{EnvEmp}_\ell$ instead counts distinguishable commandable states without projecting onto goals,

$$\mathrm{EnvEmp}_\ell(\ell) = \log_2 \sum_u B(\ell)^{|\mathrm{reach}(u;M_t)|},$$

where $B(\ell) \in \{2, 4, 8\}$. The InfoGain rule scores the probability of detecting unresolved structure times its remaining posterior entropy.

**Scoring the intervention decision.** $\mathrm{EnvEmp}_{(v,p)}$ scores expected downstream reach from $(v, p)$, including prior-weighted reach through untested edges. $\mathrm{RepEmp}_{(v,p)}$ scores the expected increase in plannable goals if the intervention is tested. InfoGain sums the entropy of unknown outgoing edges.

**Control and fitting.** The hierarchical control contains group-level choice preferences, shrunken participant deviations, previous-choice stickiness, and, for intervention choice, intervention familiarity. It gains $+928$ nats over uniform for granularity and $+475$ for intervention choice. Candidate utilities are therefore evaluated only by the incremental fit they add to this control. Each utility enters a regularized softmax ($\ell_2$=2) with one population coefficient and is fit by leave-one-task-out cross-validation. We report held-out $\Delta$LL beyond this control (Fig. 4c).

**Specification curve.** Across 12 belief-update specifications (detection threshold $\times$ edge confirmations $\times$ effect attribution), fixed-goal $\mathrm{RepEmp}_\ell$ is the strongest granularity rule in a plurality of cells, but its median increment is near zero and no positive interval persists across the curve. Rules whose goal set grows with resolution are uniformly negative. The $\mathrm{EnvEmp}$ intervention point estimate is stable to a $16\times$ range of control shrinkage and a time-in-task control, but remains inferentially inconclusive.

# E Experiment 2 Supplementary Results

## E.1 Simulation protocol

The graph and curriculum for each seed are specified in Sec. C.3. Each task permits 1,000 interventions (10 episodes of 100 steps) and the agent carries its beliefs across task boundaries. The agent uses the same Bayesian belief tracker in every condition; an edge is counted as learned when its posterior exceeds .5. Each condition pairs a granularity rule with an intervention rule, drawn from $\mathrm{RepEmp}$, $\mathrm{EnvEmp}$, InfoGain, and uniform random; the Factored condition pairs $\mathrm{RepEmp}$ granularity with $\mathrm{EnvEmp}$ intervention selection. Observation noise is parameterized as $(\sigma_{\texttt{coarse}}, \sigma_{\texttt{medium}}, \sigma_{\texttt{fine}})$. We evaluate flat $(0.5, 0.5, 0.5)$, R-`coarse` $(0.5, 2.0, 3.0)$, R-`medium` $(2.0, 0.5, 3.0)$, and R-`fine` $(2.0, 2.0, 0.5)$. R `medium` is the default regime in the main comparison.

At each logged step, we compare the belief graph with the ground-truth graph using F1 and normalized SHD. The ground-truth edge set is projected through the condition's discretization $\phi_\ell$; we never up-project a learned graph to unsupported detail. Conditions, seeds, and tasks are fully crossed, yielding 150 paired seed–task instances per split. Contrasts use paired $t$-tests with Bonferroni correction ($\alpha$=.05/9); uniform random is the reference floor.

## E.2 Granularity selection across noise regimes

We validate each regime without consulting an adaptive selection rule. Structural identifiability measures the fraction of true edges for which an intervention produces an observational distribution distinguishable from the no-edge distribution at the available per-intervention budget. As a behavioral cross-check, fixed-granularity agents select interventions uniformly. The two criteria agree on the graph-recovery optimum in each regime (Tab. 6). This validation concerns graph recovery; it does not guarantee agreement with $\mathrm{RepEmp}$, which scores commandable goal outcomes.

Tab. 7 reports the share of decisions each rule spends at each granularity. The diagnostic positive result is the R-`medium` shift, where $\mathrm{RepEmp}$ moves from 97% `fine` in the flat regime to 63% `medium`, then returns to 97% `fine` in R-`fine`. $\mathrm{EnvEmp}$ remains fine-preferring, and InfoGain remains approximately uniform.

The shipped R-`coarse` regime is a negative result. Although fixed-policy graph recovery is highest at `coarse`, $\mathrm{RepEmp}$ selects `fine` on 91% of decisions and obtains .139 held-out F1, the divergence between graph recovery and commandable goal outcomes noted above. Its analytic preference for `coarse` requires $\sigma_{\texttt{medium}} > 3.52$ when $\sigma_{\texttt{coarse}}$=0.5; the shipped value is 2.0.

Table 6: Regime validation. Bold entries identify the best granularity under structural identifiability and fixed-policy graph recovery.

| | Identifiable edges (%) | | | Fixed-granularity F1 | | |
|---|---|---|---|---|---|---|
| Regime | `coarse` | `medium` | `fine` | `coarse` | `medium` | `fine` |
| flat | 96.0 | 99.7 | **100.0** | .520 | .708 | **.753** |
| R-`coarse` | **96.0** | 67.2 | 19.5 | **.520** | .370 | .185 |
| R-`medium` | 49.4 | **99.7** | 19.5 | .343 | **.708** | .185 |
| R-`fine` | 49.4 | 67.2 | **100.0** | .343 | .370 | **.753** |

Table 7: Share of `coarse`/`medium`/`fine` decisions by rule and regime; the modal level is bold. The final row gives the independently validated graph-recovery optimum.

| Rule | flat | R-`coarse` | R-`medium` | R-`fine` |
|---|---|---|---|---|
| RepEmp | .02/.02/**.97** | .02/.08/**.91** | .02/**.63**/.35 | .02/.02/**.97** |
| EnvEmp | .02/.02/**.97** | .02/.02/**.97** | .02/.02/**.97** | .02/.02/**.97** |
| InfoGain | .34/.33/.33 | .34/.33/.33 | .34/.33/.33 | .33/.33/.33 |
| Graph-recovery optimum | `fine` | `coarse` | `medium` | `fine` |

Post hoc sweeps locate, but do not remove, this boundary. Raising $\sigma_{\texttt{medium}}$ to $4.02$ increases the full-budget coarse share only to $23\%$. At the same noise and a budget of 20, the coarse share reaches $59\%$ (modal in $20/30$ seeds), but F1 falls to .176, versus .326 at the full budget. We treat these sweeps as exploratory diagnostics, not evidence that RepEmp robustly tracks coarse-optimal graph-recovery regimes.

# F Grid World Environments and Curator-Actor Architecture

This section documents the open-vocabulary experiments referenced in Sec. 5.1: the Curator-Actor architecture, the two environment families (BabyAI and Zelda), and the training and evaluation protocols. The implementation accompanying these experiments will be released with the paper.

## F.1 Curator-Actor architecture

**Library and task model.** The persistent library $\mathcal{L}$ is a typed PDDL knowledge store with types, type-hierarchy edges, predicates, and operators. Types are atomic symbols (e.g., `agent`, `key`, `door`, `object`) and type-hierarchy edges record subtype relations (`key - object`, `door - object`). Predicates carry a PDDL signature and a Python grounding closure that evaluates the predicate on raw observations:

```
(:predicates
  (holding ?a - agent ?o - object)
  (next_to ?a - agent ?o - object)
  (locked  ?d - door)
  (is_key_for ?k - key ?d - door))
```

Operators are PDDL action schemas with typed preconditions and effects:

```
(:action unlock_door
  :parameters (?a - agent ?k - key ?d - door)
  :precondition (and (holding ?a ?k)
                     (next_to ?a ?d)
                     (locked  ?d)
                     (is_key_for ?k ?d))
  :effect       (and (not (locked ?d))))
```

Every element carries a status flag in $\{\texttt{candidate}, \texttt{trusted}\}$. The task-local model $M_t$ is the PDDL domain assembled from the `trusted` subset of $\mathcal{L}$ relevant to the current environment, plus any `candidate` elements currently under evaluation, plus the grounding code needed to evaluate predicates on raw observations.

**Curator.** The Curator $\mathcal{C}$ is the LLM-backed component. It runs the following loop:

**Algorithm 1** The Curator-Actor loop for continual model construction

**Input:** environment $e$, library $\mathcal{L}$, proposal vocabulary $\Sigma$, budgets $B_{\text{task}}, B_{\text{lib}}$
$\mathcal{D} \leftarrow \varnothing$; $M_0 \leftarrow \mathcal{C}.\textsc{Seed}(\mathcal{L}, e)$ *// initial task model from $\mathcal{L}$*
**for** $t = 1, \ldots, B_{\text{task}}$ **do**
  $(M_t, \mathcal{G}_t, \{\pi_g\}_{g \in \mathcal{G}_t}) \leftarrow \mathcal{C}.\textsc{Propose}(\mathcal{L}, o_e, \mathcal{D};\ M_{t-1}, \Sigma)$ *// task model, probe goals, symbolic plans*
  **for** each $g \in \mathcal{G}_t$ **do**
    $(\tau_g, \Gamma_e(M_t)_g) \leftarrow \mathcal{A}.\textsc{Execute}(M_t, g, \pi_g)$ *// primitive grounding only*
  **end for**
  $\widehat{V}_t(\Delta M_t) \leftarrow \mathcal{C}.\textsc{Score}\big(\Gamma_e(M_t), \Gamma_e(M_{t-1}); \mathcal{G}_t\big)$, where $\Delta M_t = M_t \setminus M_{t-1}$ *// Eq. (3), summed over $\sigma \subseteq \Delta M_t$*
  **if** $\widehat{V}_t(\Delta M_t) > 0$ **then** accept $M_t$, append $\{\tau_g\}$ to $\mathcal{D}$; **else** $M_t \leftarrow M_{t-1}$ and reject the candidate elements (quarantine)
  **if** $\Gamma_e(M_t)$ has plateaued on the probe set **then**
    Backward-prune any element of $M_t$ whose removal leaves $\Gamma_e(M_t)$ unchanged
  **end if**
**end for**
Re-evaluate each retained element's $\widehat{V}_t$ on the cross-environment probe buffer (regime-appropriate estimator)
$\mathcal{L} \leftarrow \mathcal{C}.\textsc{GreedyCurate}\big(\mathcal{L} \cup M_t,\ B_{\text{lib}}\big)$ *// greedy marginal-coverage update*

1. *Propose.* From the experience buffer $\mathcal{D}_t$, the current $M_t$, and $\mathcal{L}$, an LLM proposer emits candidate types, predicates (with Python grounding code), and operator schemas. The proposer also revises existing elements (refine signature, split, compose) and emits top-environment goal predicates that are expressible under $M_t \cup \Sigma_t$ but not currently achievable under $M_t$.
2. *Probe.* For each candidate $a$, $\mathcal{C}$ constructs a probe goal set $\mathcal{G}_t(a)$: goals blocked under $M_t$ but expressible under $M_t \oplus a$ (positive probes) and goals already solvable that act as regression checks (negative probes). For each $g \in \mathcal{G}_t(a)$, $\mathcal{C}$ compiles $g$ into a PDDL problem against $M_t$ and calls Fast Downward [28]. When the PDDL compile fails (e.g., a candidate operator has malformed typing) or the external solver times out, $\mathcal{C}$ falls back to a fuzzy beam-search planner that operates directly on the operator schemas in $M_t$. In both cases the output is an operator sequence $\pi_g^{M_t} = (a_1, \ldots, a_k)$, which is handed to the Actor along with $g$.
3. *Score.* Candidates $a$ are evaluated with Eq. (3). The Actor estimates the change in achievement probability on the witness set $W_t(a)$; candidate validity weights this gain, and the representational penalty determines whether the marginal benefit is large enough for promotion. To save Actor calls, candidates are first pre-ranked by a cheap surrogate computed from $\mathcal{L}$, $\mathcal{D}_t$, and the candidate's typed signature. Only top-ranked candidates are executed; the resulting signatures $\Gamma_e(M_t \oplus a)_g$ and $\Gamma_e(M_t)_g$ determine the promotion score.
4. *Curate.* Candidates that meet minimum thresholds on $\widehat{V}_t$, on reliability (success rate across probe attempts), and on a minimum test count are promoted to `trusted`. At the cross-environment refactor stage described in Sec. 3, the same $\widehat{V}_t$ is re-evaluated against the cross-environment probe buffer—which samples a fixed budget of imagined state-goal pairs from the buffer and scores reachability under $\mathcal{L}$ vs. $\mathcal{L} \cup \{\sigma\}$—and the trusted set is greedily re-ranked under the library budget by marginal $\widehat{V}_t$.

Between curriculum environments, the final library $\mathcal{L}$ from the previous environment is carried forward unchanged and seeds $M_t$ on the next environment.

**Actor.** The Actor receives a $(M_t, g, \pi_g^{M_t})$ triple from the Curator and returns evidence about $\Gamma_e(M_t)_g$. No LLM is involved at plan-and-execute time, and the Actor never plans symbolically: its job is to realize the operator sequence $\pi_g^{M_t} = (a_1, \ldots, a_k)$ in primitive actions and report whether each $a_i$ landed its effects. Each operator $a_i$ is realized as a primitive sequence by heuristic search over the agent's primitive action set. The search uses an agent-egocentric local world model: a small representation $(\text{patch}, \text{carried_slot})$ of the agent's neighborhood, where $\text{patch}$ is a fixed-radius window of cells rotated so the agent heading is up, and $\text{carried_slot}$ encodes the carried object outside the cell grid. Predicates that bind to in-window objects, plus agent-indexed predicates (e.g. `holding`, `facing`), are projected into the local model on demand by running their grounding closures. The search heuristic counts unmet operator preconditions under the projected predicates, a goal-count relaxation in the spirit of classical planning heuristics; we make no admissibility claim, and the

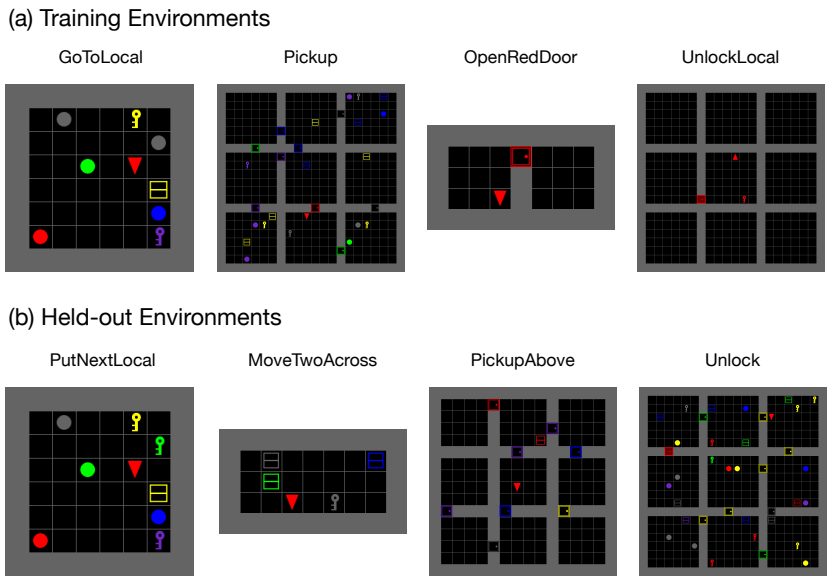


Figure 8: **BabyAI training and held-out environments used in Exp. 3.** Training environments are ordered by minimum operator-chain length; held-out environments combine those operators in configurations not seen during training.

environment verifies the resulting predicate flips. The local model is learned online: a self-supervised predictor maps $(\text{patch}, \text{carried_slot}, \text{primitive})$ to the local predicate-flip set and is updated from the Actor's primitive trajectories. Calls into the live environment during search are amortized by rollouts in this learned model.

### F.2 Environments

**BabyAI.** We use four BabyAI environments for training: `GoToLocal`, `Pickup`, `OpenRedDoor`, and `UnlockLocal`. All four share the same primitive action space (turn, move, pickup, drop, toggle). The ordering is intended to give the Curator opportunities to compose elements promoted at earlier environments (e.g., navigation routines reused for key acquisition before unlocking). A held-out split of four environments, never seen during training, tests transfer to configurations whose goal templates differ from the training environments: `PutNextLocal` introduces spatial object–object relations; `MoveTwoAcrossS5N2` requires multi-object manipulation; `PickupAbove` changes the relative orientation of agent and target; and `Unlock` is a variant of `UnlockLocal` with different layout and object constraints. Fig. 8 shows representative screenshots.

**Zelda (EMPA).** We use the *Zelda* game from the EMPA suite [57], which provides typed objects—player, monster, key, door, wall—and a native reward function. The first three environments are used for training and the last two are held out for evaluation (Fig. 9). Held-out environments share the primitive dynamics of training but introduce new layouts and precondition configurations, testing whether predicates and operators promoted during training (e.g., key acquisition, monster avoidance) transfer to settings where the same goals must be reached through different spatial arrangements.

### F.3 Training and evaluation

**Training budget.** Within an environment, the Curator-Actor loop runs until either the candidate set stabilizes—no new promotions for $K_{\text{stab}}$=3 consecutive cycles—or the per-environment LLM-call budget is exhausted. We allocate 200 LLM calls per environment on BabyAI and 400 on Zelda, matching WorldCoder's budget on each family for direct comparison. Motif's pipeline issues an LLM call for every labeled trajectory pair and consequently uses $\sim 5{,}000$ calls per environment; this is a structural cost of preference labeling, not a compute advantage. PPO has no LLM calls and is matched to Motif on environment-step budget instead.

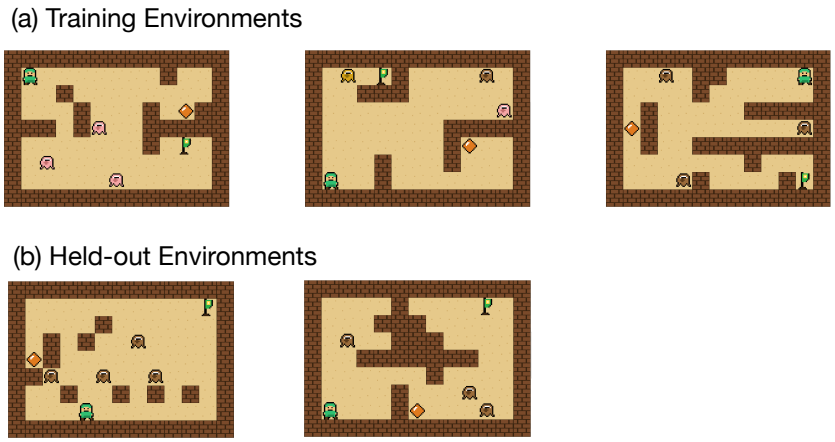


Figure 9: **Zelda training and held-out environments used in Exp. 3.** The first three environments are used for training and the last two for held-out evaluation; primitive dynamics are identical across environments but layouts and precondition configurations differ.

Crucially, the LLM-call budget translates into a small environment-step footprint. Curator and WorldCoder each consume on the order of thousands of primitive environment steps per environment—two orders of magnitude below the $\sim 500\text{k}$ and $\sim 1\text{m}$ steps used by Motif and by PPO—because each LLM call drives at most a handful of probe-goal executions rather than continuous policy rollout. We treat this gap as a property of symbolic methods, but flag it explicitly so the transfer comparisons are not read as unequal-compute results. Immediately after an environment finishes training, we record a *self-evaluation* score on the same environment using the then-current $\mathcal{L}$; this score anchors the backward-transfer comparison defined below.

All LLM-backed components (Curator's proposer, scorer, and curator; WorldCoder's synthesizer; Motif's labeler) use the same model: `deepseek-r1:70b` served locally via Ollama on a single A100 GPU. We report mean and standard deviation across 3 seeds per condition, with seeds controlling LLM sampling and environment stochasticity. Within a seed, evaluation is averaged over $N_{\text{eval}} = 50$ rollouts per (environment, stage) cell on BabyAI and Zelda. The success metric, $\text{reward_success}$, is the fraction of evaluation rollouts in which the agent reaches the environment's primary goal as defined by the environment's native reward function. A rollout terminates on goal achievement, agent death (Zelda), or step-limit ($T_{\max} = 256$ primitive steps).

**Promotion criteria.** A candidate element $\sigma$ is promoted to `trusted` when it has reliability at least .70, has been dispatched on at least three probes, and its unpenalized marginal coverage gain exceeds .50. With $c_\sigma \equiv 1$ and $\lambda = .50$, the final condition is equivalent to a positive penalized score in Eq. (3). At the environment boundary, $\widehat{V}_t$ is re-evaluated against a cross-environment probe buffer using the `imagined_goal` estimator. We sample $N_{\text{imag}} = 50$ state-goal pairs uniformly from the buffer, plan against $\mathcal{L}$ and $\mathcal{L} \cup \{\sigma\}$ for each, and set $\sigma$'s contribution equal to the number of newly reachable goals.

**Forward transfer.** For each adjacent pair $(\text{env}_i, \text{env}_{i+1})$ in the training curriculum, we freeze $\mathcal{L}$ at the end of $\text{env}_i$ and evaluate it on $\text{env}_{i+1}$ *without further library updates*: the agent receives no training trajectories on $\text{env}_{i+1}$ and the Curator does not propose new candidates. The local world model (Sec. F.1) continues to update from primitive trajectories, since this update happens at no extra environment budget. This isolates the short-range generalization of promoted symbols.

**Backward transfer.** After the full curriculum has been trained, we re-evaluate the final $\mathcal{L}$ on each earlier training environment under the same frozen-library protocol as forward transfer. Backward transfer is the difference between an environment's success rate immediately after it was trained and its success rate after all subsequent training has finished:

$$\text{Backward}_i \;=\; \text{final-eval}(i) - \text{self-eval}(i),$$

where self-eval$(i)$ is the score recorded immediately after $\text{env}_i$ was trained and final-eval$(i)$ is the score under the final $\mathcal{L}$. Negative values indicate forgetting; positive values indicate retroactive

improvement, where abstractions promoted at later environments make earlier environments easier. Rather than collapse this to a scalar, Figs. 10 and 11 plot the full success trajectory eval$(i, j)$ at every evaluation stage $j \geq i$, since the shape of the trajectory distinguishes monotonic accumulation (Curator) from peak-and-decay (WorldCoder, PPO).

**Held-out transfer.** For each environment family we hold out a disjoint set of environments not used during training. The frozen final $\mathcal{L}$ is evaluated zero-shot on each held-out environment under the same protocol as forward and backward transfer: no training trajectories, no candidate proposals, only library lookup and Actor execution. This tests compositional transfer to goal templates and configurations whose specific combinations did not appear in training.

### F.4 Prompt templates

The Curator-Actor pipeline issues two LLM calls per cycle to construct $M_t$, plus one call to propose a probe problem. Each Curator cycle first calls *Discover* to propose new actions, predicates, and fixes from recent trajectories, then calls *Formalize* to turn those proposals into typed PDDL schemas and Python grounding code. The split mirrors the conceptual distinction between *what to propose* (a generative step that draws on both observed trajectories and prior knowledge) and *how to express it formally* (a compilation step that enforces syntactic and type constraints). We show the domain-agnostic templates below; `{{...}}` marks runtime substitutions, and the full prompts (with environment-specific scaffolding for BabyAI position tuples, Zelda direction conventions, etc.) are released with the code.

**Construct stage 1 — Discover**

```
System.
You are an expert in symbolic abstraction and world-model learning.  Operate at a
high level of abstraction:  read low-level observations but lift them into abstract
symbolic descriptions suitable for task-level planning.  You may draw on both observed
evidence and prior knowledge of the domain; both are valid.

User.
Analyze the interaction experiences and propose high-level symbolic structure for
task-level planning.
EXISTING DOMAIN
EXPERIENCES:  {{trajectories}}
ENVIRONMENT METADATA:  {{env_metadata}}

TASK.
     1. Identify high-level patterns in state transitions.
     2. Propose new operators, predicates and types.
     3. If an existing symbol almost fits, propose a fix rather than a new symbol.

OUTPUT (strict JSON):
{
  "observed_patterns": [{"description": "...", "confidence": 0..1}],
  "proposed_actions": [
    {"name": "...", "parameters": [...],
     "preconditions_sketch": "...", "effects_sketch": "...",
     "evidence": {"type": "experience-driven"|"gap-driven", ...},
     "justification": "..."}
  ],
  "proposed_predicates": [{"name": "...", "meaning": "...", ...}],
  "fix_proposals":      [{"name": "...", "current_issue": "...",
                          "suggested_fix": "..."}]
}
```

**Construct stage 2 — Formalize**

```
System.
You are a compiler, not a generator:  stay faithful to what the patterns specify.
Your job is to convert discovered patterns into canonical structured artifacts--typed
predicate definitions, action schemas with declared parameters, and Python grounding
functions.
```

User.
CURRENT WORLD MODEL (status-annotated as in stage 1): {{current_domain}}
DISCOVERED PATTERNS (compile these faithfully): {{patterns_from_stage_1}}

TASK.

1. Formalize each entry in proposed_actions as a structured action schema with typed parameters, precondition, and effect.
2. Formalize each entry in proposed_predicates as a typed predicate definition.
3. Apply each entry in fix_proposals as a corrected definition with intent: "fix".
4. Generate a Python grounding function for each new or fixed predicate.

OUTPUT (strict JSON):

```
{
  "new_predicates": [
    {"name": "...", "parameters": [...], "meaning": "...",
     "intent": "new"|"fix"}
  ],
  "new_actions": [
    {"name": "...", "parameters": [...],
     "precondition": <expr>, "effect": <expr>,
     "intent": "new"|"fix"}
  ],
  "grounding_code": "Python source string"
}
```

**Propose-problem prompt**

System.
You are an expert in designing problems for an autonomous agent that builds and validates symbolic world models (PDDL domains).
Generate a single PDDL problem that serves the agent's current strategic intent. A problem can serve any of the following purposes simultaneously:

- Validate a hypothesis about a predicate or action (diagnostic / counterfactual).
- Build a skill (navigation, manipulation, etc.).
- Discover new symbolic structure.

User.
STRATEGIC INTENT: {{strategic_intent}}
FOCUS SYMBOLS (exercise these): {{focus_symbols}}
CURRENT DOMAIN: {{current_domain_pddl}}
ENVIRONMENT OBJECTS (pick from this list): {{available_objects}}

GUIDELINES.

- Use only object names from the environment list and only values from typed constants. Do not invent names.
- Include only objects participating in the goal.
- Express goal as a logical formula over predicates already declared in the domain, using atom, and, or, not, or seq nodes.

OUTPUT (strict JSON):

```
{
  "goal_name":   "...",
  "goal_description": "...",
  "why_useful":  "why this problem helps the strategic intent",
  "objects":     [{"name": "...", "type": "..."}],
  "init":        [],
  "goal":        <goal_expr>,
  "focus_symbols_exercised": [...],
  "success_criteria": "..."
}
```

### F.5 Baseline

**PPO.** We use Proximal Policy Optimization [51] with intrinsic motivation as a model-free continual RL baseline. The agent is trained with both extrinsic task reward and intrinsic novelty bonus from Random Network Distillation [RND; 8], which rewards visiting states whose features the predictor network cannot yet match. For intrinsic reward selection, to make a fair comparison, we varied three conditions: no intrinsic reward, count-based and RND-based intrinsic rewards. We showed the results of RND-based intrinsic rewards as it performs best across three conditions for PPO.

The policy network is a three-layer CNN (channels 16-32-64, kernel size 2) followed by an `AdaptiveAvgPool2d`$(4 \times 4)$ that produces a fixed 1024-dimensional feature regardless of input resolution, then a 64-unit fully connected layer feeding separate actor and critic heads. For both BabyAI and EMPA, the observation is the symbolic full-observation grid (object type, color index, state), with per-channel normalisation matching the integer ranges of each field. All PPO hyperparameters follow standard settings: clipped surrogate objective ($\epsilon = .2$), GAE advantage estimation ($\gamma = .99$, $\lambda = .95$), 4 optimisation epochs per rollout batch, and linear learning-rate annealing within each environment phase.

**Motif.** Motif [32] constructs an intrinsic reward by eliciting trajectory preferences from an LLM and training a reward model on those preferences. The pipeline runs four steps per curriculum environment and follows the same sequential continual-learning structure as PPO:

1. *Collect.* A fresh RND agent (independent of the Motif policy) explores the current environment and saves the resulting episodes.
2. *Label.* $5000$ trajectory pairs are sampled from those episodes. Each pair is converted to a text caption—listing the objects the agent sees, their distances, and the chosen action at each step—and sent to an LLM. The LLM returns a binary preference ($A > B$, $B > A$, or $A = B$) without access to the mission string.
3. *Train reward model.* An MLP reward model (3-layer CNN encoder matching the `IntrinsicAgent` backbone, 512-unit hidden layer, scalar output head) is trained from scratch on the labeled pairs using a Bradley-Terry preference loss.
4. *RL with intrinsic reward.* The `IntrinsicAgent`—a 3-layer CNN (32-64-64 channels, $3 \times 3$ kernels) with `AdaptiveAvgPool2d`$(4 \times 4)$ and 512-unit FC layer, feeding separate extrinsic and intrinsic critic heads—is trained with PPO. The reward model's scalar output is used as the intrinsic bonus, together with extrinsic task reward, for the training signal (mirroring PPO's intrinsic-only setup).

**Why Motif fails in the autotelic setting.** Motif was originally demonstrated on Crafter, where the LLM can judge behavioral quality from goal-agnostic signals such as health loss, resource gain, and crafting events without knowing the specific task. Our setting withholds the mission during training, since an agent that generalizes across open-ended task distributions must not be hard-wired to any particular goal [13], and knowledge acquired through task-specific reward shaping is goal-indexed rather than compositional. Motif's labeler therefore has no basis for ranking one trajectory above another, because every primitive action—turn left, pick up, do nothing—is potentially correct depending on a goal it cannot see. Its reward model achieves $50.2\%$ preference-prediction accuracy on a held-out validation split, indistinguishable from chance, so the intrinsic bonus it supplies carries no learning signal.

Inspecting the labeled pairs reveals the failure mode concretely. For example, in the `GoToLocal` phase, both segments share an *identical* observation; only the action differs:

> **Segment A:** *You see a yellow ball 3 steps left and 3 steps forward. You see a blue box 3 steps forward. You see a purple key 2 steps right and 2 steps forward.* [. . .] ***Action: do nothing.***
>
> **Segment B:** [identical observation] **Action: pick up the front object.**
>
> **LLM:** *"In Segment B, the agent attempts to pick up an object, which isn't necessary for a* ***goto task*** *and thus doesn't help. However, doing nothing (A) is preferable as it avoids an irrelevant action."* $\Rightarrow$ **[Rank] A > B**

The LLM hallucinates this is a *goto* task. If the actual mission were "pick up the grey key," Segment B is the correct behaviour and the label is backwards. Across the full labeled dataset, $78.8\%$ of LLM responses invoke task-specific language ("goal," "goto task," "mission") despite having no access to

Table 8: **Training-time LLM cost.** Total LLM usage during training for WorldCoder, the Curator ablation without empowerment-based scoring, and the full Curator. All LLM-based methods use `deepseek-r1:70b`. Counts are aggregated across random seeds and include LLM calls, prompt tokens, completion tokens, and total tokens. The full Curator reduces calls and total tokens relative to WorldCoder while achieving stronger transfer (Tab. 3).

| Model | Env | LLM calls | Prompt tokens | Completion tokens | Total tokens |
|---|---|---|---|---|---|
| WorldCoder | BabyAI | 814 | 5,586,741 | 910,037 | 6,496,778 |
| Hoarder | BabyAI | 631 | 4,827,384 | 1,054,712 | 5,882,096 |
| Curator | BabyAI | 608 | 4,322,112 | 822,196 | 5,144,308 |
| WorldCoder | Zelda | 603 | 3,586,278 | 540,929 | 4,127,207 |
| Hoarder | Zelda | 368 | 3,032,226 | 604,643 | 3,636,869 |
| Curator | Zelda | 321 | 2,485,482 | 457,386 | 2,942,868 |

the mission string, indicating that the model compensates for missing context by hallucinating task assumptions rather than reasoning from first principles. The only genuinely task-agnostic signal it reliably extracts—detecting wall collisions—appears in 77.5% of responses, but such events are too sparse to yield a meaningful reward gradient. Preference labels over trajectories therefore cannot supply a task-agnostic training signal in this setting, because the behavior being ranked is better or worse only relative to a goal the labeler does not have.

**WorldCoder.** We run WorldCoder [56] from its public release with no algorithmic changes. Our only modifications are (i) swapping the OpenAI backend for a local Ollama server serving `deepseek-r1:70b`, the same backbone used by the Curator and Motif, and (ii) wrapping the per-episode driver in a sequential-curriculum loop so that the step budget, evaluation cadence, and held-out splits are the ones used for PPO and Motif. WorldCoder reads the same symbolic observation the other methods receive rather than pixels, and plans over its own synthesized code using the environment's primitive action set, so no separate grounding step is needed. All remaining hyperparameters keep their released defaults.

The synthesized code, a transition function plus one reward function per mission string, is carried over verbatim between curriculum environments, and only the experience buffer and the `mission_accomplished` set are cleared at the environment boundary. When a new environment introduces transitions inconsistent with the carried-over transition function, the LLM edits that file in place. WorldCoder has no library object that can prune or retire an element, so every prior code path persists unless the LLM happens to delete it during a repair. This is the mechanism behind the library-size comparison in Sec. 5.2.

# G Experiment 3 Supplementary Results

## G.1 Cross-stage transfer

Fig. 10 evaluates each method's carried representation at every training stage on every BabyAI environment. The dashed line in each panel marks the stage at which that environment finishes its own training, where values to the left are forward transfer (success on the environment before it has been trained, using the library accumulated from earlier environments) and values to the right are backward transfer (retention or retroactive improvement after later environments are added). Three patterns are worth noting.

On the hardest environment $\mathrm{env}_3$, the Curator reaches $\sim$.30 success before $\mathrm{env}_3$ training begins (at $+\mathrm{env}_2$), and finishes at $\sim$.45 after its own training stage. The library accumulated from $\mathrm{env}_0$-$\mathrm{env}_2$ already handles a substantial fraction of $\mathrm{env}_3$ zero-shot, because predicates and operators promoted at earlier environments (e.g., `navigate_to`, `pickup`, `toggle`) compose into the longer action chains $\mathrm{env}_3$ requires. WorldCoder, by contrast, starts $\mathrm{env}_3$ near zero, because its mission-specific reward functions do not compose.

On environments $\mathrm{env}_0$-$\mathrm{env}_2$, the Curator preserves earlier-environment success after the dashed line and on $\mathrm{env}_2$ shows clear retroactive improvement, where predicates promoted at `UnlockLocal` (notably `unlocked(?door)`) tighten operator preconditions on `Pickup` and `OpenRedDoor`, so earlier environments become *easier* as the curriculum progresses. WorldCoder and PPO show the textbook

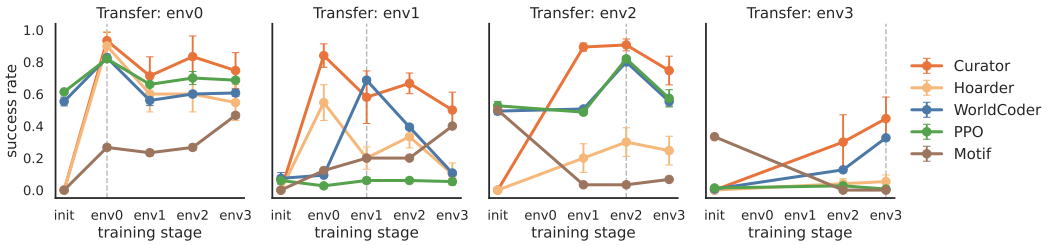


Figure 10: **BabyAI cross-stage transfer.** Each panel evaluates a fixed environment $\mathrm{env}_i$ as the curriculum proceeds. The dashed line marks the stage at which $\mathrm{env}_i$ finishes its own training; values to its left show forward transfer (success on $\mathrm{env}_i$ before it has been trained, achieved by the library accumulated from earlier environments), and values to its right show backward transfer (retention or retroactive improvement on $\mathrm{env}_i$ as later environments are added). The Curator transfers forward (e.g., $\mathrm{env}_3$ panel: $\sim$ .30 at $+\mathrm{env}_2$ before $\mathrm{env}_3$ training begins) and backward (no decay after the dashed line; on $\mathrm{env}_3$, monotonic accumulation as $\mathrm{env}_3$ training itself proceeds). WorldCoder and PPO peak at each environment's matched stage and decay thereafter (the textbook forgetting profile).

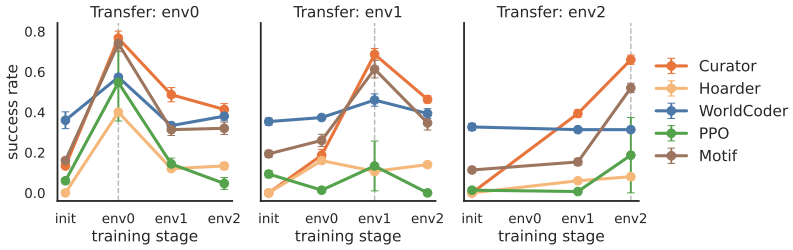


Figure 11: **Zelda cross-stage transfer.** Same protocol as Fig. 10. The Curator transfers forward (visible on $\mathrm{env}_1$, $\mathrm{env}_2$ panels before each environment is trained) and preserves earlier-environment performance afterward, with retroactive improvement on $\mathrm{env}_1$. WorldCoder peaks lower than the Curator and decays; the Hoarder ablation again fails on both axes.

forgetting profile—peak at the matched training stage, then decay (e.g., on $\mathrm{env}_1$, WorldCoder peaks $\sim$.69 at $+\mathrm{env}_1$ and drops to $\sim$.10 by $+\mathrm{env}_3$ as later edits to the carried code break earlier transitions).

The empowerment-ablated Hoarder, which keeps every LLM-proposed element rather than curating, shows neither forward nor backward gains, tracking the Curator's curve at $\mathrm{env}_0$ briefly before flattening out and never recovering on later environments.

Fig. 11 shows the same patterns on Zelda. The Curator transfers forward into $\mathrm{env}_1$ ($\sim$.20 before $\mathrm{env}_1$ training begins, climbing to $\sim$.49 at $+\mathrm{env}_1$ and staying at $\sim$.43 after $+\mathrm{env}_2$) and into $\mathrm{env}_2$, while WorldCoder peaks lower ($\sim$.36 on $\mathrm{env}_1$) and decays. The Hoarder again fails on both axes.

### G.2 Held-out transfer

Figs. 12 and 13 report zero-shot reward_success of the frozen library $\mathcal{L}$ on configurations not seen during training. The frozen-library protocol is the same as forward and backward transfer (Sec. F.3): no training trajectories on the held-out environment, no candidate proposals, only library lookup and Actor execution.

On $\mathrm{env}_4$ (`PutNextLocal`), $\mathrm{env}_6$ (`PickupAbove`), and $\mathrm{env}_7$ (`Unlock` variant), the Curator reaches $\sim$.45-.75 success without ever having trained on these configurations, ahead of WorldCoder by 15-30 absolute points and ahead of all policy baselines by an order of magnitude. The library composes, since predicates promoted at `UnlockLocal` (e.g., `unlocked(?door)`, `is_key_for`) and operators learned at `Pickup`/`OpenRedDoor` re-bind to new objects and layouts in the held-out environments without re-synthesis.

On $\mathrm{env}_5$ (`MoveTwoAcrossS5N2`), every method drops below .15, including the Curator. The environment requires achieving a conjunction of binary inter-object spatial relations (`next_to(o1, o2)` and `next_to(o3, o4)` simultaneously), but every training-environment goal is unary or agent-indexed

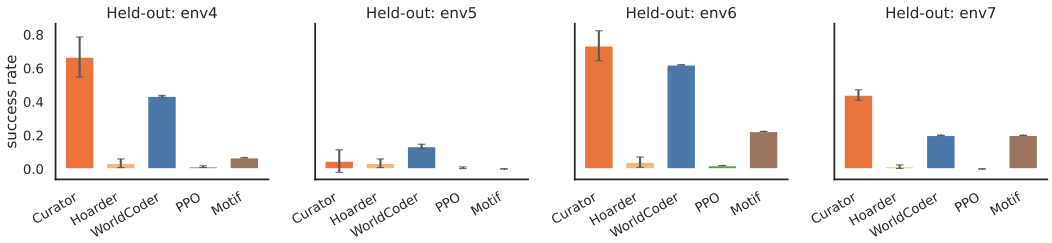


Figure 12: **BabyAI held-out transfer.** Zero-shot reward_success of the frozen library $\mathcal{L}$ on four held-out environments, with error bars giving the standard deviation over 3 seeds.

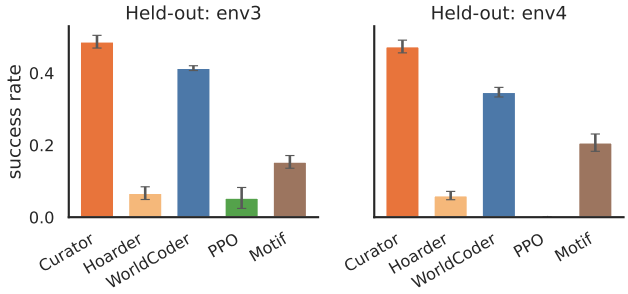


Figure 13: **Zelda held-out transfer.** Same protocol as Fig. 12, on the two held-out Zelda environments.

(`holding`, `unlocked`, `at-location`). The bottleneck is the LLM proposer's goal diversity, since even when it generates a candidate `next_to(?o1, ?o2)` predicate during training, the witness goals $\mathcal{W}_t$ available on training environments never test compositional object-object relations, so the predicate fails the empowerment threshold and is not promoted. The criterion behaves correctly given the probes it sees; the probes simply do not cover the structurally distinct goals $\mathrm{env}_5$ demands. We read this as a limitation of the proposer's goal-generation diversity, not of the empowerment criterion itself.

On both held-out Zelda environments, the Curator reaches $\sim$.48 success and outperforms WorldCoder by 8-12 absolute points; policy baselines (PPO, Motif) remain near floor, and Hoarder collapses to $\sim$.06. The smaller margin reflects that Zelda's training-and-held-out gap is narrower (layout differences with the same primitive dynamics), so WorldCoder's mission-keyed reward functions transfer better than they do on BabyAI's compositionally distinct held-out environments.

Hoarder, which keeps every LLM-proposed element rather than curating, drops to $\sim$.03-.06 on every held-out environment in both families. The Curator's held-out gains therefore follow from the empowerment-based selection that keeps $\mathcal{L}$ compact and broadly applicable, rather than from the proposer generating useful candidates.

### G.3 Library compactness and planning load

The main-text Tab. 3 reports the curriculum-averaged library footprint behind the bloat claim in Sec. 5.2. This subsection provides the per-environment decomposition and the methodological details of the measurement. We measure the carried-forward representation in characters of source code, broken into its two natural components on each side. WorldCoder carries a world model (Python transition function) plus a reward model containing one Python reward function per mission seen so far. The Curator carries the final PDDL domain file the planner compiles against, plus the *grounding code* attached to each promoted predicate.

**Per-environment breakdown.** Tab. 9 reports both decompositions per environment, mean over seeds. Two patterns are consistent across the families.

*WorldCoder.* Its reward model grows monotonically across the four training environments, taking the reward-side character count from 6.2 KB to 20.4 KB (3.3×). The world-model side stays around

Table 9: **Carried-forward representation size across training environments.** Representation size is measured as source-code characters retained after each environment. WorldCoder size includes its generated world-model and reward code; Curator size includes the PDDL domain and predicate-grounding code used by the planner. Lower totals indicate a more compact carried-forward representation.

| | | WorldCoder | | | Curator | | |
|---|---|---|---|---|---|---|---|
| Family | Environment | WM.py | rwd.py | total | PDDL | grnd.py | total |
| BabyAI | env0 (GoToLocal) | 4 982 | 6 207 | 11 189 | 2 811 | 3 883 | 6 694 |
| BabyAI | env1 (Pickup) | 4 449 | 12 305 | 16 754 | 2 811 | 3 883 | 6 694 |
| BabyAI | env2 (OpenRedDoor) | 5 218 | 15 443 | 20 661 | 5 475 | 4 091 | 9 566 |
| BabyAI | env3 (UnlockLocal) | 4 337 | 20 360 | 24 697 | 5 141 | 4 091 | 9 232 |
| Zelda | env0 (`empa_250`) | 2 799 | 1 169 | 3 969 | 2 173 | 2 109 | 4 283 |
| Zelda | env1 (`empa_251`) | 5 906 | 2 686 | 8 592 | 2 275 | 2 226 | 4 501 |
| Zelda | env2 (`empa_252`) | 4 986 | 3 871 | 8 857 | 2 572 | 1 893 | 4 465 |

5 KB throughout, since the transition function is edited in place rather than appended to. Total carried representation grows from 11.2 to 24.7 KB.

*Curator.* Its carried representation stays small and roughly flat. Across the EMPA/Zelda environments the symbolic side is 2-3 KB and the grounding code is 2-4 KB, for a total of 2-6 KB per environment—about 4-10$\times$ smaller than WorldCoder's BabyAI reward bundle and approximately constant in the curriculum step. The trusted-element counts behind those byte numbers are 5-6 predicates and 0-3 operator schemas per environment.