Beyond Fixed Tasks: Unsupervised Environment Design for Task-Level Pairs

Daniel Furelos-Blanco[1], **Charles Pert**[1], **Frederik Kelbel**[1], **Alex F. Spies**[1], **Alessandra Russo**[1], **Michael Dennis**[2*]

[1]Imperial College London
[2]Google DeepMind
{d.furelos-blanco18, c.pert22, frederik.kelbel20, afspies, a.russo}@imperial.ac.uk, dennismi@google.com

Abstract

Training general agents to follow complex instructions (*tasks*) in intricate environments (*levels*) remains a core challenge in reinforcement learning. Random sampling of task-level pairs often produces unsolvable combinations, highlighting the need to co-design tasks and levels. While unsupervised environment design (UED) has proven effective at automatically designing level curricula, prior work has only considered a fixed task. We present ATLAS (Aligning Tasks and Levels for Autocurricula of Specifications), a novel method that generates *joint autocurricula* over tasks and levels. Our approach builds upon UED to automatically produce solvable yet challenging task-level pairs for policy training. To evaluate ATLAS and drive progress in the field, we introduce an *evaluation suite* that models tasks as reward machines in Minigrid levels. Experiments demonstrate that ATLAS vastly outperforms random sampling approaches, particularly when sampling solvable pairs is unlikely. We further show that mutations leveraging the structure of both tasks and levels accelerate convergence to performant policies.

Code — https://github.com/spike-imperial/atlas
Extended version — https://arxiv.org/abs/2511.12706

1 Introduction

Training generally-capable agents that follow diverse instructions (*tasks*) in varied environments (*levels*) is a central challenge in reinforcement learning (OEL Team et al. 2021). This dual complexity emerges across domains—from cooking agents executing recipes in different kitchens, to navigation agents following directives through unfamiliar cities. To generate this open-ended complexity, methods like EUREKA (Ma et al. 2024), OMNI-EPIC (Faldor et al. 2025), and others (Zhang et al. 2024; Klissarov et al. 2025) have often relied on LLM-driven code-generation.

However, there has been growing interest in an alternative approach of expressing tasks through *formal languages*, including temporal logics (Kuo, Katz, and Barbu 2020; Vaezipoor et al. 2021; Qiu, Mao, and Zhu 2023; Jackermeier and Abate 2025) and finite-state machines (Yalcinkaya et al. 2023, 2024). Unlike natural language (Luketina et al. 2019), formal languages offer unambiguous semantics and precise progress tracking, making them well-suited for generalization across task-level pairs. These approaches typically train agents by sampling from uninformed task-level distributions, a strategy known as *domain randomization* (DR; Sadeghi and Levine 2017; Tobin et al. 2017).

DR can succeed when sampled task-level pairs are mostly solvable. However, as the number of task-level combinations grows, the proportion of solvable pairs tends to decrease. Even among solvable pairs, DR generates samples of *arbitrary difficulty*, causing unsolvable or overly challenging levels to dominate training. This raises the question: *How can we train agents effectively from solvable yet appropriately challenging task-level pairs?*

Unsupervised environment design (UED; Dennis et al. 2020) has demonstrated success in automatically generating *level* curricula, producing solvable and progressively more challenging levels that enable effective generalization. However, existing UED methods only generate curricula over levels for a *fixed task*, yet general agents require both.

We introduce ATLAS (Aligning Tasks and Levels for Autocurricula of Specifications), which extends UED's principled curriculum generation to both tasks and levels, ensuring that agents train on solvable yet appropriately challenging pairs. We express tasks as *reward machines* (RMs; Toro Icarte et al. 2018)—finite-state machines that compactly represent reward functions. Our *contributions* include:

Joint task-level autocurricula ATLAS co-designs tasks and levels to generate aligned curricula, ensuring solvable yet challenging pairs. A policy network, conditioned on graph embeddings of the RMs, learns effectively from the resulting curriculum.

Structure-aware task mutations We leverage RM structure to guide task mutations while co-evolving levels, achieving faster learning than approaches that solely curate solvable but challenging random samples.

Evaluation We introduce an evaluation suite combining RM tasks with Minigrid levels (Chevalier-Boisvert et al. 2023) that, unlike previous work, explicitly targets settings where task-level pairs are rarely solvable.

Our *experiments* show that ATLAS generates joint task-level curricula that enable agents to master increasingly complex levels (with more rooms and objects) and tasks

*Contributed in an advisory capacity.

(with more RM states). By prioritizing solvable pairs at the frontier of agent capability, ATLAS consistently outperforms DR across diverse test instances. Interestingly, forming curricula by repeatedly mutating tasks and levels from a simple starting point can result in policies with comparable performance to those based on random sampling alone.

2 Background

We review unsupervised environment design and the use of reward machines in reinforcement learning.

2.1 Unsupervised Environment Design

Unsupervised environment design (UED; Dennis et al. 2020) aims to generate training environments that adapt to an agent's capabilities, inducing an *autocurriculum* (Leibo et al. 2019; Narvekar et al. 2020; Portelas et al. 2020). The key is to *parameterize* RL environments (e.g., grid size) and adapt these parameters during training. The goal is to train agents that generalize across all possible parameterizations.

Formally, UED problems are *underspecified partially observable Markov decision processes* (UPOMDPs; Dennis et al. 2020). A standard POMDP is a tuple $\mathcal{M} = \langle A, O, S, \mathcal{T}, \mathcal{I}, \mathcal{R}, \gamma \rangle$ where A is a set of actions, O is a set of observations, S is a set of states, $\mathcal{T} : S \times A \to \Delta(S)$ is the transition function, $\mathcal{I} : S \to O$ is the observation function, $\mathcal{R} : S \to \mathbb{R}$ is the reward function, and γ is the discount factor. UPOMDPs augment POMDPs with a *parameter space* Θ, which represents the space of all environment configurations. Each parameterization $\theta \in \Theta$ instantiates a fully specified POMDP $\mathcal{M}^\theta$, often called *level* (Cobbe et al. 2020; Jiang, Grefenstette, and Rocktäschel 2021). The expected discounted return (or *value*) of a *policy* π in level $\mathcal{M}^\theta$ is $V^\theta(\pi) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t]$, where T is a horizon. The goal is to produce a sequence of distributions over Θ that maximizes the policy's value across levels.

2.2 Methods for UED

UED methods often frame curriculum design as a game between two players: a *teacher*, which proposes levels, and a *student*, which trains on them. The teacher generates levels by maximizing a *utility score*. We review two scoring strategies: *constant* and *regret-based* scoring.

Constant. All levels have equal utility, ignoring the student's performance. Parameters θ are uniformly sampled from the parameter space Θ. This strategy is commonly known as *domain randomization* (DR; Sadeghi and Levine 2017; Tobin et al. 2017).

Regret-Based. The regret for a level $\mathcal{M}^\theta$ is the value gap $V^\theta(\pi^*) - V^\theta(\pi)$ between the optimal policy π^* and the current policy π. Unlike DR, regret scoring deprioritizes unsolvable levels and focuses on those at the frontier of the agent's capabilities. Theoretically, a teacher that maximizes regret induces a minimax regret student policy at equilibrium (Dennis et al. 2020).

Since finding the optimal policy is intractable, UED methods resort to regret approximations. For example, MaxMC (Jiang et al. 2021) uses the highest undiscounted return seen so far, $R^\theta_{\max}$, as a proxy for optimal performance in $\mathcal{M}^\theta$ and computes regret as $(1/T)\sum_{t=0}^{T} R^\theta_{\max} - V^\theta(o_t)$, where $V^\theta(o_t)$ is the value of the observation at time t.

Prioritized Level Replay (PLR; Jiang, Grefenstette, and Rocktäschel 2021) curates a *buffer* of high-regret levels. At each episode, PLR chooses between (i) with probability p, replaying levels from the buffer and updating their estimates, and (ii) with probability $1 - p$, sampling new levels via DR and adding them to the buffer. In both cases, the student policy trains on the selected levels.

In this paper, we consider two extensions of PLR. *Robust PLR* (PLR$^\perp$; Jiang et al. 2021) trains the student only on replayed levels. *ACCEL* (Parker-Holder et al. 2022) extends PLR$^\perp$ by *mutating* the last replayed levels (e.g., moving an object) before adding them to the buffer, thereby exploring the neighborhood of high-regret levels rather than solely relying on random sampling.

2.3 Reward Machines

Reward machines (RMs; Toro Icarte et al. 2018) are finite-state machines for specifying reward functions using high-level propositional events. They provide a flexible task representation, supporting derivations from other formal languages (Camacho et al. 2019) and mappings from natural language (Tuli et al. 2022; Liu et al. 2023; Li et al. 2025).

Formally, an RM is a tuple $\langle U, P, \delta, \mathcal{R}, u_0, u_A \rangle$, where U is a finite set of states; P is a finite set of propositions that forms the *alphabet* of the RM; $\delta : U \times 2^P \to U$ is the state-transition function, which maps an RM state and a subset of propositions into an RM state; $\mathcal{R} : U \times U \to \mathbb{R}$ is the reward-transition function, which maps an RM state pair into a reward; $u_0 \in U$ is the initial state of the RM; and $u_A \in U$ is the accepting state of the RM, which represents the successful completion of the task.

We refer to proposition sets $L \in 2^P$ as *labels*. These are produced by a *labeling function* $\mathcal{L} : O \to 2^P$ from environment observations. Given an RM state u and a label L, the RM transitions to $u' = \delta(u, L)$ and emits reward $\mathcal{R}(u, u')$.

3 Task-Level Generalization Framework

We introduce a framework for problem-conditioned RL, where a *problem specification* (hereafter, *problem*) is a tuple consisting of a *task*, an instruction the agent must follow, and a *level*, the environment instance in which the agent acts.

We explore two key *questions* within this framework. First, how agents generalize from *rarely solvable* task-level pairs—a critical but understudied regime. Such pairs arise when tasks are infeasible in a level (e.g., opening a red door in a level with no doors). Second, how to extend UED beyond fixed tasks to settings where tasks and levels are *co-designed*.

We instantiate this framework using RMs for tasks and Minigrid environments (Chevalier-Boisvert et al. 2023) for levels, with samplers that generate problems in both rarely solvable and frequently solvable regimes. We also contribute 150 hand-designed problems that test capabilities such as long-term planning, implicit subgoals, and exploration. See Appendix A for details.

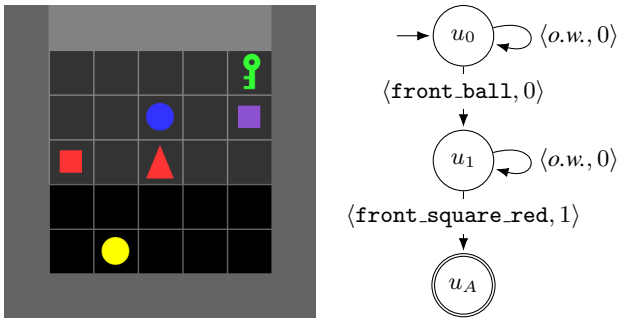


Figure 1: A *problem* consisting of a Minigrid *level* and an RM *task* for *"go to a ball, then go to a red square"*.

3.1 Levels

We instantiate *levels* as Minigrid environments (Chevalier-Boisvert et al. 2023). A level consists of a grid containing *objects* (keys, squares, balls, doors) with different *colors* (red, green, blue, purple, yellow, gray). Doors have three *states*: locked, open, or closed, with locked doors requiring keys of matching color to open. The agent observes a 5×5 region in front of it.

Example 1. Figure 1 (left) shows a Minigrid level with the agent (red triangle) and its observation (highlighted area).

Sampling. We develop samplers that generate levels with a random number of rooms and objects. All objects are randomly determined. Agent and non-door objects are randomly placed. Rooms are connected via doors.

3.2 Tasks

Tasks are instantiated as RMs that encode BabyAI instructions (Chevalier-Boisvert et al. 2019). These instructions consist of *go to*, *open*, *pick up*, and *put next* commands, each applied to specific objects, optionally conditioned on color and state. The RM *alphabet* is defined by mapping these commands to propositions:

- $\texttt{front\_}\langle \texttt{o}_1\rangle\texttt{\_}\langle \texttt{c}_1\rangle\texttt{\_}\langle \texttt{s}_1\rangle$ indicates the agent is in front of object $\texttt{o}_1$ with color $\texttt{c}_1$ and state $\texttt{s}_1$,
- $\texttt{carrying\_}\langle \texttt{o}_1\rangle\texttt{\_}\langle \texttt{c}_1\rangle\texttt{\_}\langle \texttt{s}_1\rangle$ indicates the agent is carrying object $\texttt{o}_1$ with color $\texttt{c}_1$ and state $\texttt{s}_1$, and
- $\texttt{next\_}\langle \texttt{o}_1\rangle\texttt{\_}\langle \texttt{c}_1\rangle\texttt{\_}\langle \texttt{s}_1\rangle\texttt{\_}\langle \texttt{o}_2\rangle\texttt{\_}\langle \texttt{c}_2\rangle\texttt{\_}\langle \texttt{s}_2\rangle$ indicates that object $\texttt{o}_1$ with color $\texttt{c}_1$ and state $\texttt{s}_1$ is next to object $\texttt{o}_2$ with color $\texttt{c}_2$ and state $\texttt{s}_2$ within the agent's visual field,

where $\texttt{o}_i \in \{\texttt{ball}, \texttt{square}, \texttt{key}, \texttt{door}\}$, $\texttt{c}_i \in \{\texttt{red}, \texttt{green}, \texttt{blue}, \texttt{purple}, \texttt{yellow}, \texttt{gray}, \epsilon\}$, and $\texttt{s}_i \in \{\texttt{open}, \texttt{closed}, \texttt{locked}, \epsilon\}$ for $i \in \{1, 2\}$. The state $\texttt{s}_i$ is unspecified (ϵ) if $\texttt{o}_i \neq \texttt{door}$. The propositions capture each instruction type: `front` for *go to* and *open*, `carrying` for *pick up*, and `next` for *put next*. Crucially, RMs enable sequencing and alternating formulas over these propositions, mirroring the connectors *then* and *and* in BabyAI instructions.

Example 2. Figure 1 (right) shows an RM for the instruction *"go to a ball, then go to a red square"*. A reward of 1 is given upon completing the task, and 0 otherwise. Given the level on its left, the observation of the agent is mapped into the label below, which satisfies the transition from u_0 to u_1:

$$\left\{\begin{matrix}\texttt{front\_ball} & \texttt{next\_square\_purple\_key\_green}\\ \texttt{front\_ball\_blue} & \texttt{next\_square\_key\_green}\\ \texttt{next\_square\_key} & \texttt{next\_square\_purple\_key}\end{matrix}\right\}.$$

Sampling. We introduce two RM sampling strategies:

- *Sequential.* Generates RMs with a single path from u_0 to u_A, where the path length is sampled from a predefined range. Figure 1 is a length 2 example.
- *Random Walk-Based.* Generates RMs with one or more (possibly cyclic) paths from u_0 to u_A, subsuming the sequential case. Transitions are determined by a random walk over a uniformly initialized Markov transition matrix. See Appendix A.2 for details.

In both strategies, each transition is labeled with a proposition p sampled uniformly from the alphabet, while its negation $\neg p$ is added to all other outgoing transitions from the same state to enforce *determinism*.

Sampled RMs support a variety of *reward functions*, including sparse rewards (1 only when reaching u_A), stepwise rewards (1 on each transition to a new state, e.g. Tuli et al. 2022), and shaped rewards based on the distance to u_A (e.g., Camacho et al. 2017; Furelos-Blanco et al. 2021).

3.3 Problem Sampling

There are two main strategies to sample a problem:

Independent. Tasks and levels are sampled separately.

Level-Conditioned. Task generation is constrained by the objects in the sampled level. Only propositions involving those objects may label the edges in the RM.

Level-conditioning increases the likelihood of sampling *solvable* problems, but does not guarantee it. For example, a task may require unlocking a door with an existing but unreachable key. Guaranteeing solvability is challenging, as it may require solving the problem during generation.

4 Problem-Conditioning via Autocurricula

We introduce ATLAS (Aligning Tasks and Levels for Autocurricula of Specifications), a method for learning policies that *generalize* across problem specifications by leveraging *autocurricula*—automatically adapting the training problems to match the agent's capabilities. This is critical in settings where *solvable* problems are rarely sampled.

Our approach extends UED beyond level generation to *jointly* adapt over both tasks and levels. By co-designing these, regret-based UED methods generate problems that are not only solvable (i.e., the task is feasible within the corresponding level) but also challenging.

Figure 2 illustrates ATLAS, which comprises two components: a problem-conditioned policy network and a UED-driven curriculum generation loop. We describe the components of ATLAS in the following sections.

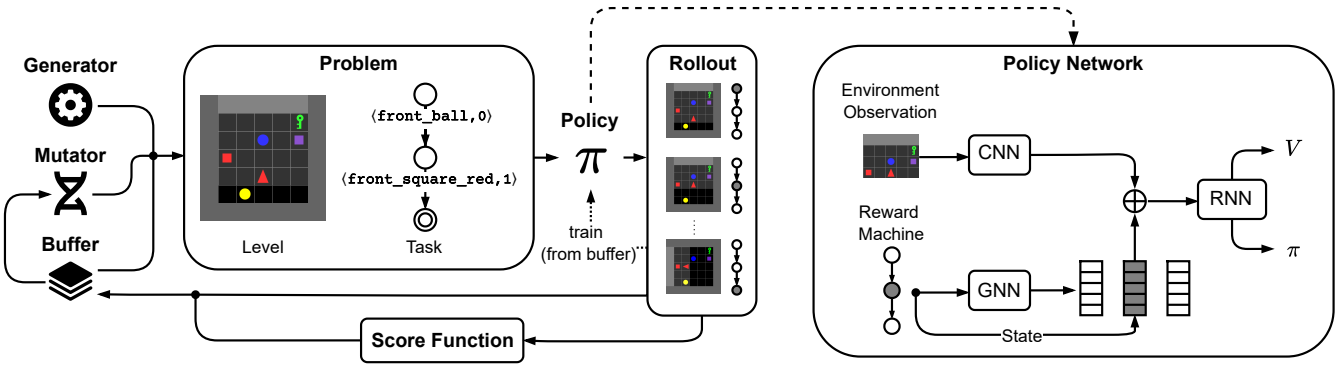


Figure 2: Overview of ATLAS instantiated with PLR$^{\perp}$ and ACCEL. The UED loop **(left)** samples problems—i.e., task-level pairs—from either a generator or a buffer of high-regret problems. ACCEL provides problems that result from mutating selected buffer problems. The policy network **(right)** processes observations via a convolutional neural network (CNN) and RM tasks via a graph neural network (GNN). The GNN produces representations for all RM states. The current state's embedding is concatenated with the CNN features and passed through a recurrent neural network (RNN) to capture history. The resulting representation is used to generate actions and value estimates. Policy rollouts are used to train the network (for buffer-sourced problems), and to compute regret scores, which determine if new problems enter the buffer or update existing ones. Unlike PLR$^{\perp}$ and ACCEL, DR trains policies only from problems produced by the generator.

4.1 Policy Architecture

To generalize across task-level combinations, we use an actor-critic architecture conditioned on both environment observations and RM task states. The *key* design choice is to encode RMs—labeled directed graphs—using a *graph neural network* (GNN). Unlike fixed embeddings (e.g., one-hot encodings of the RM state index), GNNs naturally handle varying RM topologies and edge labelings, supporting generalization across RMs. The policy is trained using PPO (Schulman et al. 2017). See Appendix C for details.

4.2 Problem Autocurricula via UED

We extend UED approaches to support joint generalization over both tasks and levels. For brevity, we refer to ATLAS instantiations of these methods as DR, PLR$^{\perp}$, and ACCEL, corresponding to the underlying UED approach. In the latter two, generalization is driven by automatically induced curricula over both levels and RM tasks.

Unlike the original ACCEL (Parker-Holder et al. 2022), which only applied level mutations, ATLAS' instantiation incorporates *task-aware mutations* leveraging the RM structure. This enables exploring a broader neighborhood of problems by modifying both tasks and levels.

A mutation consists of a sequence of edits, where the number of edits (sampled from a predefined range) and each edit type are selected uniformly at random. We define three *types of edits* below (see Appendix D for examples):

Level Edits. Edits applied to the environment. In Minigrid, this includes moving the agent, adding/removing rooms, and adding/removing/replacing/moving an object.

Task Edits. Edits applied to the RM structure. These include switching a proposition and adding/removing a state.

Hindsight Edits. Inspired by *hindsight relabeling* (Andrychowicz et al. 2017), these edits generate new subproblems from partial progress. When a rollout ends in an intermediate RM state u (neither initial nor accepting), there are two possible edits:

- *Preceding edits.* Keep the original level but set u as the accepting RM state, yielding a simpler problem.
- *Succeeding edits.* Use the reached environment state as the new level and set u as the initial RM state, creating a continuation problem.

Since these edits depend on the original problem, they can only be applied as the first step in a mutation sequence.

5 Experiments

We address the following research questions:

RQ1 Joint Curriculum Effectiveness. Do approaches that generate curricula over both levels and tasks (PLR$^{\perp}$, ACCEL) outperform DR?

RQ2 Mutation Benefits. Does incorporating mutations (ACCEL) offer advantages over curation-only approaches (PLR$^{\perp}$)?

RQ3 Joint Curriculum Emergence. Does ATLAS induce autocurricula over both levels and tasks?

RQ4 Mutation Analysis. Which mutation types become most prevalent during training?

We address these questions in our main results, supported by key ablations below and extended analysis in Appendix E.

5.1 Experimental Setup

We describe the *default* training setup used in our experiments. Additional details are provided in Appendix E.2.

Problem Generation. Levels and tasks are sampled *independently*, creating a challenging setting in which most sampled problems are not solvable. *Levels* are generated by sampling (i) a number of rooms in $\{1, 2, 4, 6\}$, (ii) a number of objects from a grid-dependent range, (iii) the objects themselves, and (iv) the agent position. *Tasks* are specified as RMs generated via the sequential sampler, with path lengths randomly chosen between 1 and 5. Although our approach supports various reward functions (see Section 3.2), we use sparse rewards—1 on transitions to u_A and 0 otherwise—as they are general yet challenging, making them well-suited for assessing performance and curricula emergence.

Algorithms. We evaluate ATLAS instantiated with DR, PLR$^{\perp}$, and ACCEL. For ACCEL, we distinguish between (i) ACCEL, where problems are sampled using the setup above; and (ii) ACCEL-0, where sampled problems consist of single-room levels with one object and RMs with one transition, and complexity emerges solely through mutations. We apply the *mutations* described in Section 4.2, with sequence lengths sampled uniformly in the range 7–10.

Metrics. We evaluate performance using two complementary metrics, averaged over five seeds with 95% confidence intervals. Each problem is evaluated 10 times per seed. First, we compute the *conditional value at risk* (CVaR; Rutherford et al. 2024), which quantifies *robustness* by reporting the solve rate for the $\alpha\%$ worst-performing problems in a large sampled set. Second, we report *test performance* using the aggregate *inter-quartile mean* (IQM; Agarwal et al. 2021) of solve rates on our hand-designed evaluation set, assessing both in-distribution and out-of-distribution generalization.

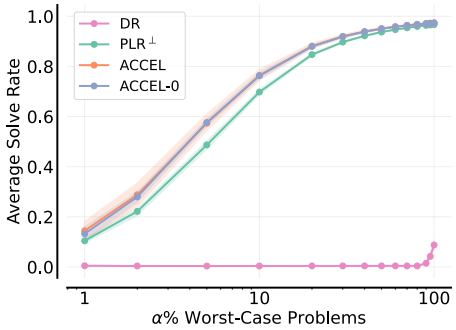
5.2 Main Results

We present our core experimental results addressing the research questions outlined earlier.

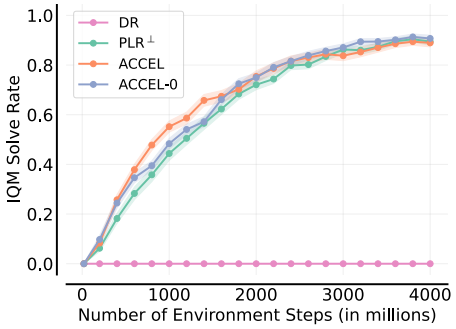
Performance (RQ1, RQ2). Figure 3a displays CVaR results measuring agent *robustness*. ACCEL variants consistently outperform PLR$^{\perp}$ across most α values, particularly for worst-case scenarios (low α). At $\alpha = 100\%$ (average performance), both methods perform similarly and substantially outperform DR, which solves almost no problems.

Figure 3b shows zero-shot performance over time on our hand-designed evaluation set. While ACCEL variants and PLR$^{\perp}$ achieve comparable final results, ACCEL converges faster in early training. Notably, ACCEL-0 achieves strong performance, demonstrating that problem complexity can emerge purely through targeted mutations, without relying on initial problem diversity. DR again underperforms due to the scarcity of solvable training problems.

Figure 4 breaks down performance on three challenging hand-designed scenarios. MYOPIC requires long-term planning: the agent must face a square *but* it cannot be the one behind the locked blue door, as unlocking that door renders the problem unsolvable. PATROL involves navigating with underspecified instructions (i.e., carrying the keys is required to unlock the doors, but not mentioned). CHOICE combines both: agents must explore the grid while ensuring some doors remain locked. In all cases, regret-based methods outperform DR by a wide margin.



(a) CVaR of the solve rate.



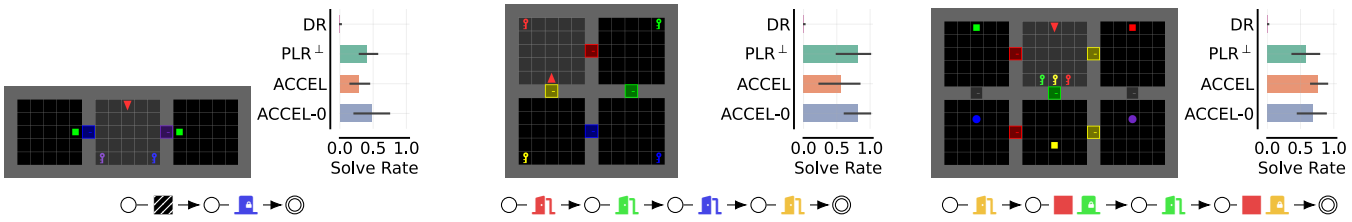
(b) Aggregate zero-shot performance on the hand-designed test set.

Figure 3: Performance of ATLAS variants on (a) the worst-case problems and (b) a challenging hand-designed test set.

The success of PLR$^{\perp}$ and ACCEL stems from filtering solvable problems. Independent sampling yields only 2.7% solvable problems per batch (see Appendix E.3), making task-level co-design essential. Both methods curate a buffer of high-regret problems, resulting in the generation of challenging yet solvable problems. Indeed, the fraction of solvable buffer problems steadily grows, eventually nearing 100% (see Appendix E.4)—interestingly, the growth is faster in ACCEL than for PLR$^{\perp}$. DR, in contrast, trains on predominantly unsolvable problems throughout.

Curriculum Analysis (RQ3). We analyze how curricula emerge across both tasks and levels. Figure 5 illustrates the evolution of buffer problems during training. Initially, PLR$^{\perp}$ and ACCEL curate buffers with simple problems—few rooms, objects, and RM states. Over time, problem complexity increases along all dimensions. ACCEL variants reach higher RM state counts than PLR$^{\perp}$, reflecting a stronger emphasis on task complexity. As expected, ACCEL-0 starts with the simplest problems (two states, one room, one object) as it only relies on mutations from these.

Figure 6 provides examples of generated training problems near the end of training. For PLR$^{\perp}$ and ACCEL, these are sampled from their respective buffers, while DR examples are drawn directly from the generator. Both PLR$^{\perp}$ and ACCEL prioritize dense six-room layouts, but ACCEL favors RMs with more states. The problems generated by



(a) MYOPIC (b) PATROL (c) CHOICE

Figure 4: Zero-shot performance of ATLAS on hand-designed problems. Symbols represent balls (●), squares (■), keys (🔑), and closed/locked/open/unspecified doors (■/■/■/■). Unspecified doors can match any state. Single symbols indicate `front` propositions, pairs indicate `next` propositions. Striped patterns represent an unspecified color (e.g., ▨ stands for `front_square`).

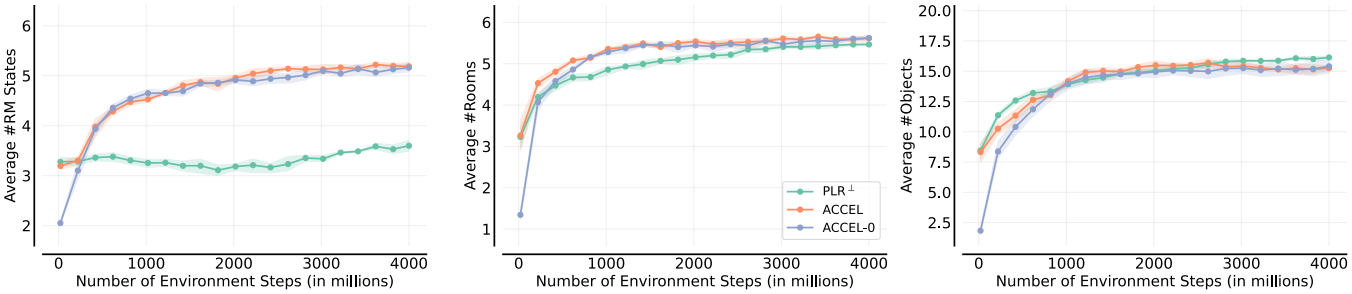


Figure 5: Emergent complexity metrics for problems in the buffer.

PLR$^{\perp}$ and ACCEL are complex, requiring efficient exploration, implicit door unlocking, and even maintaining doors locked to preserve solvability (e.g., the gray doors in ACCEL). In contrast, DR trains on mostly unsolvable or overly complex problems, which the agent struggles to learn from.

Mutation Analysis (RQ4). We observe that the *average number of edits per problem* increases over training. This indicates that (i) longer edit sequences are beneficial and (ii) the buffer composition shifts from primarily randomly generated problems to predominantly mutated ones. This trend shows that ACCEL successfully compounds complexity over time, rather than relying on random sampling alone.

We analyze how different *edit types* shape the curriculum by measuring their frequency in the buffer, weighted by the sampling probabilities of the problems they generate. Task and level edits contribute roughly equally. In contrast, hindsight edits are rare, likely due to limited applicability: they require rollouts to end in intermediate RM states (neither u_0 nor u_A) and can only appear as the first edit. Exploring more general hindsight edits is an interesting direction for future work. See Appendix E.4 for illustrations.

5.3 Problem Sampling Ablations

In the default setup, levels and RM tasks are sampled *independently*, yielding only 2.7% solvable problems per batch. In this setting, PLR$^{\perp}$ and ACCEL vastly outperformed DR. To test the hypothesis that this gap stems from the scarcity of solvable problems, we perform an ablation with *level-conditioned* problem sampling (see Section 3.3), which increases solvability to 83.4%.

We make three key observations. First, as hypothesized, DR improves substantially, matching the CVaR and test performance (≈91.6%) of PLR$^{\perp}$ and ACCEL variants. Second, PLR$^{\perp}$ and ACCEL see only marginal gains (≈1–3%) over the independent setting, highlighting their robustness when solvable problems are rare. Third, a curriculum still emerges: for PLR$^{\perp}$, it becomes more pronounced, with RM tasks containing 4–5 states increasingly prioritized. See Appendix E.5 for further analysis.

5.4 Task Sampling Ablations

To evaluate generalization under more complex training distributions, we replace the sequential task sampler with the *random walk-based* task sampler, restricted to generate RMs as *directed acyclic graphs* (see Section 3). This class subsumes sequential tasks and introduces multiple acyclic paths from u_0 to u_A, creating richer temporal dependencies. As in the default setup, the maximum number of states is 6. We restrict comparisons to DR and PLR$^{\perp}$ under independent problem sampling.

We find that PLR$^{\perp}$ continues to substantially outperform DR for both CVaR and zero-shot performance on the hand-designed set. However, solve rates on the hand-designed evaluation set drop by ≈35% compared to the sequential setting, indicating that increased task complexity can affect overall performance. See Appendix E.6 for full results.

5.5 Mutation Ablations

We assess ACCEL's performance under two types of ablations: (i) *disabling specific edit types*, and (ii) *varying the edit sequence length*. Disabling either level or task edits significantly reduces performance, while combining both types

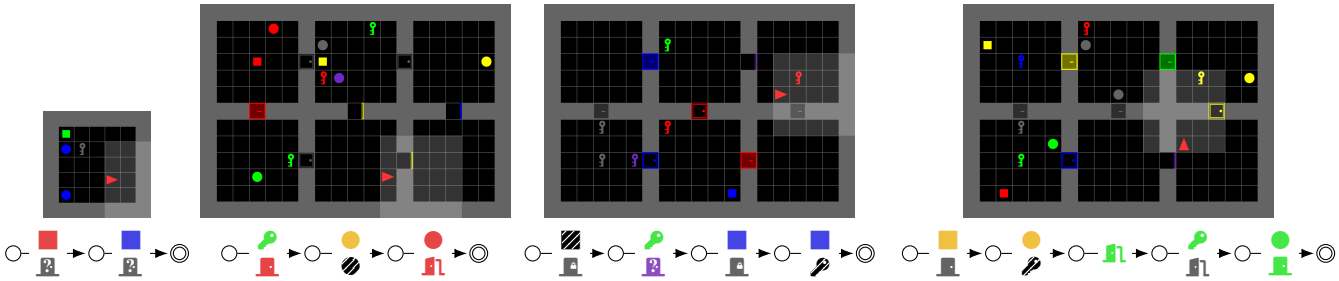


Figure 6: Generated problems by ATLAS (DR, PLR$^{\perp}$, ACCEL, ACCEL-0). See Figure 4 for symbol details.

yields the best results. This highlights the necessity of joint task-level mutations for effective training.

ACCEL is sensitive to sequence length: short sequences severely hinder performance, with single-edit sequences causing a ≈40% drop and 3-edit sequences leading to a ≈15% drop. In contrast, long sequences (20 edits) have minimal effect (≈2% variation). See Appendix E.7 for details.

6 Related Work

We summarize key related work here, with an extended discussion in Appendix B.

Unsupervised Environment Design (UED). To the best of our knowledge, prior UED research has focused on level generation for a fixed task. ATLAS is a UED-based approach that explores the joint co-design of tasks and levels.

Seminal UED approaches rely on regret-based utility scores (Dennis et al. 2020; Jiang et al. 2021), which we also employ in ATLAS. However, recent work identifies three key limitations: high-regret levels need not be diverse (Li, Varakantham, and Li 2023); regret can become irreducible, causing training to stagnate (Beukman et al. 2024); and common regret approximations correlate with success rate rather than true regret (Rutherford et al. 2024). To address these issues, alternative scores—diversity (Li, Varakantham, and Li 2023), novelty (Teoh, Li, and Varakantham 2024), and learnability (Rutherford et al. 2024)—have been proposed. We hypothesize irreducible floors may emerge under any scoring function. Our method supports future research using task-level variations to evaluate these limitations.

Alternative approaches to automatic level design beyond UED have been proposed. POET (Wang et al. 2019) trains populations of specialist agents using evolutionary strategies, while ATLAS trains general agents from random and co-evolved task-level pairs. PCGRL (Khalifa et al. 2020) frames level design as an RL problem, where levels are incrementally edited to optimize a given quality objective.

Formal Language Conditioning. Most closely related to our work, Yalcinkaya et al. (2023, 2024) condition policies on GNN embeddings of *automata* representing reach-avoid task sequences. Yalcinkaya et al. (2023) derive training automata from observed proposition trajectories (akin to our hindsight edits), while Yalcinkaya et al. (2024) apply task mutations. Unlike these approaches, ATLAS targets high-regret tasks for curriculum generation and co-evolves *both* tasks and levels rather than tasks alone.

Linear temporal logic (LTL; Pnueli 1977) approaches typically train using formulas sampled from context-free grammars. Kuo, Katz, and Barbu (2020) and Vaezipoor et al. (2021) encode task structure by embedding the formula's syntax tree through compositional RNNs and GNNs, respectively. Qiu, Mao, and Zhu (2023) and Jackermeier and Abate (2025), unlike previous work and ATLAS, tackle *infinite-horizon* tasks by mapping formulas into equivalent automata whose paths determine the conditioning—the former is sequentially conditioned on each subtask along a path, while the latter is conditioned on derived reach-avoid sequences.

These approaches implement DR, sometimes with fixed curricula (Jackermeier and Abate 2025), which may collapse when solvable problems are rarely sampled. In contrast, ATLAS leverages regret-based UED to generate *autocurricula* of solvable yet challenging problems. We hypothesize prior work with DR succeeds because their domains ensure high solvability: all propositions are observable across levels, making most tasks completable. We address this evaluation gap through settings with naturally low solvability rates, revealing some limitations of DR.

7 Conclusions

We introduce ATLAS, a novel method for generating joint autocurricula over problems (task-level pairs). By co-designing tasks and levels via regret-based UED, ATLAS achieves robust generalization even when most sampled problems are unsolvable—a setting in which domain randomization fails. Additionally, ACCEL-0 achieves strong performance and builds rich curricula by repeatedly mutating initially simple problems. We contribute 150 hand-designed test problems on which we verify that these findings hold, validating our approach and providing a foundation for further research on task-level generalization.

Future Work. Our framework opens several promising research directions. First, extending the approach to other temporal formalisms—including LTL, programs, or hierarchies of RMs (Furelos-Blanco et al. 2023)—would demonstrate the broad applicability of joint curriculum generation via UED. Second, developing scoring functions and mutation operators that better exploit task structure could yield even more effective curricula. Finally, scaling to domains like Craftax (Matthews et al. 2024) would test the effectiveness of joint curricula in richer settings.

Acknowledgements

We thank the reviewers and Roko Parać for their feedback, Isabella Pearce for her help in designing test problems, and Dugan Witherick for his support in setting up the resources provided by the Imperial College Research Computing Service (http://doi.org/10.14469/hpc/2232).

This work is partially supported by DEVCOM Army Research Lab under grant W911NF2220243 and EPSRC projects EP/X040518/1 and EP/Y037421/1. CP is supported by UK EPSRC grant 2760033. FK is supported by UK EPSRC grant 2757464.

References

Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A. C.; and Bellemare, M. G. 2021. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Andrychowicz, M.; Crow, D.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2017. Hindsight Experience Replay. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Beukman, M.; Coward, S.; Matthews, M. T.; Fellows, M.; Jiang, M.; Dennis, M. D.; and Foerster, J. N. 2024. Refining Minimax Regret for Unsupervised Environment Design. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2017. Non-Markovian Rewards Expressed in LTL: Guiding Search Via Reward Shaping. In *Proceedings of the International Symposium on Combinatorial Search (SOCS)*.

Camacho, A.; Toro Icarte, R.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T. H.; and Bengio, Y. 2019. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chevalier-Boisvert, M.; Dai, B.; Towers, M.; Perez-Vicente, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P. S.; and Terry, J. K. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Cobbe, K.; Hesse, C.; Hilton, J.; and Schulman, J. 2020. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A. M.; Russell, S.; Critch, A.; and Levine, S. 2020. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Faldor, M.; Zhang, J.; Cully, A.; and Clune, J. 2025. OMNI-EPIC: Open-endedness via Models of human Notions of Interestingness with Environments Programmed in Code. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Furelos-Blanco, D.; Law, M.; Jonsson, A.; Broda, K.; and Russo, A. 2021. Induction and Exploitation of Subgoal Automata for Reinforcement Learning. *Journal of Artificial Intelligence Research*, 70: 1031–1116.

Furelos-Blanco, D.; Law, M.; Jonsson, A.; Broda, K.; and Russo, A. 2023. Hierarchies of Reward Machines. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Jackermeier, M.; and Abate, A. 2025. DeepLTL: Learning to Efficiently Satisfy Complex LTL Specifications for Multi-Task RL. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jiang, M.; Dennis, M.; Parker-Holder, J.; Foerster, J. N.; Grefenstette, E.; and Rocktäschel, T. 2021. Replay-Guided Adversarial Environment Design. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Jiang, M.; Grefenstette, E.; and Rocktäschel, T. 2021. Prioritized Level Replay. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Khalifa, A.; Bontrager, P.; Earle, S.; and Togelius, J. 2020. PCGRL: Procedural Content Generation via Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Klissarov, M.; Henaff, M.; Raileanu, R.; Sodhani, S.; Vincent, P.; Zhang, A.; Bacon, P.-L.; Precup, D.; Machado, M. C.; and D'Oro, P. 2025. MaestroMotif: Skill Design from Artificial Intelligence Feedback. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kuo, Y.; Katz, B.; and Barbu, A. 2020. Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of LTL formulas. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Leibo, J. Z.; Hughes, E.; Lanctot, M.; and Graepel, T. 2019. Autocurricula and the Emergence of Innovation from Social Interaction: A Manifesto for Multi-Agent Intelligence Research. *arXiv preprint*, arXiv:1903.00742.

Li, A. C.; Klassen, T. Q.; Wang, A.; Alamdari, P. A.; and McIlraith, S. A. 2025. Ground-Compose-Reinforce: Grounding Language in Agentic Behaviours using Limited Data. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Li, W.; Varakantham, P.; and Li, D. 2023. Generalization through Diversity: Improving Unsupervised Environment Design. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Liu, J. X.; Yang, Z.; Idrees, I.; Liang, S.; Schornstein, B.; Tellex, S.; and Shah, A. 2023. Grounding Complex Natural Language Commands for Temporal Tasks in Unseen Environments. In *Proceedings of the Conference on Robot Learning (CoRL)*.

Luketina, J.; Nardelli, N.; Farquhar, G.; Foerster, J.; Andreas, J.; Grefenstette, E.; Whiteson, S.; and Rocktäschel, T. 2019. A Survey of Reinforcement Learning Informed by Natural Language. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Ma, Y. J.; Liang, W.; Wang, G.; Huang, D.; Bastani, O.; Jayaraman, D.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2024. Eureka: Human-Level Reward Design via Coding Large Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Matthews, M. T.; Beukman, M.; Ellis, B.; Samvelyan, M.; Jackson, M. T.; Coward, S.; and Foerster, J. N. 2024. Craftax: A Lightning-Fast Benchmark for Open-Ended Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M. E.; and Stone, P. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21: 181:1–181:50.

OEL Team; Stooke, A.; Mahajan, A.; Barros, C.; Deck, C.; Bauer, J.; Sygnowski, J.; Trebacz, M.; Jaderberg, M.; Mathieu, M.; McAleese, N.; Bradley-Schmieg, N.; Wong, N.; Porcel, N.; Raileanu, R.; Hughes-Fitt, S.; Dalibard, V.; and Czarnecki, W. M. 2021. Open-Ended Learning Leads to Generally Capable Agents. *arXiv preprint*, arXiv:2107.12808.

Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J. N.; Grefenstette, E.; and Rocktäschel, T. 2022. Evolving Curricula with Regret-Based Environment Design. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Pnueli, A. 1977. The Temporal Logic of Programs. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*.

Portelas, R.; Colas, C.; Weng, L.; Hofmann, K.; and Oudeyer, P. 2020. Automatic Curriculum Learning For Deep RL: A Short Survey. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Qiu, W.; Mao, W.; and Zhu, H. 2023. Instructing Goal-Conditioned Reinforcement Learning Agents with Temporal Logic Objectives. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Rutherford, A.; Beukman, M.; Willi, T.; Lacerda, B.; Hawes, N.; and Foerster, J. N. 2024. No Regrets: Investigating and Improving Regret Approximations for Curriculum Discovery. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Sadeghi, F.; and Levine, S. 2017. CAD2RL: Real Single-Image Flight Without a Single Real Image. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint*, arXiv:1707.06347.

Teoh, J.; Li, W.; and Varakantham, P. 2024. Improving Environment Novelty Quantification for Effective Unsupervised Environment Design. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Tuli, M.; Li, A. C.; Vaezipoor, P.; Klassen, T. Q.; Sanner, S.; and McIlraith, S. A. 2022. Learning to Follow Instructions in Text-Based Games. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Vaezipoor, P.; Li, A. C.; Toro Icarte, R.; and McIlraith, S. A. 2021. LTL2Action: Generalizing LTL Instructions for Multi-Task RL. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Wang, R.; Lehman, J.; Clune, J.; and Stanley, K. O. 2019. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. *arXiv preprint*, arXiv:1901.01753.

Yalcinkaya, B.; Lauffer, N.; Vazquez-Chanlatte, M.; and Seshia, S. 2023. Automata Conditioned Reinforcement Learning with Experience Replay . In *Proceedings of the Workshop on Goal-Conditioned Reinforcement Learning (GCRL) at the Conference on Neural Information Processing Systems (NeurIPS)*.

Yalcinkaya, B.; Lauffer, N.; Vazquez-Chanlatte, M.; and Seshia, S. 2024. Compositional Automata Embeddings for Goal-Conditioned Reinforcement Learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Zhang, J.; Lehman, J.; Stanley, K. O.; and Clune, J. 2024. OMNI: Open-endedness via Models of human Notions of Interestingness. In *Proceedings of the International Conference on Learning Representations (ICLR)*.