POLYHEDRAL REINFORCEMENT LEARNING: DEEP RL WITH GEOMETRIC STATE REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

In many reinforcement learning (RL) problems, the state already has geometric structure: feasible regions, guards, reachable sets, or collections of objects are represented explicitly by linear constraints. Standard deep RL often discards this structure by flattening the state into a fixed-length vector, forcing the network to recover relationships already explicit in the problem description. We introduce *polyhedral reinforcement learning*, in which states are finite unions of polytopes and actions induce transitions between them. Representing such states poses a distinctive learning problem: a polytope is a variable-sized, unordered object with several mathematically equivalent but computationally different descriptions. It may be specified by its vertices, by the half-spaces that bound it, by the incidence relations between vertices and facets, or, in structured domains, by a canonical matrix or spatial rendering. We therefore develop representation-aware actor–critic architectures matched to these descriptions: DeepSets over vertices and half-spaces, graph neural networks over relational structure, and convolutional encoders over canonical matrices and rasterised projections. Timed automata provide a controlled comparison in which the same system can be presented through real-valued clock valuations or through its symbolic polyhedral zone. Under the same training budget, point-based agents perform poorly when decisions depend on relations between clocks, while several polyhedral encoders achieve near-perfect success; this advantage largely disappears when such relations are irrelevant. Across five additional domains with intrinsically polyhedral states, no representation dominates universally: performance is strongest when the encoding directly exposes the geometric relationships relevant to the decision. These results show that geometric state representation can provide a powerful inductive bias for RL, and that its benefit depends on the task's relational structure.

1 INTRODUCTION

Deep reinforcement learning is often designed as though the structure of the state must be recovered from data. In many structured decision-making problems, however, that structure is already available. Feasible regions are given by systems of linear constraints, objects by vertices or half-spaces, workspaces by polyhedral partitions, and guards, invariants, and reachable sets by geometric abstractions. Replacing such objects by a fixed-length vector can hide symmetries and relational structure that the problem specification already makes explicit.

This raises a basic representation question: *can an RL agent learn more effectively by operating directly on the geometric objects supplied by the domain?* The issue is not simply whether a sufficiently expressive neural network could reconstruct the relevant geometry from coordinates. Rather, it is whether making that geometry explicit provides a useful inductive bias under realistic training budgets. For example, a decision may depend on whether two regions intersect, which cells share a boundary, or how one clock relates to another. A vector-valued observation leaves the network to *infer* such relationships; a polyhedral description can encode them directly.

We study this question through *polyhedral reinforcement learning*, a setting in which states are finite unions of polytopes and actions induce transitions between them. Polyhedral states arise naturally in motion planning, where free space and contact regions are described by convex sets (Lozano-Pérez, 1983; Deits & Tedrake, 2015; Marcucci et al., 2023); in abstraction-based control, where systems

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

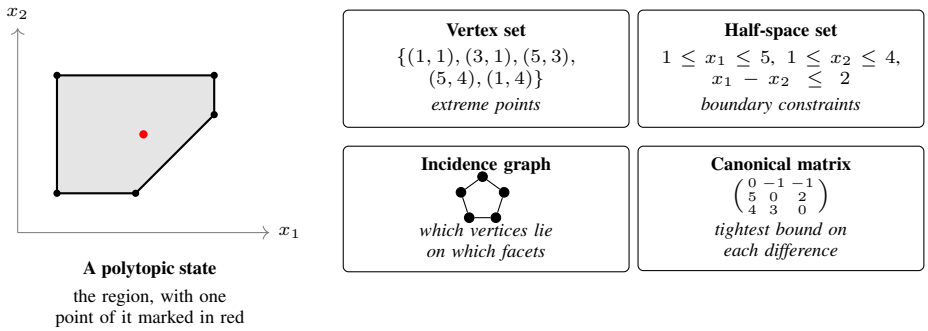

Figure 1: One state, four descriptions. The shaded region is a polytope; in a timed automaton it is a zone, the set of clock valuations consistent with the execution so far, and the red point is one valuation inside it. A conventional RL agent sees only that point. The four boxes give the forms the shaded region itself can take: its vertices, the half-spaces bounding it, the incidence structure of its boundary, and its canonical matrix of pairwise bounds.

are represented by symbolic models over polyhedral cells (Tabuada, 2009; Reissig et al., 2017); and in models such as timed and hybrid automata (Alur & Dill, 1994; Alur et al., 1991).

Polyhedral descriptions are central to verification. Hybrid automata and their probabilistic variants represent regions, guards, invariants, and reachable sets through linear constraints (Alur et al., 1991; Henzinger, 1996; Sproston, 2000; Frehse, 2008; Althoff et al., 2021). In timed and probabilistic timed automata, sets of clock valuations are manipulated symbolically as *zones* (Kwiatkowska et al., 2006, 2007; Jurdziński et al., 2009; Norman et al., 2013). These objects are not auxiliary annotations: they are the states on which symbolic verification algorithms operate. An RL agent capable of consuming them directly can therefore interface with a formal model rather than first reducing that model to a conventional coordinate representation.

The same phenomenon appears in less symbolic domains. In Tangram (Yamada et al., 2019), pieces and target regions are convex polygons; in Rush Hour (Fogleman, 2018), vehicles are axis-aligned rectangles constrained by their spatial relationships; and in navigation over a partitioned workspace, decisions depend on which cells border the current one and what those neighbouring cells contain. In each case, the natural state is a structured collection of geometric objects, and the quantities relevant to control are often relationships among those objects.

A representation problem. Exploiting this structure is not as simple as feeding the coefficients of a polytope to an MLP. A polytope has no unique fixed-length representation. It may be described by its vertices, by the half-spaces defining its boundary, or by the incidence relations between vertices and facets. These descriptions are variable-sized and invariant to arbitrary reorderings, and they expose different aspects of the same geometric object. In structured settings, a polytope may additionally admit a canonical matrix representation or a spatial rendering. Figure 1 illustrates these alternatives. The shaded region is a single geometric state, yet its vertex, constraint, incidence, and matrix descriptions present very different inputs to a learning algorithm. This leads to a second question beyond whether geometry helps at all: *which representation makes the decision-relevant geometry easiest for a policy network to exploit?*

To study these questions, we formalize the setting as a *polytopic Markov decision process* (PMDP), whose states are finite unions of rational polytopes and whose transitions are induced by transformations of those regions. The formalism provides a common interface for domains where geometric state descriptions arise from the model. We characterize computational properties of PMDPs: sufficiently expressive transformations yield infinite-state systems with undecidable exact value-threshold problems, while discounted objectives are approximable via finite-horizon truncation.

We then develop neural encoders matched to the natural representations of a polytope. DeepSet encoders (Zaheer et al., 2017) operate on unordered vertex and half-space sets; graph neural networks encode relational structure; and convolutional networks process canonical matrices and rasterised projections. These encoders form the representation layer of otherwise standard policy and value networks and can therefore be paired with conventional deep RL algorithms. Our experiments use PPO (Schulman et al., 2017) and DQN (Mnih et al., 2015).

When does geometric representation help? Timed automata (Alur & Dill, 1994) provide a controlled setting for addressing this question. The same decision problem can be exposed to the learner through concrete real-valued clock valuations or through the symbolic zones computed by the automaton's standard semantics. The point-valued observation distinguishes reachable concrete states, so poor performance cannot be attributed simply to state aliasing. What changes is the representation: relations among clocks that must be inferred from individual values are explicit in the constraints defining a zone. The resulting separation is striking. On GridTimed and TRMGrid, where successful decisions depend on relations among clocks, point-based agents perform poorly under a fixed training budget, while several polyhedral representations learn substantially stronger policies and in some instances approach perfect success. On ModeSel, where the optimal decision can largely be recovered without those relations, the advantage largely disappears. We complement this comparison with a diagnostic computed from the exact optimal policy before training: it measures how much of the policy can be recovered without access to the relevant relational clock information. The diagnostic separates the cases in which polyhedral representation provides a large empirical advantage from the case in which it does not. Thus, the scope of the phenomenon can be connected to a property of the decision problem rather than identified only after observing learning curves.

Which polyhedral representation should be used? We next compare the encodings on five domains whose states are intrinsically polyhedral, including Tangram, Rush Hour, partition-based navigation, and two hybrid-system benchmarks. Here the answer is deliberately less uniform: no representation dominates across all domains. Instead, the clearest differences appear when a representation exposes precisely the relationships required by the decision. In partition-based navigation, for example, the action depends on which cells border the current cell and what those neighbours contain. An encoder supplied directly with this neighbourhood information performs better than alternatives, including a graph encoder built from the same underlying adjacency relation. Across the experiments, the consistent lesson is therefore not that one geometric encoding is universally preferable, but that representation should expose the structure on which the policy's decisions depend.

Our perspective is related to symbolic reinforcement learning, which similarly gives the learner access to structure already present in the problem specification. Regular reinforcement learning represents symbolic states using regular languages and transitions using rational transductions (Dohmen et al., 2024), while reward-machine methods expose temporal task structure through automata (Icarte et al., 2018; 2022). Physics-informed, timed, and continuous-time reward machines further incorporate known physical or temporal structure (Ajeleye et al., 2025; Falah et al., 2025; Majumdar et al., 2025). Polyhedral reinforcement learning is complementary: rather than exposing primarily linguistic or temporal structure, it treats geometric state descriptions given by linear constraints as first-class inputs to the learner.

Contributions. Our contributions are threefold. First, we introduce polyhedral reinforcement learning and formalize it through polytopic MDPs, providing a common model for decision processes whose states are finite unions of polytopes and establishing basic computational and approximation properties. Second, we develop representation-aware neural architectures for the principal descriptions of polyhedral states, including set, graph, matrix, and rasterised representations. Third, through a controlled comparison on timed automata and experiments on five intrinsically polyhedral domains, we show that geometric representations can provide a substantial learning advantage when the decision depends on relations already explicit in the geometry, and that the effectiveness of a particular encoding depends on how directly it exposes those decision-relevant relations.

2 Polytopic Markov Decision Processes

We formalize decision processes whose states are geometric objects. A *rational polyhedron* in $\mathbb{R}^d$ is a set $H = \{x \in \mathbb{R}^d : Cx \leq b\}$ with $C \in \mathbb{Q}^{m\times d}$ and $b \in \mathbb{Q}^m$; a bounded rational polyhedron is a *rational polytope*. Let Poly_d denote the set of rational polytopes in $\mathbb{R}^d$, and let $\mathsf{UPoly}_d = \{\bigcup_{j=1}^k P_j : k \geq 1,\ P_j \in \mathsf{Poly}_d\}$ be the set of their finite unions, which we call *polytopic regions*.

We use standard discounted-MDP notation. An MDP $M = \langle S, \hat{s}, A, p, r\rangle$ has a countable state space S, initial state $\hat{s} \in S$, finite action set A, transition kernel $p(\cdot \mid s, a)$, and bounded reward $r(s, a)$. For a policy π and discount $\gamma \in [0, 1)$, the value of π from s is $V^\pi_\gamma(s) = \mathbb{E}^s_\pi\left[\sum_{i=0}^\infty \gamma^i r(X_i, Y_{i+1})\right]$, where X_i and Y_{i+1} are the state and action at step i, and the optimal value is $V^*_\gamma(s) = \sup_\pi V^\pi_\gamma(s)$.

Polytopic MDPs. A polytopic MDP makes the geometric representation part of the state itself. An execution starts from a region $P_0 = I$. After the agent chooses an action a_i, the environment samples a geometric transformation θ_i and returns $P_{i+1} = \theta_i(P_i)$. Thus, the same object is both the semantic state of the decision process and the structured input available to the learning algorithm.

Definition 1 (Polytopic MDP)**.** *A* polytopic Markov decision process *(PMDP) is a tuple*

$$\boldsymbol{P} = \langle d, I, \Theta, A, \boldsymbol{p}, \boldsymbol{r} \rangle ,$$

where $d \in \mathbb{N}$ is the ambient dimension, $I \in \mathsf{UPoly}_d$ is the initial region, Θ is a finite set of computable transformations $\theta : \mathsf{UPoly}_d \to \mathsf{UPoly}_d$, A is a finite action set, $\boldsymbol{p} : \mathsf{UPoly}_d \times A \to \mathcal{D}(\Theta)$ assigns a distribution over transformations to every region–action pair, and $\boldsymbol{r} : \mathsf{UPoly}_d \times A \to \mathbb{R}$ is a bounded reward function.

The semantics of $\boldsymbol{P}$ is the countable MDP $[\![\boldsymbol{P}]\!] = \langle S, I, A, p, r \rangle$ whose state space is the orbit $S = \mathsf{Orb}_\Theta(I) = \{\theta_n \circ \cdots \circ \theta_1(I) : n \geq 0,\ \theta_1, \ldots, \theta_n \in \Theta\}$. For $P, P' \in S$ and $a \in A$,

$$p(P' \mid P, a) = \sum_{\theta \in \Theta:\, \theta(P) = P'} \boldsymbol{p}(\theta \mid P, a), \qquad r(P, a) = \boldsymbol{r}(P, a).$$

The discounted value of $\boldsymbol{P}$ is the discounted value of $[\![\boldsymbol{P}]\!]$.

The orbit is countable because every reachable region is generated by a word over the finite transformation set Θ. A PMDP is *finite* when $\mathsf{Orb}_\Theta(I)$ is finite, and *computable* when its transformations, transition probabilities, and rewards can be computed from rational descriptions of its regions.

Computational properties. Finiteness of A and Θ does not imply a finite semantic state space: successive geometric transformations may generate infinitely many distinct regions. With sufficiently expressive transformations, even exact discounted-value questions become undecidable.

Theorem 1. *For every fixed discount factor $\gamma \in (0, 1)$, deciding whether $V^*_\gamma(I) > 0$ is undecidable for computable PMDPs whose transformations can encode rational piecewise-constant-derivative successor maps over finite polyhedral partitions.*

Proof sketch. We reduce from reachability in piecewise-constant-derivative systems, which is undecidable (Asarin & Maler, 1998). Given such a system with initial point x_0 and target point x_f, we construct a deterministic, single-action PMDP whose unique execution follows the corresponding successor evolution. A distinguished absorbing region yields positive reward exactly when the simulated system reaches x_f. Hence $V^*_\gamma(I) > 0$ if and only if x_f is reachable. The full construction appears in Appendix A.1.

Undecidability of exact infinite-horizon questions does not prevent approximation under discounting. If rewards are bounded, the contribution of the distant future decreases geometrically, so only a finite unfolding is needed to approximate the value to any prescribed accuracy.

Theorem 2. *Let $\boldsymbol{P}$ be a computable PMDP satisfying $|\boldsymbol{r}(P, a)| \leq B$ for every reachable P and enabled action a. For every $\gamma \in [0, 1)$ and $\epsilon > 0$, one can compute a finite-horizon value $\widehat{V}$ such that $\left|\widehat{V} - V^*_\gamma(I)\right| \leq \epsilon$. It suffices to use a horizon n satisfying $\frac{\gamma^n B}{1-\gamma} \leq \epsilon$.*

Proof sketch. For every policy, the absolute contribution of rewards from step n onward is at most $\gamma^n B/(1 - \gamma)$. Because A and Θ are finite, the depth-n unfolding contains only finitely many region–time pairs. Dynamic programming on this unfolding therefore computes an n-step optimal value whose error is bounded by the discounted tail. Full details appear in Appendix A.2.

The same finite-unfolding argument also transfers standard finite-horizon learning guarantees to PMDPs. The relevant complexity parameter is the size of the unfolding rather than the ambient dimension alone; this size may grow exponentially with the horizon.

Theorem 3. *Fix $\gamma \in [0, 1)$ and $\epsilon, \delta > 0$, and choose n such that $\frac{\gamma^n B}{1-\gamma} \leq \frac{\epsilon}{4}$. Let N_n be the number of region–time pairs in the depth-n unfolding. Under a generative sampling model, an ϵ-optimal policy for a computable PMDP with finite A and Θ and rewards bounded by B can be learned with probability at least $1 - \delta$ using a number of samples polynomial in $N_n, |A|, n, \frac{B}{\epsilon}$, and $\log \frac{1}{\delta}$.*

Proof sketch. The depth-n unfolding is a finite-horizon MDP. Under the generative model, a standard finite-horizon PAC algorithm can learn a policy whose truncated value is within $\epsilon/2$ of the truncated

optimum with probability at least $1 - \delta$. Extending this policy arbitrarily beyond step n, both its infinite-horizon value and the optimal infinite-horizon value differ from their respective truncated values by at most $\epsilon/4$. Therefore $V^*_\gamma(I) - V^{\tilde{\pi}}_\gamma(I) \leq \frac{\epsilon}{4} + \frac{\epsilon}{2} + \frac{\epsilon}{4} = \epsilon$.

Thus a PMDP need not have a finite semantic state space for discounted planning or learning to admit finite approximations. Discounting restricts the relevant computation to a finite geometric unfolding, although the size of that unfolding can itself be large.

3 Representation-Aware Encoders for Polytopic States

A state of a PMDP is a polytopic region $P = \bigcup_{j=1}^{k} P_j$, with each $P_j \in \mathsf{Poly}_d$. Encoding such a state for deep RL presents two levels of variability. First, each component polytope may itself have a variable-size description. Second, a region may contain a variable number of component polytopes, with no meaningful ordering among them. We therefore seek encoders that produce fixed-dimensional embeddings while remaining invariant to permutations both within each component representation and across the components of the region.

Polytope representations. A rational polytope $P_j \in \mathsf{Poly}_d$ admits two standard descriptions. Its *vertex*, or V-*representation*, is $P_j = \mathrm{conv}\{v_1, \ldots, v_{p_j}\}$, where $v_\ell \in \mathbb{Q}^d$ and conv denotes the convex hull; we write $V(P_j) = \{v_1, \ldots, v_{p_j}\}$. Its *half-space*, or H-*representation*, is $P_j = \{x \in \mathbb{R}^d : C_j x \leq b_j\}$, with $C_j \in \mathbb{Q}^{m_j \times d}$ and $b_j \in \mathbb{Q}^{m_j}$. Writing $c_\ell^\top$ for the ℓ-th row of C_j, we identify this description with the unordered set $H(P_j) = \{(c_1, b_1), \ldots, (c_{m_j}, b_{m_j})\}$, where (c_ℓ, b_ℓ) denotes the constraint $c_\ell^\top x \leq b_\ell$. Although these descriptions define the same geometric object, they expose different information to a learner: vertices make extreme points and absolute geometry explicit, whereas half-spaces expose the constraints that define the boundary. We therefore treat them as distinct inputs rather than converting them to a common flattened representation.

DeepSet encoders. Let $X(P_j) = \{x_1, \ldots, x_{n_j}\}$ denote either unordered description of a component polytope: $x_i \in \mathbb{Q}^d$ for a V-representation and $x_i = (c_i, b_i) \in \mathbb{Q}^{d+1}$ for an H-representation. Following DeepSets (Zaheer et al., 2017), we use two nested permutation-invariant aggregations,

$$e_{\mathsf{poly}}(P_j) = \rho_{\mathsf{poly}}\left(\bigoplus_{x \in X(P_j)} \phi_{\mathsf{poly}}(x)\right), \qquad e_{\mathsf{reg}}(P) = \rho_{\mathsf{reg}}\left(\bigoplus_{j=1}^{k} \phi_{\mathsf{reg}}(e_{\mathsf{poly}}(P_j))\right),$$

where the ϕ networks are shared across elements, $\bigoplus$ is a permutation-invariant pooling operator, and each ρ maps the pooled features to a fixed-dimensional embedding. The first aggregation removes dependence on the ordering of vertices or constraints within a component; the second removes dependence on the ordering of component polytopes. The resulting embedding $e_{\mathsf{reg}}(P)$ can be supplied to a policy $\pi_\eta(\cdot \mid P)$, a value function $V_\eta(P)$, or a shared actor–critic network.

Graph encoders. Set encoders capture the elements of a polytope but do not explicitly expose how those elements are related. We therefore also consider graph representations whose edges encode geometric relations. A generic construction starts from the facet structure of a polytope. Suppose that P_j is available through both $V(P_j)$ and $H(P_j)$. We construct a facet-adjacency graph $G(P_j) = (N, E)$ with one node n_i for each constraint (c_i, b_i) and connect n_i and n_ℓ whenever the corresponding facets meet, i.e., whenever there is $v_q \in V(P_j)$ with $c_i^\top v_q = b_i$, and $c_\ell^\top v_q = b_\ell$. A graph neural network (Kipf & Welling, 2017) performs message passing on $G(P_j)$ and aggregates the resulting node embeddings to produce $e_{\mathsf{poly}}(P_j)$. The component embeddings are then pooled as above to obtain a representation of the full region.

The graph construction can also exploit relational structure present in a domain. In the timed automata, for instance, the natural graph has pairwise difference bounds as nodes and clock valuations as edges. Thus the common principle is not a particular graph topology, but the use of message passing when the state description provides explicit relations among its geometric variables or components.

Convolutional encoders. The preceding encoders operate on variable-size, unordered descriptions, but in structured domains a polyhedral state can also admit an array-valued representation with

meaningful indexing, for which convolutional networks provide a natural alternative (LeCun et al., 1998). The first such representation arises for systems of difference constraints such as $x_i - x_j \leq c$: recording the tightest bound for every ordered pair and closing the system under implied bounds yields the canonical *difference bound matrix* (DBM) used in timed-system verification (Dill, 1989), and we encode the resulting $(d+1) \times (d+1)$ array as a small multi-channel image.

The second is based on spatial rendering. In two dimensions we rasterise a polytopic region on a fixed grid, marking the pixels inside it; in higher dimensions we rasterise its pairwise coordinate projections and stack the resulting images as channels, and when a state contains several distinguished objects, such as puzzle pieces or vehicles, each may be rendered in its own channel. Unlike the set and graph representations, rasterisation trades exact geometry for a fixed-size spatial input suited to convolutional architectures.

Relational augmentation. The encoders above describe the geometry of individual components, but in some domains the optimal decision depends on relations *between* them, such as which polytopes share boundaries. We therefore allow a component embedding to be augmented with an explicit relational descriptor $r(P_j) \in \mathbb{R}^q$, replacing $e_{\mathsf{poly}}(P_j)$ by

$$\widetilde{e}_{\mathsf{poly}}(P_j) = \psi\big(e_{\mathsf{poly}}(P_j) \,\|\, r(P_j)\big),$$

where $\|$ denotes concatenation and ψ is a small neural network; the region-level aggregation is unchanged. In the navigation domain of Section 4.2, $r(P_j)$ summarizes the facet-adjacency relation with five quantities: the number of cells bordering P_j, how many of these are hazard cells and how many are goal cells, the total number of shared facets, and the total length of the shared boundary. These are deliberately coarse one-hop statistics that neither identify individual neighbours nor preserve the adjacency graph, so comparing this encoder with message passing over the same graph tests whether a compact summary of the decision-relevant relations suffices, or whether the richer structure exposed by a GNN adds further benefit.

4 Experimental Evaluation

Our evaluation addresses two questions. Section 4.1 asks whether exposing a polyhedral state description can improve learning at all. Timed automata provide a controlled comparison: the same underlying system can be presented either through concrete clock valuations or through the polyhedral zones used by its symbolic semantics. Section 4.2 then asks which polyhedral representation is most useful, across five domains whose states are geometric by construction. Complete benchmark specifications, architectures, hyperparameters, and additional results appear in Appendix B.

4.1 Polyhedral State Encoding on Timed Automata

A timed automaton (Alur & Dill, 1994) is a finite automaton equipped with real-valued clocks that advance uniformly with time. Transitions carry *guards*, which constrain when they may be taken, and *resets*, which set selected clocks to zero. A concrete state consists of a discrete location together with the current clock valuation, yielding an uncountable state space. Symbolic analysis instead groups valuations into *zones*, sets defined by difference constraints such as $x_i - x_j \leq c$. Zones are represented canonically by the difference bound matrices of Section 3. In our benchmarks, the reachable zones are polytopes, and the standard symbolic operations—time elapse, guard intersection, and clock reset—map zones to zones. The resulting symbolic transition system is therefore naturally represented as a PMDP.

Benchmarks. We use three timed decision problems, with two instances of each, evaluated over ten seeds on GridTimed and TRMGrid and three on ModeSel.

- **GridTimed.** The agent traverses a grid containing switches and timed gates. Pressing switch i resets clock x_i, and the corresponding gate can be entered only while $x_i \in [a_i, b_i]$; arriving either too early or too late prevents entry. A global clock imposes a deadline. Each action combines one of four movement directions with a delay of 0, 1, or 2 time units, giving $|A|$=12. Successful decisions depend on elapsed time relative to the windows opened by previous switch activations.
- **TRMGrid.** The agent must visit two of three sub-goals and then reach the exit while satisfying a timed reward-machine specification. Each ordered pair of sub-goals has its own timing window,

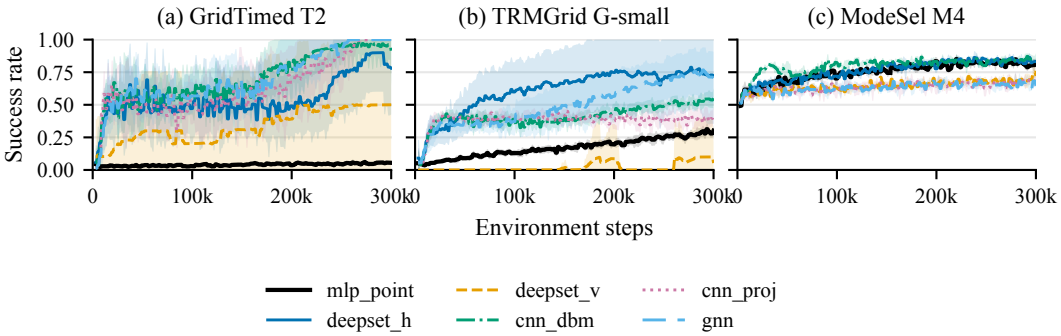

Figure 2: Learning curves on one instance of each benchmark: (a) GridTimed T2, (b) TRMGrid G-SMALL, (c) ModeSel M4. The horizontal axis is environment steps, out of a budget of 300,000; the vertical axis is the fraction of episodes that succeeded, as a moving average over 200 episodes. Lines are means across seeds, ten on GridTimed and TRMGrid and three on ModeSel, and bands show one standard deviation. The non-symbolic baseline `mlp_point` is drawn in black. The wide band on `deepset_v` marks seeds reaching different outcomes, not run-to-run noise.

and the second visit imposes a bound on the remaining time to reach the exit. The three sub-goals induce six possible routes whose feasibility changes with elapsed time. Unlike GridTimed, the main decision is which feasible branch of the temporal specification to commit to.

- **ModeSel.** The agent schedules sequences of tasks on machines. Each task may run in a fast mode that incurs a penalty or a slow mode without that penalty, while each job must satisfy its own deadline relative to a global clock. Actions are pairs that select a task and execution mode.

Encoders. We compare five representations of the zone with one point-valued baseline. `deepset_h` and `deepset_v` encode its half-space and vertex representations, respectively; `cnn_dbm` encodes its difference bound matrix; `cnn_proj` encodes rasterised pairwise projections; and `gnn` operates on a graph whose nodes are clocks and whose edges carry pairwise difference bounds. The baseline, `mlp_point`, is an MLP over the current clock values and discrete location and is the only model that does not receive the zone.

Training and evaluation. All agents are trained with PPO and generalised advantage estimation (Schulman et al., 2015) for a fixed budget of 300,000 environment steps, using shared hyperparameters across encoders and benchmarks. An episode succeeds when the goal is reached or, for ModeSel, when every job meets its deadline. We report success during training and under deterministic evaluation of the trained policy. GridTimed and TRMGrid are deterministic, so deterministic evaluation produces a single trajectory for each trained seed; the resulting solve rate is therefore the fraction of seeds that learned a successful policy.

Results. Figure 2 shows learning curves and Table 1 final success rates for all six instances, compared at a fixed sample budget since several encoders are still improving when training ends. The largest separation is on GridTimed: although its observation distinguishes every concrete state, the point baseline reaches only 3.9–5.4% success, and its greedy policies cycle until the deadline without reaching the goal, whereas several polyhedral encoders reach 90% or more and `cnn_proj` solves both instances from every seed. TRMGrid shows a smaller gap under a different decision structure, with the baseline at 28–31% against about 73% for the best polyhedral encoders, although the advantage is not uniform: `deepset_v` performs poorly on both instances. ModeSel provides the contrast. There the baseline reaches 89% and 80%, competitive with the best polyhedral encoders, so access to a polyhedral representation does not by itself guarantee an advantage.

When does the polyhedral representation help? The three benchmarks suggest that the relevant distinction is not whether a state admits a zone representation, but whether good decisions depend on the relations that the zone makes explicit. To test this explanation independently of learning, we solve each instance exactly before training and measure how much of its optimal policy can be recovered from restricted state information.

On ModeSel, the discrete configuration already determines most optimal action choices, and adding coarse information about elapsed time recovers almost all of them. GridTimed and TRMGrid behave differently: whether to approach a gate, which temporal branch remains feasible, and how much

Table 1: Final success rate by encoder and instance, measured over the last 200 training episodes. `mlp_point` is the non-symbolic baseline; the other five encode the same zone in different forms.

	GridTimed		TRMGrid		ModeSel	
Encoder	T2	T3	G-SMALL	G-LARGE	M3	M4
`mlp_point`	0.054	0.039	0.307	0.282	0.887	0.797
`deepset_h`	0.839	0.900	0.731	0.671	0.888	0.873
`deepset_v`	0.498	0.200	0.008	0.001	0.867	0.673
`cnn_dbm`	0.928	0.761	0.534	0.716	0.905	0.855
`cnn_proj`	1.000	1.000	0.375	0.736	0.820	0.680
`gnn`	1.000	0.893	0.733	0.527	0.837	0.648

timing slack remains depend directly on relationships among clock values. These relationships appear explicitly as pairwise constraints in a zone but must be reconstructed from individual clock values by the point-based network. The diagnostic therefore separates the benchmark on which the point baseline is competitive from those exhibiting a substantial representation gap.

Differences among polyhedral encodings. The five polyhedral representations are themselves not interchangeable. `cnn_proj` is particularly effective and stable on GridTimed, solving both instances on every seed even though pairwise projections do not preserve all of the exact difference-bound information available to other representations. This suggests that the policy requires a coarser geometric signal than the complete system of bounds. Its advantage does not carry uniformly to TRMGrid. The vertex representation shows the opposite behavior: `deepset_v` is competitive on ModeSel, but its GridTimed runs split between successful and unsuccessful seeds and it almost entirely fails on TRMGrid. Among the representations tested here, the half-space encoding is the most consistently competitive across all three benchmark families.

4.2 Domains with Native Polytopic Structure

Timed automata let us compare a polyhedral description against a concrete point-valued one. We now consider domains in which the geometric objects are part of the state description itself: puzzle pieces, vehicles, partition cells, and modes of hybrid systems. The question is no longer whether to expose polyhedral structure, but which representations are most useful.

- **Tangram.** The state consists of convex polygonal pieces that the agent moves one grid step at a time until each covers its designated region in a target silhouette. Two tiers vary the number and shape of the pieces, from four squares to two squares and four triangles.
- **Rush Hour.** Vehicles are axis-aligned rectangles on a 6×6 board, each constrained to move along its own axis without overlapping with another vehicle. The agent must clear a path for the target vehicle to exit. We use puzzles from an exhaustive configuration database (Fogleman, 2018), with three difficulty levels corresponding to optimal solution lengths of ten, twelve, and fifteen moves and fifteen puzzles at each level.
- **PHA Navigation.** A square workspace is partitioned into convex cells. The agent has continuous position and velocity, chooses among eight directions, and follows a linear drift until it enters a neighbouring cell. Goal cells are rewarded and hazard cells penalised. Three tiers range from a regular unit grid to an irregular tessellation.
- **Lawn Mower.** The workspace is partitioned into rectangular cells. The mower travels diagonally, and agent chooses how it reflects at cell boundaries. Newly covered cells are rewarded, while entering a hazard terminates the episode. Three tiers increase the grid from twelve to thirty cells.
- **Water Level.** We adapt the standard water-level-monitor hybrid automaton of Alur et al. (1995). Its six modes are represented by polytopes over the water level and a clock.

Encoders and evaluation. H-Rep, V-Rep, and GNN are the encoders of Section 3. Tangram, Rush Hour, and PHA Navigation add a convolutional encoder over a rasterised state, and PHA Navigation adds Relational, H-Rep augmented with the one-hop statistics of Section 3. Training uses PPO, or DQN on Lawn Mower and Water Level, with the same budget for every encoder in a domain. We use ten seeds, except five on PHA Navigation; Rush Hour instead replicates over its fifteen puzzles per difficulty.

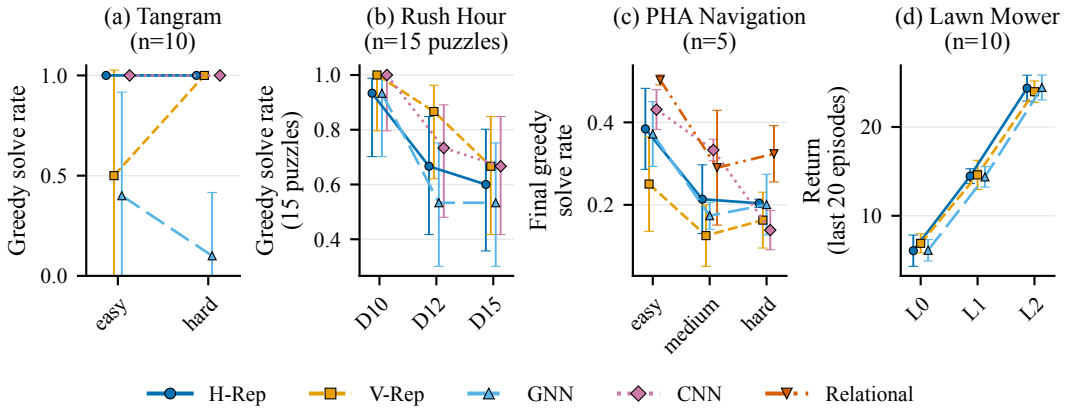

Figure 3: Final performance across difficulty tiers in four polyhedral domains. Each panel uses the metric and vertical scale appropriate to that domain. Points are means across seeds except for Rush Hour, where they denote solve rates over fifteen puzzles; error bars show one standard deviation or, for Rush Hour, a 95% confidence interval. Seed counts are shown in the panel titles.

Results. Figure 3 summarizes the comparison. On Tangram, H-Rep and the convolutional encoder solve every tier on every seed, whereas the graph encoder solves only one of ten seeds on the hardest tier. On Rush Hour, solve rates fall as solutions lengthen, but the confidence intervals overlap at every difficulty, so we claim no ordering; the encoders are similarly indistinguishable on Lawn Mower. Water Level is non-discriminating by design, since only two of its six modes offer a choice and one action dominates at both; we include it to show how a typical hybrid automaton fits the PMDP formalism. PHA Navigation gives the clearest effect: Relational has the highest mean on the easy and hard tiers and exceeds the graph encoder on all three, although it receives only a one-hop summary of the adjacency over which the graph encoder passes messages. Solve rates there remain modest and are still rising when the budget ends, so these are fixed-budget comparisons.

What the comparison shows. No representation dominates across the native-polyhedral domains. H-Rep is never the lowest-performing encoder, consistent with its comparatively stable behavior on the timed benchmarks, but the largest differences arise when an encoding directly exposes the relationships needed for control. PHA Navigation is the clearest example: actions depend on which cells border the current region and on the contents of those neighbours, and the encoder supplied with a compact summary of exactly this information performs best on two of the three tiers. Taken together with the timed-automata experiments, the results support a more specific conclusion than "polyhedral representations help." Their benefit is largest when the policy depends on geometric relationships that the representation makes explicit; when those relationships are irrelevant or easily reconstructed from simpler observations, the advantage can disappear. The choice of representation therefore acts as a task-dependent inductive bias rather than a universally superior encoding.

5 Conclusion

This work treats the polyhedral description of a state as something a reinforcement-learning agent can learn from directly. Polytopic MDPs are expressive enough that exact value questions become undecidable, yet discounted objectives remain approximable and learnable through a finite unfolding, so learning remains possible where exact analysis is not. Our experiments suggest three lessons. First, how the state is described can decide whether a task is learned at all: on two timed benchmarks, the same state given as a zone was learned by several encoders, while given as a vector of clock values it was solved by no seed's final policy within the same budget. Second, among polyhedral descriptions, the half-space form is a dependable default, never the weakest across the domains we studied, even though the best choice varies with the task. Third, where the right action depends on how polytopes relate to one another, stating those relations explicitly helps more than leaving them to be discovered: a set encoder given a few neighbourhood statistics outperformed a graph network with access to the full adjacency. Together these point to a clear direction for future work, namely encoders tailored to polytopic domains that expose the relations a task depends on, and ultimately methods that learn which relations to expose instead of relying on hand-chosen features.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on the design of experiments and the evaluation protocol, including seed counts, verification runs, and the choice of statistical tests; and processing experimental results, by extracting benchmark specifications from our code and consolidating per-run training logs into summary tables. We have not used generative AI tools to generate synthetic datasets or for translation, and qualitative or thematic data analysis is not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: polishing parts of the paper and suggesting its structure; modifying figures and writing the code that produces them; searching for information; and proposing keywords and a summary.

We have reviewed all AI-assisted work. All text was reviewed and revised by the authors. Figures were regenerated from the raw training logs, and every plotted value was checked against independently extracted result tables, with any disagreement reported rather than overwritten. Benchmark specifications extracted from our code were required to cite the source location of each value or to report it as unknown. Statistical tests were recomputed from stored per-puzzle outcomes. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The PMDP formalism and its assumptions are stated in Section 2, and complete proofs of Theorems 1–3 appear in Appendix A. Every benchmark is specified in full in Appendix B, including dynamics, rewards, termination conditions, and instance parameters; all instances are fixed in advance, and the Rush Hour puzzles are drawn by exact optimal move count from a public database (Fogleman, 2018). Encoder architectures, parameter counts, training hyperparameters, and the evaluation protocol are given in Appendix C; every encoder within a given domain and tier receives the same sample budget, with no adaptive stopping. The supplementary material contains anonymised source code for all environments, encoders, training, and evaluation, together with the per-episode training logs and the scripts that produce every figure and table.

REFERENCES

Daniel Ajeleye, Ashutosh Trivedi, and Majid Zamani. Physics-informed reward machines. *arXiv preprint arXiv:2508.14093*, 2025.

Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):369–395, 2021.

Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2): 183–235, 1994.

Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *International hybrid systems workshop*, pp. 209–229. Springer, 1991.

Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138(1):3–34, 1995.

Eugene Asarin and Oded Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398, 1998.

Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Algorithmic Foundations of Robotics XI*, pp. 109–124. Springer, 2015.

David L Dill. Timing assumptions and verification of finite-state concurrent systems. In *International Conference on Computer Aided Verification*, pp. 197–212. Springer, 1989.

Taylor Dohmen, Mateo Perez, Fabio Somenzi, and Ashutosh Trivedi. Regular reinforcement learning. In *International Conference on Computer Aided Verification*, pp. 184–208. Springer, 2024.

Amin Falah, Shibashis Guha, and Ashutosh Trivedi. Continuous-time reward machines. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 5056–5064, 2025.

Michael Fogleman. Rush hour puzzle database and solver. https://www.michaelfogleman.com/projects/rush/, 2018. Accessed: May 3, 2026.

Goran Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *International journal on software tools for technology transfer*, 10(3):263–279, 2008.

Thomas A Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pp. 278–292. IEEE, 1996.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2107–2116. PMLR, 2018.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Marcin Jurdziński, Marta Kwiatkowska, Gethin Norman, and Ashutosh Trivedi. Concavely-priced probabilistic timed automata. In *International Conference on Concurrency Theory*, pp. 415–430. Springer, 2009.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Marta Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1): 33–78, 2006.

Marta Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. Symbolic model checking for probabilistic timed automata. *Information and Computation*, 205(7):1027–1077, 2007.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Tomás Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

Anirban Majumdar, Ritam Raha, Rajarshi Roy, David Parker, and Marta Kwiatkowska. About time: Model-free reinforcement learning with timed reward machines. *arXiv preprint arXiv:2512.17637*, 2025.

Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. *Science Robotics*, 8(84), 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Gethin Norman, David Parker, and Jeremy Sproston. Model checking for probabilistic timed automata. *Formal methods in system design*, 43(2):164–190, 2013.

Gunther Reissig, Alexander Weber, and Matthias Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jeremy Sproston. Decidable model checking of probabilistic hybrid automata. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 31–45. Springer, 2000.

Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

Fernanda Miyuki Yamada, Joao Paulo Gois, and Harlen Costa Batagelo. Solving tangram puzzles using raster-based mathematical morphology. In *2019 32nd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pp. 116–123. IEEE, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.