Interpretable Rule Learning for Reactive Activity Recognition in Event-Driven RL Environments

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many real-world environments, from smart homes to industrial systems, produce asynchronous event streams driven by latent activities with complex temporal structures. Recognizing these patterns requires reasoning over temporal dependencies that reactive policies alone do not capture. We propose a hybrid reinforcement learning framework that combines symbolic program synthesis with reactive policy optimization for interpretable activity recognition. This hybrid approach enables the agent to disambiguate overlapping activities, generalize across history patterns, and maintain interpretable decision logic. Our method discovers temporal rules as logical abstractions over event histories, using a compositional grammar based on Allen's interval algebra. Monte Carlo Tree Search (MCTS) explores the rule space, refining candidates to maximize cumulative reward. The resulting rules define symbolic contexts that augment the observable state and support decision-making in a near-Markovian surrogate process. Evaluations on a synthetic benchmark with concurrent, asynchronous activities show strong task performance and symbolic fidelity compared to neural and evolutionary baselines.

1 Introduction

In many real-world environments, decisions are not triggered by time but by the occurrence of structured event sequences, requiring agents to recognize and respond to temporal patterns in interaction histories rather than static observations. This is especially true in domains such as activity recognition in smart homes or sensor-rich industrial systems, where streams of asynchronous events encode underlying physical or behavioral processes [4, 18, 11]. In such settings, an event denotes a discrete unit of observation, while an activity refers to a temporally extended pattern comprising multiple asynchronous but temporally related events. Environments are characterized by multiple overlapping activities that emit asynchronous, exogenous events in variable-duration patterns that defy simple clock-driven modeling [5, 26]. In this case, decision-making hinges on recognizing and reacting to these irregular time series rather than relying on fixed-time-step transitions [28, 7]. For example, in Complex Event Processing (CEP) systems [2], rules defined over symbolic event patterns drive detection and action, much like in Generalized Semi-Markov Decision Processes (GSMDPs), where concurrent temporal processes guide behavior [33].

At its core, reinforcement learning can be understood as the problem of learning when and how to react to a recognized structure in the environment. In the Markovian case, this reduces to identifying the current state and choosing an appropriate response. In more complex temporal settings, however, recognition involves detecting extended patterns that unfold over time, often without clear or bounded durations. While time-series models can handle bounded intervals, they often fall short in representing and generalizing complex activities [4]. Despite the success of recurrent and attention-

Submitted to the 18th European Workshop on Reinforcement Learning (EWRL 2025). Do not distribute.

based architectures, models typically lack the capacity for symbolic abstraction and compositional generalization [34], making them ill-equipped for critical environments where interpretability is key.

To overcome these limitations, we examine encoding histories into symbolic structures that capture both temporal and attribute-level dependencies between observations. Temporal logics and automata have been used to specify non-Markovian reward structures [3, 13, 31], while symbolic regression and program synthesis methods such as Monte Carlo Tree Search (MCTS) and large language models (LLMs) [16, 29, 15] have shown promise in learning structured policies from data. These approaches are theoretically supported by the idea of learning representations that generalize across interaction histories [20]. Still, most existing methods either rely on predefined symbolic specifications or lack mechanisms to learn such a structure directly from interaction.

In this work, we propose a reinforcement learning framework for event-driven environments that learns symbolic rule representations over event histories. These rules capture recurring temporal patterns and associate them with environment actions, transforming the original non-Markovian task into an approximate MDP over symbolic contexts. They are discovered through a top-down structured search over a compositional grammar, optimized using MCTS guided by cumulative reward signals in the induced environment.

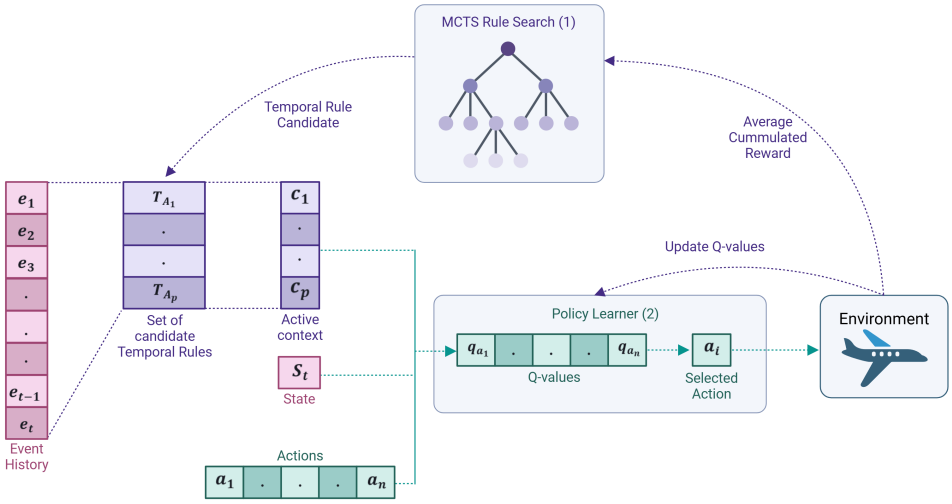

Figure 1: Overview of the proposed two-level architecture. The system receives a stream of temporally-annotated events, which are processed by a symbolic rule abstraction module to produce a latent context. This context, combined with the current observable state, is passed to a policy module that selects actions. The rule abstraction module is optimized through Monte Carlo Tree Search (MCTS) using feedback from policy performance.

2 Related Work

History-Based Decision Processes. A wide range of extensions to the standard Markov Decision Process (MDP) framework have been developed to address history-dependent behavior. These include Dynamic Contextual MDPs (DCMDPs) [30], Context MDPs [12], Activity-Based MDPs (ABMDPs) [28], and ϕ-MDPs [14], all of which modify state representations or transition dynamics to incorporate information from past trajectories. Such frameworks aim to capture non-Markovian dynamics by summarizing historical observations or actions, either through predefined mappings or learned abstractions. In parallel, deep learning models have tackled similar challenges by modeling full interaction histories using recurrent architectures or attention-based mechanisms, such as Transformers [8].

Representation Learning and Automata-Based Abstractions. Another prominent line of work focuses on learning compact representations of latent state and history through interaction [20]. Techniques in this domain aim to improve sample efficiency and generalization by discovering

abstractions that retain predictive or decision-relevant information. Examples include bisimulation metrics, self-predictive encodings, and compressed belief states. Automata and temporal logic frameworks have been employed to model non-Markovian structure, particularly in settings with complex temporal rewards or dependencies [3, 31]. For example, automata-augmented POMDPs and symbolic observation detectors have been used to impose trajectory-level constraints or express high-level task structure [27].

Symbolic Reinforcement Learning and Interpretable Policies. Symbolic reinforcement learning aims to produce effective, human-interpretable policies, typically represented as logic rules or algebraic programs. Evolutionary systems such as XCS and symbolic regression frameworks have long been used for extracting rule-based policies from environment interaction, but often struggle with scalability in high-dimensional or temporally complex domains [17, 25]. Recent relational RL approaches use structured representations over states and actions [34], however they lack the temporal abstraction needed for reasoning over event sequences or delays. Temporal logic specifications, most notably Linear Temporal Logic (LTL), have also been applied to guide behavior in non-Markovian settings, but are generally used as predefined constraints [13].

Symbolic Program Search and Neuro-Symbolic Policy Induction. A growing class of methods treats policy learning as a symbolic program synthesis task, where the agent searches over structured rule spaces to discover effective behavior [32, 19]. Techniques using MCTS and guided symbolic regression have been applied to this end, enabling tractable exploration of combinatorial symbolic policy spaces [29, 15]. Recent neuro-symbolic approaches combine neural function approximators with top-down symbolic rule search, aiming to benefit from both flexibility and structure [22, 9]. However, most existing work in this area concentrates on symbolic-physics dynamics or state-based decisions, lacking mechanisms for learning general-purpose temporal rules from raw interaction data.

3 Background

This section outlines the formal representations and models, including the structure of event histories, symbolic temporal rules for activity recognition, and abstraction-based frameworks for reinforcement learning in non-Markovian environments.

Event Histories and Rule-Based Activity Recognition. In event-driven environments, agents do not observe fixed state snapshots but rather streams of temporally structured events. We formalize this interaction history as a discrete event sequence $h_t = \{e_1, \ldots, e_t\}$, where each event e_i is described by its type and temporal interval: $e_i = (\text{type}, [t_{\text{start}}, t_{\text{end}}])$.

Temporal Logic and Event-Based Representations. While multiple temporal logics exist for modeling time-dependent phenomena [21], Allen's Interval Algebra [1] is particularly well-suited for event-driven settings that involve reasoning over time intervals. It defines a set of 13 primitive relations, such as *before, overlaps, during,* and *meets*, allowing formal expression of how two events relate over time. In our context, each event e_i is associated with a time interval $[t_i^{start}, t_i^{end}]$, and relations are defined by a binary predicate $R_{\text{allen}}(e_i, e_j)$ that holds when the pair (e_i, e_j) satisfies a specific temporal relation:

$$R_{\text{allen}}(e_i, e_j) \equiv R(t_i^{start}, t_i^{end}, t_j^{start}, t_j^{end}).$$

To represent activity patterns within histories, we define symbolic rules that isolate relevant events. Each rule is evaluated as a conjunction of type filters $F_i(e_i)$ and Allen-style relational constraints $R_{i,j}(e_i, e_j)$:

$$T(e_1, \ldots, e_n) = \bigwedge_{i=1}^{n} F_i(e_i) \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} R_{i,j}(e_i, e_j). \tag{1}$$

Where an event type filter $F_i(e)$ checks whether an event e matches a required type label (e.g., 'door-open', 'signal-high').

History-Based Abstractions in RL. Reinforcement learning in non-Markovian environments requires compressing histories into decision-relevant summaries. A general class of models known as Feature MDPs (ϕMDPs [14]) formalize this via a mapping $\phi : \mathcal{H} \to \mathcal{S}_\phi$, where $\mathcal{H}$ denotes histories and $\mathcal{S}_\phi$ an abstract state space.

Activity-Based MDPs and Rule Abstractions. In this work, we adopt the Activity-Based Markov Decision Process (ABMDP) [28] as a formalism for capturing symbolic structure in event-driven environments. An ABMDP is defined as:

$$\mathcal{M} = (\mathcal{S}, \mathcal{C}, \mathcal{A}, P, R, T),$$

where $\mathcal{S}$ is the observable state space, $\mathcal{C}$ is a set of latent contexts, $\mathcal{A}$ is the set of actions, and T is a set of symbolic temporal rules. A context $c^* \in \mathcal{C}$ is induced from the history h by applying the rule set via the context extraction function $\mathcal{T}(h, T)$. Planning then proceeds in a context-augmented space using policies of the form $\pi(a \mid s, c^*)$. This framework aligns with the ϕMDP model [14], where $\mathcal{T}(h, T)$ can be viewed as an abstraction mapping $\phi(h)$, inducing a surrogate decision process $\mathcal{M}$ over the context-augmented space.

Q-Value Uniformity and Approximation Guarantees. A central theoretical insight in abstraction-based planning is *Q-value uniformity*: if two histories h, h' are mapped to the same abstract state by an abstraction ϕ, their optimal Q-values should be nearly equal:

$$|Q^*(h, a) - Q^*(h', a)| < \varepsilon, \quad \forall a, \text{ and } \forall h, h' \text{ such that } \phi(h) = \phi(h').$$

Under this condition, [14, Theorem 8] guarantees that the Q-values of a policy $\pi(\phi(h))$ induced by planning in an abstract MDP, where $\Pi(h)$ is a policy over histories, satisfy:

$$\left|Q^{\Pi}(h, a) - q^{\pi}(\phi(h), a)\right| \leq \frac{\varepsilon}{1 - \gamma}.$$

This means the return of the lifted policy in the original process is close to the return in the abstract model:

$$\left|V^{\Pi}(h) - v^{\pi}(\phi(h))\right| \leq \frac{\varepsilon}{1 - \gamma}.$$

Symbolic Rule Discovery via MCTS. Monte Carlo Tree Search incrementally builds a search tree using simulated rollouts and selects expansions according to the UCT criterion [6], allowing to explore compositional rule spaces:

$$n' = \arg\max_{n_i} \left[Q(n_i) + c \cdot \sqrt{\frac{\log N(n)}{N(n_i)}} \right],$$

where $Q(n_i)$ is the value of a node, $N(n)$ and $N(n_i)$ are visit counts, and c balances exploration and exploitation.

4 Method

We propose a two-level architecture that integrates symbolic temporal abstraction search with reinforcement learning to enable interpretable and reactive behavior in event-driven environments. The system is composed of two interacting modules: (1) a symbolic rule learning and context induction module, and (2) a policy learning module that operates over induced contexts (see Fig. 1).

1. The **rule abstraction module**'s primary goal is to learn a set of symbolic temporal rules $\hat{T}$ that map event histories h_t into discrete contexts $\hat{c}_t$, each representing a structured activity. Using Monte Carlo Tree Search (MCTS) over a compositional grammar, the module refines rules to maximize reward, treating rule discovery as a symbolic program synthesis task.
2. The **policy learning module** trains a reactive policy $\pi(a \mid s_t, \hat{c}_t)$ over the augmented state space, where $\hat{c}_t$ encodes latent temporal structure. This module evaluates and exploits the induced abstractions to improve decision quality.

4.1 Rule Abstraction: Top-Down Rule Policy Search via MCTS Refinement

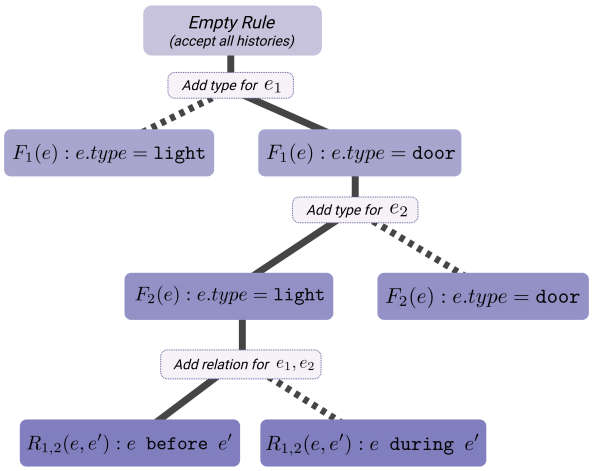

Figure 2: Illustration of the symbolic rule refinement tree of the rule "*The door opened before the light turned on.*"

Expression Tree Each rule T (as defined in Eq. 1) encodes a structured temporal pattern consisting of event-type constraints and pairwise temporal relations. It is built from a grammar defined over two atomic elements: event type filters, which restrict individual event variables to specific semantic categories (e.g., `door_open`, `light_on`), and Allen interval relations, which define pairwise temporal constraints (e.g., `before`, `during`, `overlaps`) between the start and end intervals of events.

The compositionality of the rule grammar allows rules to be constructed incrementally through a top-down search over a refinement tree, similar to approaches in temporal interval pattern mining [24, 23]. The root node corresponds to an unconstrained rule that accepts any history. Each child node applies a refinement operation, incrementally specializing the rule. The refinement process follows a fixed structure to maintain rule consistency: it alternates between introducing a new event variable with a type filter and enumerating its possible temporal relations with all previously defined events. For example, a refinement path for three events may follow the structure: $F_1 \rightarrow F_2 \rightarrow R_{1,2} \rightarrow F_3 \rightarrow R_{1,3} \rightarrow R_{2,3}$. This process systematically explores all valid rule combinations up to a fixed number of event variables. Each node represents a partially specified rule, and leaf nodes correspond to fully defined candidates that cannot be further refined. This is illustrated in Figure 2, where a simple rule is built step by step from two events, a door opening and a light turning on.

Monte Carlo Tree Search The hierarchical and compositional structure of our rule grammar enables MCTS to efficiently explore the symbolic rule space by traversing the refinement tree and selecting promising candidates via the UCT criterion. This balances the exploitation of high-reward refinements with the exploration of less-visited branches. Rules are learned one at a time, meaning that each rule $T_i \in \hat{T}$ is optimized independently through its own MCTS process. Each MCTS instance follows the standard four-phase loop:

- **Selection.** The tree is traversed from the root by recursively selecting grammar refinements according to the UCT formula.
- **Expansion.** When a non-terminal, unexplored node is reached, new child nodes are generated by applying valid symbolic refinement operators at the current level of the grammar.
- **Simulation.** The newly constructed rule is evaluated through interaction with the environment. The agent uses the current rule set, including the candidate, to infer symbolic contexts and train a policy $\pi(a \mid s_t, \hat{c}_t)$. Multiple rollouts are simulated to estimate the quality of this policy based on cumulative reward.
- **Backpropagation.** The obtained reward is propagated up the tree, updating visit counts and value estimates along the path from the expanded leaf to the root.

Reward. To guide symbolic search, we define a rule evaluation procedure based on reward feedback from downstream policy performance, following Hutter's theory on Q-value uniformity [14]. Since the optimal Q-function over histories is generally unavailable, we use empirical return as a surrogate signal. By selecting rule refinements that maximize the cumulative reward of the resulting policy, the search is implicitly guided toward abstractions that preserve value-relevant distinctions across histories.

Let a rule set $\hat{T}$ define a context abstraction function $\hat{c}_t = \mathcal{T}(h_t, \hat{T})$. We optimise $\hat{T}$ based on the expected return of an optimized policy $\pi(a_t \mid s_t, \hat{c}_t)$:

$$\hat{T} = \arg\max_{\hat{T}} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(h_t, s_t, a_t) \;\middle|\; \pi(a_t \mid s_t, \hat{c}_t)\right], \quad \text{where } \hat{c}_t = \mathcal{T}(h_t, \hat{T}).$$

High empirical return under this objective implies that the abstraction has avoided collapsing histories with significantly different Q-values. In this way, reward serves both as a learning signal and an implicit constraint that favors abstractions aligned with theoretical guarantees on decision fidelity.

4.2 Context Induction and Policy Learning

Each time a new rule is evaluated, it is temporarily inserted into the current set of validated rules $\hat{T}$, inducing a new decision process $\mathcal{M}_{\hat{T}}$ over a space of symbolic contexts and observable states. To construct this space, each rule T_i is applied to the current event history h_t, and a check is performed to determine whether the pattern is satisfied. This produces a binary predicate:

$$\mathcal{T}(h_t, T_i) \to c_t \in \{\text{true}, \text{false}\}.$$

Rule matching is implemented through automaton progression: a rule activates if the most recent event e_t completes a valid temporal configuration defined by its grammar. A policy $\pi(a \mid s_t, \hat{c}_t)$ is then trained using any standard reinforcement learning algorithm, treating $\mathcal{M}_{\hat{T}}$ as a surrogate decision process defined over the augmented state space $(s_t, \hat{c}_t)$.

The definition of the surrogate decision process $\mathcal{M}_{\hat{T}}$ directly shapes the relevance of learned abstractions. Since symbolic rules are selected to optimize cumulative reward, the abstraction is implicitly shaped by how the task is framed. In environments with multiple overlapping or unrelated processes, it can be useful to restrict the surrogate task to a subset of decisions. For example, in a smart home where events relate to both cooking and laundry, the process can be redefined to focus solely on cooking actions, such as turning on the stove. Other actions are excluded during training. This targeted setup encourages the discovery of symbolic patterns (e.g., "fridge opened before pan on stove") that are specifically predictive of the selected action. As a result, the learned abstractions become more targeted and interpretable. This flexibility in defining $\mathcal{M}_{\hat{T}}$ enables symbolic contexts to serve as latent states that disambiguate concurrent activities and emphasize task-relevant structure.

4.3 Joint Optimization Strategy

The overall rule learning procedure proceeds iteratively: at each step, a new optimal rule is discovered through MCTS and temporarily added to the rule set $\hat{T}$, which is then used to guide policy evaluation for subsequent rule searches. To jointly optimize the symbolic rule abstractions and the downstream policy, we adopt an alternating procedure that interleaves rule search with policy learning. At each iteration, the system performs the following steps:

1. Use MCTS to explore the space of rule refinements and select a new candidate rule T_i that shows potential for improving decision-making.
2. Temporarily incorporate T_i into the current rule set $\hat{T}$ and train the policy $\pi(a \mid s_t, \hat{c}_t)$ in the symbolic context space induced by $\hat{T} \cup \{T_i\}$.
3. Evaluate the contribution of T_i by computing the expected cumulative reward under the updated policy, thereby scoring the rule based on its utility.
4. If the MCTS search for T_i satisfies the stopping criterion, which requires that rule refinements remain stable for a fixed number of iterations, commit the best candidate to the rule set by adding T_i to $\hat{T}$ and proceed to refine the next rule.

This procedure is repeated until a predefined limit on the number of rules is reached or further improvements become negligible. Rule evaluation and policy training are conducted in an online fashion: new interaction episodes are generated for each candidate rule using the current rule set and policy. While experience is not explicitly stored across iterations, the rule set $\hat{T}$ accumulates over time, progressively enriching the symbolic context space. The interleaving of rule refinement and policy optimization allows the system to incrementally construct abstractions that support symbolic generalization while adapting behavior through reinforcement learning.

5 Experiments

To evaluate the effectiveness of our approach, henceforth referred to as Temporal MCTS (TMCTS), we conduct experiments in the `OpenTheChests` environment [28]. This synthetic benchmark is explicitly constructed to test the ability of agents to detect and act upon temporally extended structures embedded within asynchronous event streams. The environment simulates multiple concurrent activities, each defined by a fixed symbolic pattern comprising event-type filters and temporal constraints grounded in Allen's interval algebra. These patterns control the unlocking of chests: each chest corresponds to an activity that can only be unlocked when its associated temporal rule is fully instantiated by observed events in the stream. The agent receives a continuous stream of timestamped events and also observes the current state of each chest (i.e., whether it is still locked or has already been opened). Its task is to monitor the event stream, infer which activity patterns are currently unfolding, and decide when to attempt opening a chest. A successful match between a completed activity and a chest opening yields a positive reward, whereas premature or incorrect attempts incur penalties. Given the discrete nature of actions and symbolic contexts, policy learning is performed using tabular Q-learning, which is sufficient to capture optimal behavior in this setting.

For our tests, each episode consists of three overlapping activity patterns, each defined by distinct symbolic temporal rules that are designed to challenge the pattern learner. Event sequences are generated with temporal jitter and include background distractor events that do not belong to any pattern, thereby introducing noise and increasing the challenge of accurate recognition. Each activity is grounded in a symbolic rule composed of either two or four temporally related events. We evaluate two experimental configurations:

- **Two-event activity configuration**: In this simpler setting, each activity is defined by a temporal dependency between two events of the same type. While the temporal structure is minimal, repeated use of the same event type across patterns introduces ambiguity and tests the agent's capacity for precise rule disambiguation.
  - *Two events of type **A**, whose **beginning and ending meet**.*
  - *Two events of type **B**, where the **second occurs during the first**.*
  - *Two events of type **C**, where the **first occurs before the second**.*
- **Four-event activity configuration**: This setup introduces more intricate dependencies by requiring recognition of longer event sequences. Activities involve both repeated and mixed event types along with a richer set of temporal constraints, thereby increasing historical complexity and pattern variability. Due to space constraints, we only show one example below, but all four-event activity definitions used in this configuration are provided in the appendix.
  - *A sequence beginning with an event **A**, followed by an event **C** that occurs **after** the first, then a second event **C** that **meets the first and overlaps the second**, and ending with another event **A**, which **follows all three**.*

Our primary objective is to assess both the decision performance and the symbolic fidelity of the learned rule abstractions. Success rate measures the agent's ability to complete the chest-opening task without incurring penalties, serving as a direct indicator of task performance. To evaluate rule precision, we measure the recall rate, defined as the percentage of the initially defined activity rules that are correctly recovered by the learned symbolic rules. This provides a direct estimate of the agent's ability to reconstruct the intended temporal structure. These metrics are tracked during training to analyze convergence behavior and variability in learning dynamics across methods.

We compare our method against two representative baselines: TXCS (Temporal eXtended Classifier System), a symbolic, evolutionary rule-learning framework adapted to handle temporal event streams (see Appendix for details), and DTQN [10], a deep learning model that learns temporal dependencies through attention mechanisms and encodings. All agents, including our TMCTS approach, operate under identical observation and action constraints, receiving the same event streams and chest state feedback. Each experiment is run with five different random seeds, and all reported metrics include 95% confidence intervals to account for stochasticity in both learning dynamics and environment variability.

6 Results and Analysis

During evaluation, we performed targeted hyperparameter tuning to ensure a fair comparison across methods. For DTQN, we adjusted the event window size to optimize temporal encoding capacity. For TXCS, tuning focused on the evolutionary mechanism, including population size, fitness thresholds, and deletion parameters to balance rule diversity and generalization. For TMCTS, we primarily tuned exploration-exploitation trade-offs within the MCTS procedure, as well as convergence criteria governing when to halt the search for new symbolic patterns.

Scenario	Algorithm	SR (%)	RR (%)	CC
Two Events	TMCTS	100	100, 100, 100	0.15
	TXCS	100	100, 66, 66	2
	DTQN	100	NA	0.1
Four Events	TMCTS	100	90, 80, 70	0.9
	TXCS	20	30, 40, 10	10
	DTQN	100	NA	0.8

Table 1: Performance of algorithms: Sucess Rate (SR) in environment, Recall Rate (RR) of rule per activity and Computational Cost (CC) in training hours.

6.1 Low Complexity : Two Events

As shown in Fig. 3, both TMCTS and DTQN successfully solve the two-event task with comparable success rates, while TXCS displays greater variance and instability. The performance fluctuations observed in TXCS are largely due to instability in population pruning and fitness evaluation. However, when extracting its best-performing rules post hoc, we find that they do exhibit strong fidelity to the ground-truth patterns. TMCTS converges rapidly in this setting, as the pattern tree depth required for two-event rules is shallow. Despite some early uncertainty due to exploratory rollouts, the correct patterns are quickly identified and reliably linked to the appropriate actions. The symbolic rules learned by TMCTS were found to exactly match the activity definitions specified in the environment (Table 1). This demonstrates the method's capacity not only for effective decision-making but also for faithful symbolic recovery of underlying temporal structures. DTQN also converges quickly, effectively encoding the short temporal sequences, and benefits from the task's simplicity. While it lacks the explicit interpretability of rule-based methods like TXCS and TMCTS, it maintains stable performance throughout training and evaluation.

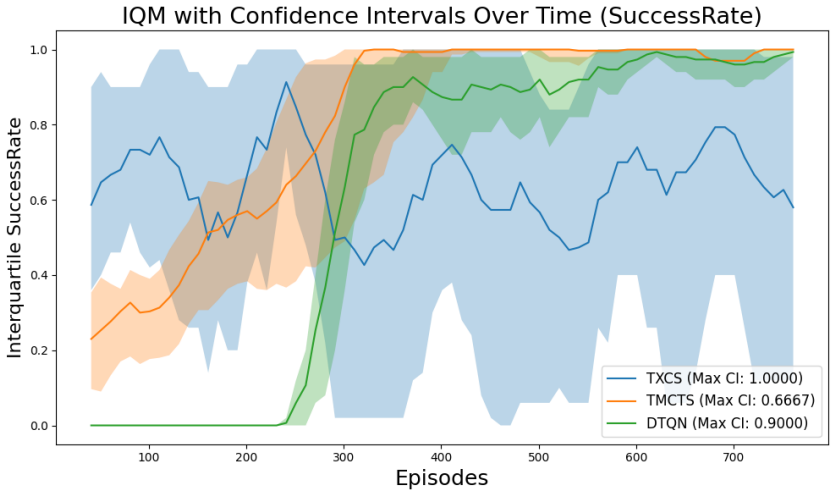

Figure 3: Success rates of DTQN, TXCS and TMCTS measured across 5 random seeds for the Two Events configuration during training.

6.2 Moderate Complexity : Four Events

In the case of higher temporal complexity, we observe similar trends to the low-complexity setting, as shown in Fig. 4. As the activity length increases to four events, TXCS encounters scalability challenges, requiring longer resolution times and increased computational resources (Table 1). The heightened pattern complexity leads to instability in rule deletion dynamics and greater variance across training runs. In contrast, both TMCTS and DTQN converge efficiently, with significantly lower computational overhead. In TMCTS, we observe mild uncertainty and sensitivity to exploration-exploitation parameters, largely due to the larger branching factor in the symbolic rule search space, which makes it harder to consistently prioritize optimal refinements. Additionally, the stochastic nature of MCTS rollouts, coupled with a noisier reward signal, can occasionally promote suboptimal patterns during early stages of search. Despite this, TMCTS recovers rules that align well with the target definitions, though recall rates are lower as shorter rules often suffice to reach optimal performances, reducing the incentive to explore deeper structures. Finally, DTQN handles moderate complexity effectively, showing no notable degradation in performance or training stability. Its success largely depends on tuning the window size to ensure that active activities are captured within the input sequence, allowing the model to track temporal dependencies despite longer histories. However, unlike symbolic approaches such as TMCTS, DTQN does not yield interpretable rules, making its internal representations less transparent.

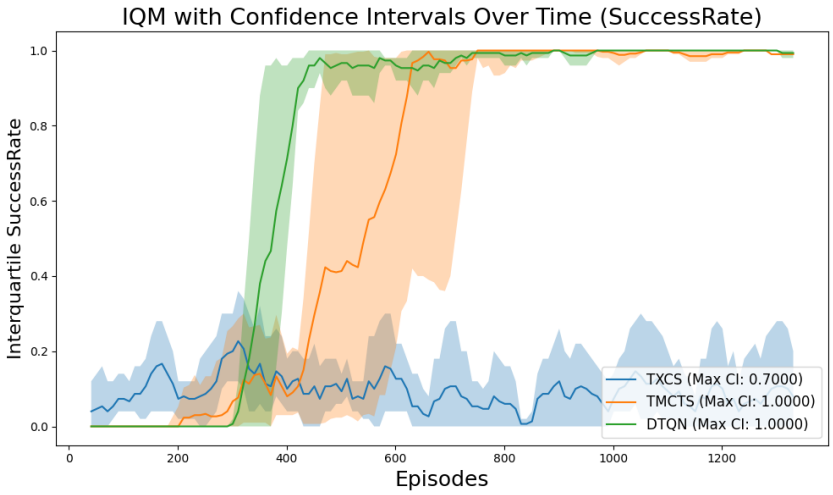

Figure 4: Success rates of DTQN, TXCS and TMCTS measured across 5 random seeds for the Four Events configuration during training.

7 Conclusion and Future Work

Our experiments demonstrate that TMCTS effectively learns interpretable and decision-relevant symbolic rules in event-driven environments. In both low and moderate temporal complexity settings, TMCTS achieves strong task performance while maintaining high rule fidelity, closely matching ground-truth activity definitions. Compared to neural and evolutionary baselines, it offers a favorable balance between interpretability and efficiency, particularly in environments where temporal logic plays a central role. While recall rates remain high, we observe a tendency toward shorter rules that achieve sufficient performance, leading to a reduction in deeper pattern exploration. This highlights both the strength and the current limitations of reward-driven symbolic abstraction. For future work, we aim to scale TMCTS to more complex, multi-agent and noisy environments, and to develop more advanced mechanisms for rule evaluation and exploration-exploitation balancing within the MCTS framework. These improvements will further enhance the robustness and generalization capacity of symbolic learning in non-Markovian decision-making settings.

References

[1] James F Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of logic and computation*, 4(5):531–579, 1994.

[2] Alexander Artikis, Anastasios Skarlatidis, François Portet, and Georgios Paliouras. Logic-based event recognition. *The Knowledge Engineering Review*, 27(4):469–506, 2012.

[3] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1160–1167, 1996.

[4] Damien Bouchabou, Sao Mai Nguyen, Christophe Lohr, Benoit LeDuc, and Ioannis Kanellos. A survey of human activity recognition in smart homes based on iot sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18):6037, 2021.

[5] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[6] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[7] Frada Burstein, Clyde W Holsapple, Alex Bennet, and David Bennet. The decision-making process in a complex situation. *Handbook on decision support systems 1: Basic themes*, pages 3–20, 2008.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: growing generalizable, interpretable knowledge with wake–sleep bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381(2251):20220050, 2023.

[10] Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer q-networks for partially observable reinforcement learning. *arXiv preprint arXiv:2206.01078*, 2022.

[11] Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos Garofalakis. Complex event recognition in the big data era: a survey. *The VLDB Journal*, 29: 313–352, 2020.

[12] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[13] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*, pages 1–22. Springer, 2020.

[14] Marcus Hutter. Extreme state aggregation beyond markov decision processes. *Theoretical Computer Science*, 650:73–91, 2016.

[15] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

[16] Jiří Kubalík, Erik Derner, Jan Žegklitz, and Robert Babuška. Symbolic regression methods for reinforcement learning. *IEEE Access*, 9:139697–139711, 2021.

[17] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason H Moore. Learning concise representations for regression by evolving networks of trees. *arXiv preprint arXiv:1807.00981*, 2018.

[18] Stéphane Lafortune. Discrete event systems: Modeling, observation, and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:141–159, 2019.

[19] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR, 2021.

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

[21] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems: specifications*, volume 1. Springer Science & Business Media, 1992.

[22] Ludovico Mitchener, David Tuckey, Matthew Crosby, and Alessandra Russo. Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework. *Machine Learning*, 111 (4):1523–1549, 2022.

[23] Robert Moskovitch. Mining temporal data. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, pages 469–490, 2023.

[24] Robert Moskovitch and Yuval Shahar. Medical temporal-knowledge discovery via temporal abstraction. In *AMIA annual symposium proceedings*, volume 2009, page 452, 2009.

[25] Richard J Preen, Stewart W Wilson, and Larry Bull. Autoencoding with a classifier system. *IEEE Transactions on Evolutionary Computation*, 25(6):1079–1090, 2021.

[26] Emmanuel Rachelson. *Temporal markov decision problems*. PhD thesis, Citeseer, 2009.

[27] Alessandro Ronca, Gabriel Paludo Licks, and Giuseppe De Giacomo. Markov abstractions for pac reinforcement learning in non-markov decision processes. *arXiv preprint arXiv:2205.01053*, 2022.

[28] Ivelina Stoyanova, Nicolas Museux, Sao Mai Nguyen, and David Filliat. Open the chests: An environment for activity recognition and sequential decision problems using temporal logic. In *31st International Symposium on Temporal Representation and Reasoning (TIME 2024)*, pages 5–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[29] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. *arXiv preprint arXiv:2205.13134*, 2022.

[30] Guy Tennenholtz, Nadav Merlis, Lior Shani, Martin Mladenov, and Craig Boutilier. Reinforcement learning with history dependent dynamic contexts. In *International Conference on Machine Learning*, pages 34011–34053. PMLR, 2023.

[31] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[32] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International conference on machine learning*, pages 5045–5054. PMLR, 2018.

[33] Håkan LS Younes and Reid G Simmons. Solving generalized semi-markov decision processes using continuous phase-type distributions. In *AAAI*, volume 4, page 742, 2004.

[34] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

A Appendix: Description of TXCS Algorithm

Temporal eXtended Classifier System (TXCS) is a hybrid rule-based learning algorithm designed for reactive activity recognition in event-driven environments. It extends the classical eXtended Classifier System (XCS) by incorporating *temporal logic* and *genetic programming* to learn interpretable decision rules that model temporal and historical dependencies.

A.1 Theoretical Foundation

TXCS represents decision-making through a set of classifiers defined as condition–action–reward tuples:

$$cl = (C, A, R, \varepsilon, F)$$

- C: Condition function identifying a temporal activity pattern,
- A: Associated action,
- R: Predicted reward when action A is taken under condition C,
- ε: Prediction error,
- F: Fitness score of the classifier.

The condition C is modeled as a matrix that defines constraints on event types and their pairwise temporal relations using Allen's interval algebra (e.g., *before*, *meets*, *during*). An activity A is recognized if:

$$C_A(e_1, \ldots, e_n) = \bigwedge_{i=1}^{n} F_{T_i}(e_i) \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} R_{i,j}(e_i, e_j)$$

where $F_{T_i}(e_i)$ filters events of type T_i and $R_{i,j}(e_i, e_j)$ checks if the temporal relation between e_i and e_j matches $R_{i,j}$.

A.2 Practical Implementation

1. **Matching:** At each time step t, a history of events h_t is analyzed. All classifiers whose conditions match subsets of h_t form the *match set* M.
2. **Action Selection:** For each candidate action a, the predicted payoff is computed as:

$$P(a) = \frac{\sum\limits_{C_i \in M, A=a} R_i \cdot F_i}{\sum\limits_{C_i \in M, A=a} F_i}$$

   The action a^* with the highest predicted payoff $P(a)$ is executed.
3. **Learning and Evolution:** Classifiers are updated using reinforcement learning. When triggered, a genetic algorithm evolves the classifier population via crossover and mutation of condition matrices. Low-fitness rules are periodically removed.

A.3 Algorithm Pseudocode

Algorithm 1 XCS Algorithm

Initialize population $\mathcal{P}$ of classifiers
for each iteration t **do**
    Observe current history h_t
    Form match set $M = \{C_i \in P \mid C_i \text{ matches } h_t\}$
    Action selection:
    for each action a **do**
        Calculate payoff $P(a)$
    end for
    Select action a^* with the highest payoff $P(a^*)$
    Execute action a^* and observe reward r
    Classifier update:
    for each classifier $C_i \in M$ **do**
        Calculate $\rho = r + \gamma \max_A P^{t+1}(A)$
        Update reward prediction: $R_i \leftarrow R_i + \alpha(\rho - R_i)$
        Update prediction error: $\varepsilon_i \leftarrow \varepsilon_i + \beta(|\rho - R_i| - \varepsilon_i)$
        Update fitness: $F_i \leftarrow F_i + \gamma(\text{accuracy}(C_i) - F_i)$
    end for
    Genetic Algorithm (GA) application:
    if GA is triggered **then**
        Apply crossover and mutation to generate new classifiers
        Replace least fit classifiers in P
    end if
end for