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

Ivelina Stoyanova
U2IS, ENSTA Paris
Institut Polytechnique de Paris
cortAIx LABS THALES
Palaiseau, France
ivelina.stoyanova@thalesgroup.com

Nicolas Museux
cortAIx LABS
THALES
Palaiseau, France
nicolas.museux@thalesgroup.com

Sao Mai Nguyen
U2IS, ENSTA Paris
Institut Polytechnique de Paris
Palaiseau, France
nguyensmai@gmail.com

David Filliat
U2IS, ENSTA Paris
Institut Polytechnique de Paris
Palaiseau, France
david.filliat@ensta-paris.fr

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 driven not by fixed time intervals but by the occurrence of structured sequences of events. This requires agents to recognize and respond to temporal patterns within interaction histories rather than rely on static observations. Such behavior is especially relevant in domains like reactive activity recognition in smart homes or industrial systems with dense sensor networks, where asynchronous event streams reflect underlying physical or behavioral processes [4, 22, 12]. For instance, opening the fridge followed shortly by turning on the oven may indicate the beginning of a cooking activity, despite variability in timing between events. In these settings, individual events are discrete observations, while activities represent temporally extended patterns composed of multiple, interrelated events. Furthermore, there are usually multiple behaviours evolving concurrently, resulting in the generation of exogenous events with irregular and variable durations that challenge traditional, clock-based modeling [5, 32]. As a result, effective decision-making requires identifying multiple interleaved activities [7, 34]. For example, two residents could be arriving home and cooking at the same time, requiring the system to unlock the door and turn on the stove ventilation in response. Systems like Complex Event Processing (CEP) [2] address this by applying symbolic rules over event patterns, similar in spirit to Generalized Semi-Markov Decision Processes (GSMDPs), which model behavior through concurrent temporal dynamics [40].

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-based architectures, models typically lack the capacity for symbolic abstraction and compositional generalization [42, 41], making them ill-equipped for critical environments where interpretability is key. To overcome these limitations, we explore abstracting histories into symbolic structures that capture both temporal and attribute-level dependencies between observations, enabling agents to ground decisions in interpretable, temporally extended contexts. This approach is motivated by prior successes in using temporal logics and automata to specify non-Markovian reward structures [3, 14, 38], as well as by recent advances in symbolic regression and program synthesis, such as Monte Carlo Tree Search (MCTS) and large language models (LLMs), for learning structured policies from data [20, 35, 18]. This line of work is theoretically grounded in the goal of learning representations that generalize across large state spaces or interaction histories [25, 16].

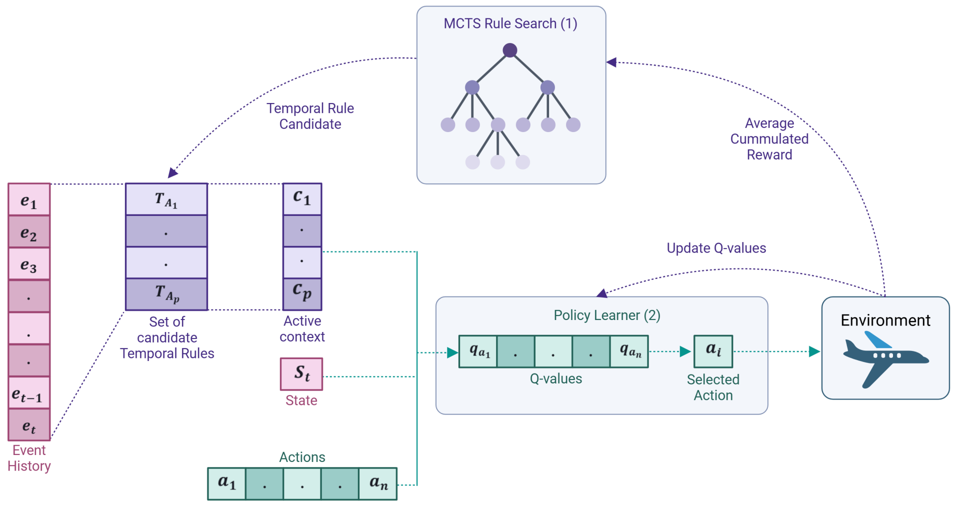


Figure 1: Overview of the proposed two-level architecture. The system receives a stream of temporally-annotated events, which are processed by a symbolic rule abstraction module to produce a latent context. This context, combined with the current observable state, is passed to a policy module that selects 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 model non-Markovian dynamics, where optimal decisions depend not only on the current state but also on past interactions. These include Dynamic Contextual MDPs (DCMDPs) [37], Context MDPs [13], Activity-Based MDPs (ABMDPs) [34], and ϕ-MDPs [16], which modify state representations or transition dynamics to incorporate historical information. While these approaches can model temporal dependencies, they typically rely on latent vector-based summaries of the past and do not associate symbolic meaning with those histories, limiting interpretability and transparency. In parallel, deep learning models such as recurrent networks and Transformers [8] have been used to encode full interaction histories end-to-end. However, their reliance on dense, learned representations constrains generalization across semantically similar histories and offers limited insight into the decision logic. To address both limitations, our model derives symbolic representations that capture high-level temporal and semantic structure in a transparent and interpretable manner.

Representation Learning and Automata-Based Abstractions. Another prominent line of work focuses on learning compact representations of latent states and histories through interaction [25]. Broadly, approaches in this area fall into two categories: those that encode histories into latent spaces using neural networks [24], and those that employ automata or specification-based abstractions to explicitly model temporal structure [41]. The first category aims to improve generalization and sample efficiency by learning low-dimensional embeddings that retain predictive or decision-relevant information. Examples include bisimulation metrics, self-predictive encodings, and compressed belief states. In contrast, automata-based and temporal logic approaches impose symbolic structure on history-dependent behavior, especially in settings with temporally extended rewards or dependencies [3, 38]. For example, automata-augmented POMDPs and symbolic observation detectors have been used to enforce trajectory-level constraints and model high-level task structure [33]. Similarly, Linear Temporal Logic (LTL) and other formal specifications are often used to encode desired behaviors, though typically as predefined constraints rather than learned abstractions [14]. However, these two approaches face complementary limitations: latent-space methods prioritize compressive state representations over structured temporal abstraction, while automata-based models often rely on rigid, hand-crafted rules that are hard to adapt to asynchronous or noisy settings.

Symbolic Reinforcement Learning and Interpretable Policies. Symbolic reinforcement learning aims to produce effective, human-interpretable policies, typically expressed as logic rules, relational policies or algebraic programs. Early work in Relational Reinforcement Learning (RRL) [9, 36] showed that first-order logic could enable generalization across different object configurations, but these approaches were often confined to simple domains and relied heavily on expert-provided abstractions. Evolutionary methods such as XCS and symbolic regression have equally long been used to extract rule-based policies, though they struggle with scalability in high-dimensional or temporally complex environments [21, 31], with more recent applications using Genetic Programming for interpretable RL in classical domains [15]. Hybrid neuro-symbolic approaches increasingly combine deep learning for perception with symbolic reasoning for abstraction [30], often using CNNs or relational networks to map raw visual input into symbolic representations [11, 42]. Others approach the problem by distilling neural policies under the form of decision trees [19]. Logic programming and planning have also been integrated (e.g. Answer Set Programming) for hierarchical policies [27] or symbolic plans as scaffolds for RL [17]. While such methods improve interpretability and transfer, they often remain limited to single-task or single-process settings and face challenges scaling to dynamic, multi-agent or high-dimensional domains.

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 [41]. Techniques using MCTS and guided symbolic regression have been applied to this end, enabling tractable exploration of combinatorial symbolic policy spaces [35]. There have been many recent applications that leverage neural models as guidance signals, such as LLMs for end-to-end symbolic regression [18] or RNNs for the synthesis of symbolic policies in reinforcement learning [23], typically using predictive accuracy or imitation loss as learning criteria. Other methods explore programmatic policy spaces by minimizing the distance to an oracle policy distilled from a neural network [39]. While these methods provide interesting approaches to blending symbolic and subsymbolic reasoning, they have yet to be implemented in asynchronous, multi-process decision-making settings where temporal dependencies are central. Instead, most existing work in this area has concentrated on symbolic-physics discovery, equation regression, or state-based control tasks.

3 Background

This work builds on the idea of optimizing decisions over a reduced space of interaction histories by leveraging symbolic temporal rules for activity recognition. We rely on abstraction-based reinforcement learning frameworks to handle non-Markovian environments, where current observations alone are insufficient for optimal decision-making. The following formalism captures event histories, temporal relations, and symbolic rules that structure the agent's decision process.

Event Histories. In event-driven environments, agents receive input not as fixed state vectors but as streams of temporally structured events. These events encode discrete observations triggered by external or internal processes over time. We formalize the interaction history up to time t as a sequence $h_t = \{e_1, \ldots, e_t\}$, where each event e_i is a tuple

$$e_i = (\tau_i, [t_i^{\text{start}}, t_i^{\text{end}}])$$

consisting of an event type $\tau_i \in \mathcal{T}$ (e.g., `door_open`, `signal_high`) drawn from a finite set of event types $\mathcal{T}$, and a temporal interval $[t_i^{\text{start}}, t_i^{\text{end}}]$ indicating when the event occurs.

Interval-Based Reasoning. While multiple temporal logics exist for modeling time-dependent phenomena [26], Allen's Interval Algebra [1] is particularly well-suited for event-driven settings that involve reasoning over time intervals. It defines 13 primitive relations (such as `before`, `overlaps`, `during`, and `meets`) that express how two events are temporally related. In our context, each event e_i is associated with a time interval $[t_i^{\text{start}}, t_i^{\text{end}}]$, 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:

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

Activity Recognition Rules. To represent structured activity patterns within event histories, we define symbolic rules that specify both semantic and temporal relationships among a set of events. Each rule T is a logical formula composed of type filters and temporal constraints, evaluated over n event variables. Formally, a rule is defined as:

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

Here, $F_i(e_i)$ is a unary predicate that checks whether the i-th event matches a specified event type. A rule T is said to match a history h_t if there exists a tuple of events from h_t that satisfies both the type and temporal constraints.

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 [16]) 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. We adopt the Activity-Based Markov Decision Process (ABMDP) framework [34] to model symbolic structure in event-driven environments. An ABMDP is defined as:

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

where $\mathcal{S}$ is the observable state space, $\mathcal{C}$ is a set of latent contexts, $\mathcal{A}$ is the set of actions, P is the transition function, R is the reward function, and T is a set of symbolic temporal rules. A latent context $c^* \in \mathcal{C}$ is inferred from the agent's interaction history h using a context extraction function $\mathcal{T}(h, T)$, which evaluates the rule set T over h to produce a symbolic abstraction. Planning is then performed in the augmented state space (s, c^*), using policies of the form $\pi(a \mid s, c^*)$. This formulation aligns with the ϕMDP model [16], where the mapping $\mathcal{T}(h, T)$ plays the role of the abstraction function $\phi(h)$ that maps non-Markovian histories to a compressed, decision-relevant representation. As a result, the ABMDP induces a surrogate Markovian decision process over abstract states.

Q-Value Uniformity and Approximation Guarantees. We begin by formalizing the abstraction property required for bounded approximation in abstract decision processes.

Definition 3.1 (Q-Value Uniformity). *Let ϕ be an abstraction mapping from histories to abstract states. We say that ϕ satisfies* Q-value uniformity *with bound $\varepsilon > 0$ if for all histories $h, h' \in \mathcal{H}$ such that $\phi(h) = \phi(h')$, and for all actions $a \in \mathcal{A}$,*

$$|Q^*(h,a) - Q^*(h',a)| < \varepsilon.$$

Under this condition, the following result holds [16, Theorem 8]:

Theorem 3.1 (Approximation Guarantee). *Let ϕ be an abstraction that satisfies Q-value uniformity with bound ε. Let $\pi(a \mid \phi(h))$ be a policy defined over the abstract process, and let $\Pi(h) = \pi(\phi(h))$ be its lifted version over histories. Then, for all $h \in \mathcal{H}$ and $a \in \mathcal{A}$,*

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

and in particular,

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

This result ensures that planning in the abstract MDP yields near-optimal behavior in the original, potentially non-Markovian environment, provided that the abstraction does not collapse value-distinct histories.

MCTS for Rule Refinement. MCTS is used to explore the symbolic rule space via simulated rollouts, treating rule search as a program synthesis problem. It selects candidate refinements using the UCT criterion [6]:

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

where $Q(n_i)$ is the estimated node value, $N(n)$ and $N(n_i)$ are visit counts, and c controls the exploration–exploitation trade-off.

4 Method

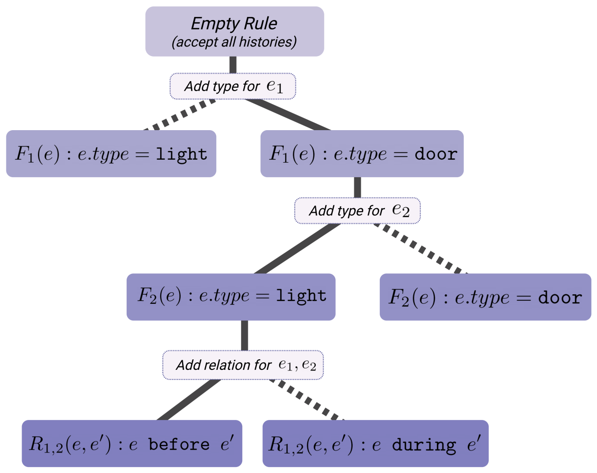
We propose a two-level architecture that integrates symbolic temporal abstraction search with reinforcement learning to enable interpretable and reactive behavior in event-driven environments. The system is composed of two interacting modules: (1) a symbolic rule learning and context induction module, 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. Each rule is discovered independently via Monte Carlo Tree Search (MCTS) over a compositional grammar, where candidate refinements are evaluated based on their downstream impact on reward. This treats rule discovery as a symbolic program synthesis task grounded in reinforcement feedback.
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 of the observed event history h_t. The policy is updated iteratively as new rules are incorporated, enabling the agent to improve decision quality by leveraging increasingly informative abstractions.

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

Expression Tree Each rule T (as defined in Eq. 1) is constructed from a compositional grammar over two atomic elements: event type filters, which constrain 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 times of events.

The grammar induces a top-down refinement tree that systematically explores the rule space, similar to methods in temporal interval pattern mining [29, 28]. The root node represents an unconstrained rule that matches any history, and each child applies a grammar-driven refinement that incrementally adds semantic or temporal structure. To ensure rule consistency, the process follows a fixed alternation:



Empty Rule
(accept all histories)
Add type for e_1
$F_1(e): e.type = \texttt{light}$
$F_1(e): e.type = \texttt{door}$
Add type for e_2
$F_2(e): e.type = \texttt{light}$
$F_2(e): e.type = \texttt{door}$
Add relation for e_1, e_2
$R_{1,2}(e, e'): e\ \texttt{before}\ e'$
$R_{1,2}(e, e'): e\ \texttt{during}\ e'$

Figure 2: Illustration of the symbolic rule refinement tree of the rule "*The door opened before the light turned on.* Solid lines indicate the selected refinement path; dotted lines show alternative branches."

introducing a new event variable via a type filter, then enumerating all temporal relations with existing events. For instance, a refinement path for three events may follow the structure $F_1 \to F_2 \to R_{1,2} \to F_3 \to R_{1,3} \to R_{2,3}$, where the first two event types are selected (F_1, F_2), followed by their temporal relation ($R_{1,2}$); then a third event type is introduced (F_3), along with its relations to both previous events ($R_{1,3}$ and $R_{2,3}$). This process systematically explores all valid rule combinations up to a predetermined maximum number of event variables. Each node in the tree corresponds to a partially specified rule with a subset of constraints, and leaf nodes represent fully defined candidates. Figure 2 illustrates this process for a simple rule involving 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 using the UCT criterion, which 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 $\hat{T}_i \in \hat{T}$ is optimized independently through its own MCTS instance. Each search follows the standard four-phase loop:

- **Selection.** The refinement tree is traversed from the root by recursively selecting child nodes according to the UCT formula, which favors nodes with high estimated value and low visit count.
- **Expansion.** When a non-terminal and unexplored node is reached, new children are generated by applying valid grammar-driven refinements, such as introducing new event variables or adding temporal constraints.
- **Simulation.** The candidate rule is inserted into the current rule set and evaluated by interacting with the environment. The agent uses the updated rule set to infer symbolic contexts and trains a policy $\pi(a \mid s_t, \hat{c}_t)$. Multiple rollouts are conducted to estimate cumulative reward, which serves as the fitness signal for the candidate rule.
- **Backpropagation.** The obtained reward is propagated up the tree, updating value estimates and visit counts 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 [16]. 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 return of the policy trained on the context-augmented state space, 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 optimize $\hat{T}$ based on the expected return of a policy $\pi(a_t \mid s_t, \hat{c}_t)$ trained under the current abstraction:

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

A high empirical return under this objective suggests that the abstraction successfully avoids collapsing histories with significantly different Q-values. In this way, the reward signal serves both as a learning objective and as an implicit constraint that encourages 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 $T_i \in \hat{T}$ is applied to the current event history h_t, and a check is performed to determine whether its pattern is satisfied. This yields a binary predicate:

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

The full symbolic context is then represented as $\hat{c}_t = [c_t^{(1)}, \ldots, c_t^{(k)}]$ for a rule set of size k. Rule matching is implemented via 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 structure of $\mathcal{M}_{\hat{T}}$ directly determines the relevance of the learned abstractions. Since rules are selected to maximize cumulative reward, the abstraction is implicitly shaped by the framing of the task. In environments with multiple overlapping or unrelated processes, it can be beneficial to restrict the surrogate task to a subset of actions. For example, in a smart home where events relate to both cooking and laundry, the process can be redefined to focus solely on cooking-related actions, such as turning on the stove, while ignoring unrelated decisions 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 behavior. As a result, the learned abstractions become more task-relevant and interpretable. This flexibility in defining $\mathcal{M}_{\hat{T}}$ enables symbolic contexts to serve as latent states that disambiguate concurrent activities and highlight decision-critical 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 to downstream behavior.
4. If the MCTS search for T_i satisfies the stopping criterion (e.g., the rule structure stabilizes over a fixed number of rollouts or performance improvements plateau) then 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 in policy performance fall below a fixed threshold. Specifically, an improvement is considered negligible if the increase in cumulative reward over a sliding evaluation window is less than a small constant δ (e.g., $\delta = 0.01$), indicating convergence. 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 continuously adapting behavior through reinforcement learning. This joint strategy ensures that rule discovery is grounded in policy impact, aligning symbolic abstraction with long-term reward optimization.

5 Experiments

To evaluate the effectiveness of our approach, henceforth referred to as Temporal MCTS (TMCTS), we conduct experiments in the `OpenTheChests` environment [34]. This synthetic benchmark is explicitly designed 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 composed of event-type filters and temporal constraints grounded in Allen's interval algebra. These patterns govern 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, along with 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 action 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 a distinct symbolic temporal rule designed to challenge the pattern learner. Event sequences are generated with temporal jitter and include background distractor events that do not belong to any activity, introducing noise and increasing the difficulty 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, reuse of event types across multiple patterns introduces ambiguity and tests the agent's ability to disambiguate rules based on timing alone.
  - *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, combined with a richer set of temporal constraints, thereby increasing historical complexity and pattern variability. An example configuration can be:
  - *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 (SR) measures the agent's ability to complete the chest-opening task without incurring penalties, serving as a direct indicator of task performance. To evaluate symbolic quality, we compute the Rule Recall (RR), defined as the percentage of ground-truth activity rules that are correctly recovered by the learned symbolic rules. This provides an estimate of the agent's ability to reconstruct the intended temporal structure underlying the environment. Both metrics are tracked throughout training to analyze convergence behavior, stability, and variability in learning dynamics across different methods.

We compare our method against two representative baselines: TXCS (Temporal eXtended Classifier System), a symbolic, evolutionary rule-learning framework adapted for temporal event streams (see Appendix for implementation details), and DTQN [10], a deep neural model that captures temporal dependencies through attention-based sequence encoding. 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 repeated with five random seeds, and all reported metrics include 95% confidence intervals to account for both learning stochasticity and environment variability.

6 Results and Analysis

To ensure a fair comparison, we performed targeted hyperparameter tuning for each method during evaluation. For DTQN, we adjusted the event window size and attention parameters to optimize temporal encoding capacity. For TXCS, tuning focused on the evolutionary strategy, including population size, fitness thresholds, and deletion criteria, to balance rule diversity and generalization. For TMCTS, we primarily tuned the exploration-exploitation trade-off in the MCTS procedure, as well as convergence criteria that determine when to halt the search for new symbolic patterns.

Scenario	Algorithm	SR (%)	RR (%)	CC (hrs)
Two Events	TMCTS	100	100, 100, 100	0.15 ± 0.1
	TXCS	100	100, 66, 66	2.00 ± 0.35
	DTQN	100	NA	0.10 ± 0.05
Four Events	TMCTS	100	90, 80, 70	0.90 ± 0.25
	TXCS	20	30, 40, 10	10.00 ± 0.5
	DTQN	100	NA	0.80 ± 0.1

Table 1: Performance of algorithms: Success Rate (SR) in environment, Recall Rate (RR) of rule per activity, and Computational Cost (CC) in training hours with 95% confidence intervals.

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. These fluctuations are largely attributable to the evolutionary dynamics of TXCS, particularly instability in population pruning and fitness evaluation. However, a post hoc analysis of its best-performing rules reveals strong fidelity to the ground-truth patterns. TMCTS converges rapidly in this setting, as the required pattern tree depth for two-event rules is shallow. Despite some early uncertainty due to exploratory rollouts, the correct patterns are quickly identified and consistently linked to appropriate actions. The symbolic rules discovered by TMCTS exactly match the activity definitions used to generate the environment (see Table 1), demonstrating the method's capacity for both effective decision-making and faithful symbolic recovery of temporal structure. DTQN also converges quickly in this scenario, effectively capturing short temporal dependencies through its attention-based encoding. While it lacks the explicit interpretability of symbolic methods like TMCTS and TXCS, it maintains stable performance throughout both training and evaluation.

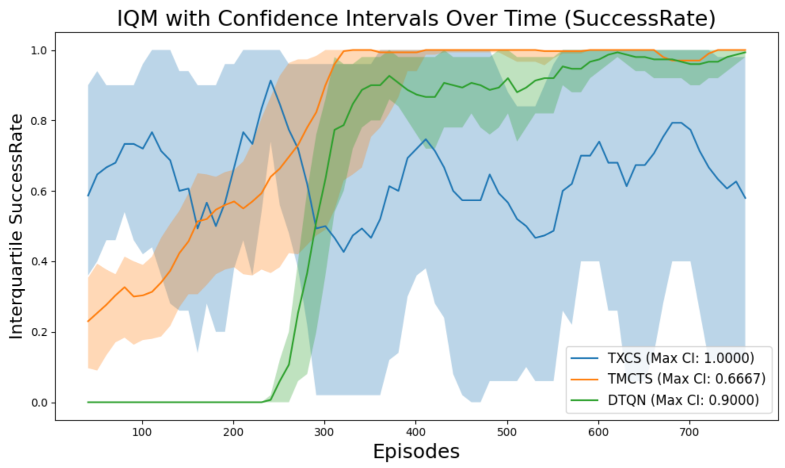


Figure 3: Success rates of DTQN, TXCS and TMCTS measured across 5 random seeds for the Two Events configuration during training.

6.2 Moderate Complexity : Four Events

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

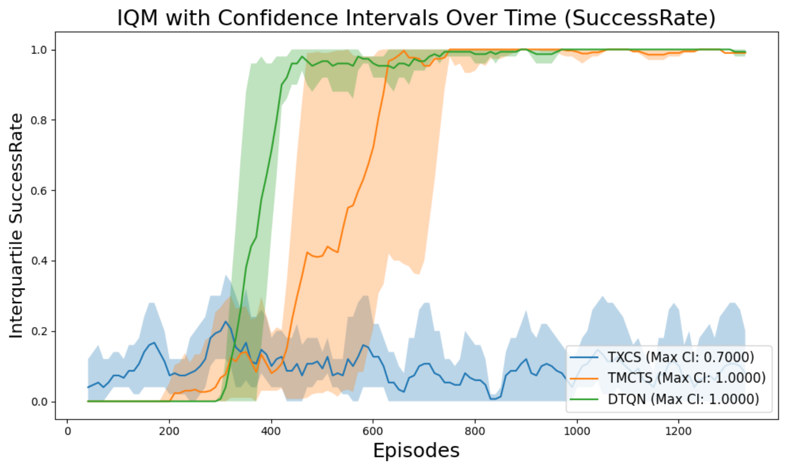


Figure 4: Success rates of DTQN, TXCS and TMCTS measured across 5 random seeds for the Four Events configuration during training.

6.3 Interpretability and Scalability

A key advantage of TMCTS lies in its interpretability: the learned symbolic rules are compositional and compact, enabling explicit characterization of temporally extended activities. This allows each decision to be traced back to a small set of active patterns, providing insight into both the environment's structure and the policy's behavior. In contrast to black-box models like DTQN, TMCTS maps event histories to human-readable abstractions, making it especially suitable for domains where transparency is critical.

We anticipate that TMCTS remains practical in more complex environments. Many real-world activities, though long, are triggered by a few key temporal markers, which our rule grammar can represent compactly. Moreover, MCTS supports efficient exploration of large rule spaces by focusing search on promising refinements. Scalability challenges may emerge when useful rules require deep compositions or when relevant patterns are rare. In such cases, adaptations to MCTS may be needed, such as using oracles to guide policy rollouts, adopting risk-seeking strategies that favor high-return refinements, or biasing expansions toward more informative symbolic structures. Despite these challenges, our results suggest that TMCTS provides a robust and interpretable framework for temporal abstraction, with promising scalability as complexity increases.

7 Conclusion and Future Work

Our experiments show that TMCTS effectively learns interpretable, decision-relevant symbolic rules in event-driven environments. In both low and moderate temporal complexity settings, it achieves strong task performance while closely recovering ground-truth activity definitions. Compared to neural and evolutionary baselines, TMCTS offers a favorable trade-off between interpretability and efficiency, especially in domains with rich temporal structure. While recall remains high, we observe a bias toward shorter rules that suffice for performance, limiting deeper pattern discovery. This highlights both the strengths and limitations of reward-driven symbolic abstraction. In future work, we aim to scale TMCTS to more complex, multi-agent, and noisy environments, and to develop enhanced mechanisms for rule evaluation and exploration–exploitation balancing. These improvements will support broader generalization and robustness in non-Markovian decision-making tasks.

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] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1):7–52, 2001.

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

[11] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*, 2016.

[12] 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.

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

[14] 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.

[15] Daniel Hein, Steffen Udluft, and Thomas A Runkler. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence*, 76: 158–169, 2018.

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

[17] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 540–550, 2020.

[18] 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.

[19] Hector Kohler, Quentin Delfosse, Waris Radji, Riad Akrour, and Philippe Preux. Evaluating interpretable reinforcement learning by distilling policies into programs. *arXiv preprint arXiv:2503.08322*, 2025.

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

[21] 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.

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

[23] 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.

[24] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

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

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

[27] 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.

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

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

[30] Carlos Núñez-Molina, Pablo Mesejo, and Juan Fernández-Olivares. A review of symbolic, subsymbolic and hybrid methods for sequential decision making. *ACM Computing Surveys*, 56 (11):1–36, 2024.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] Prasad Tadepalli, Robert Givan, Kurt Driessens, P Tadepalli, and R Givan. Relational reinforcement learning: An overview. In *Proceedings of the ICML-2004 workshop on relational reinforcement learning*, pages 1–9, 2004.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] Chao Yu, Xuejing Zheng, Hankz Hankui Zhuo, Hai Wan, and Weilin Luo. Reinforcement learning with knowledge representation and reasoning: A brief survey. *arXiv preprint arXiv:2304.12090*, 2023.

[42] 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:

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

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

```
1: Initialize population $\mathcal{P}$ of classifiers
2: for each iteration $t$ do
3:     Observe current history $h_t$
4:     Form match set $M = \{C_i \in P \mid C_i$ matches $h_t\}$
5:     Action selection:
6:     for each action $a$ do
7:         Calculate payoff $P(a)$
8:     end for
9:     Select action $a^*$ with the highest payoff $P(a^*)$
10:    Execute action $a^*$ and observe reward $r$
11:    Classifier update:
12:    for each classifier $C_i \in M$ do
13:        Calculate $\rho = r + \gamma \max_A P^{t+1}(A)$
14:        Update reward prediction: $R_i \leftarrow R_i + \alpha(\rho - R_i)$
15:        Update prediction error: $\varepsilon_i \leftarrow \varepsilon_i + \beta(|\rho - R_i| - \varepsilon_i)$
16:        Update fitness: $F_i \leftarrow F_i + \gamma(\text{accuracy}(C_i) - F_i)$
17:    end for
18:    Genetic Algorithm (GA) application:
19:    if GA is triggered then
20:        Apply crossover and mutation to generate new classifiers
21:        Replace least fit classifiers in $P$
22:    end if
23: end for
```

B Pseudo-code for method

Algorithm 2 BuildRefinementTree(n_{max})

```
1: Initialize root node $r \leftarrow$ empty rule (matches any history)
2: Create a queue $Q \leftarrow \{r\}$
3: Initialize tree $T \leftarrow \{r\}$
4: while $Q$ is not empty do
5:     Pop node $node$ from $Q$
6:     Let $k$ be the number of event variables in $node$
7:     if $k < n_{\text{max}}$ then
8:         for all event types $\tau \in \mathcal{T}$ do
9:             Create new event variable $e_{k+1}$ with type filter $F_{k+1}(e) := (\tau)$
10:            $child \leftarrow node$ extended with $F_{k+1}$
11:            for all $i = 1$ to $k$ do
12:                for all temporal relations $r \in \mathcal{R}_{\text{Allen}}$ do
13:                    Add relation $R_{i,k+1}(e_i, e_{k+1}) := r$ to $child$
14:                end for
15:            end for
16:            Add $child$ to $T$ and $Q$
17:        end for
18:    end if
19: end while
20: return $T$
```

Algorithm 3 Joint Rule Abstraction and Policy Learning

```
1: Initialize empty rule set $\hat{T} \leftarrow \emptyset$
2: Initialize policy $\pi(a \mid s, \hat{c})$ (e.g., random or pretrained)
3: for $i = 1$ to MaxRules do
4:     Initialize MCTS for rule candidate $T_i$
5:     while not converged do
6:         $T_i \leftarrow$ MCTS-ROLLOUT($\hat{T}, \pi$)
7:         $\hat{T}' \leftarrow \hat{T} \cup \{T_i\}$
8:         Train policy $\pi'(a \mid s, \hat{c})$ using context $\hat{c}_t = \mathcal{T}(h_t, \hat{T}')$
9:         $J \leftarrow$ Estimate expected return of $\pi'$ (e.g., via rollouts)
10:        BACKPROPAGATE($T_i$, $J$) in MCTS tree
11:    end while
12:    Commit best $T_i$ from MCTS to rule set: $\hat{T} \leftarrow \hat{T} \cup \{T_i\}$
13:    Update policy $\pi \leftarrow \pi'$ using final $\hat{T}$
14: end for
15: return Final rule set $\hat{T}$ and policy $\pi$
```

C Metric Definitions

We define two metrics to evaluate performance:

- **Success Rate (SR)** is the percentage of evaluation episodes where the agent achieves the maximum possible reward. That is, all relevant chests are correctly opened and no penalties are incurred:

$$\text{SR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[R_i = R_{\text{max}}]$$

  where R_i is the total reward received in episode i, and R_{max} is the known optimal reward for that episode.

- **Rule Recall (RR)** quantifies symbolic fidelity by comparing the recovered symbolic rules to the ground-truth rule structures used to generate the environment. It is defined as the percentage of correctly recovered rule components (type filters and temporal constraints):

  $$\text{RR} = \frac{\text{\# correctly recovered rule terms}}{\text{\# total rule terms}} \times 100$$

  For example, if a ground-truth rule consists of 10 symbolic terms and the learner recovers 8 correctly, the RR is 80%.