Open the Chests: An environment for Activity Recognition and Sequential Decision Problems Using Temporal Logic

Ivelina Stoyanova
U2IS, ENSTA Paris, Institut Polytechnique de Paris, France
THALES, France

Nicolas Museux
THALES, France

Sao Mai Nguyen
U2IS, ENSTA Paris, Institut Polytechnique de Paris, France

David Filliat
U2IS, ENSTA Paris, Institut Polytechnique de Paris, France

Abstract

This article presents `Open the Chests`, a novel benchmark environment designed for simulating and testing activity recognition and reactive decision-making algorithms. By leveraging temporal logic, `Open the Chests` offers a dynamic, event-driven simulation platform that illustrates the complexities of real-world systems. The environment contains multiple chests, each representing an activity pattern that an interacting agent must identify and respond to by pressing a corresponding button. The agent must analyze sequences of asynchronous events generated by the environment to recognize these patterns and make informed decisions. With the aim of theoretically grounding the environment, the Activity-Based Markov Decision Process (AB-MDP) is defined, allowing to model the context-dependent interaction with activities. Our goal is to propose a robust tool for the development, testing, and bench-marking of algorithms that is illustrative of realistic scenarios and allows for the isolation of specific complexities in event-driven environments.

2012 ACM Subject Classification Computing methodologies → Simulation environments

Keywords and phrases Event-Based Decision Making, Activity Recognition, Temporal Logic, Reinforcement Learning, Dynamic Systems, Complex Event Processing, Benchmark Environment, Real-Time Simulation

Supplementary Material *Software (Source Code)*: `Open The Chests`

1 Introduction

The emergence of smart technologies and automated information processing has generated an increasing interest in the fields of activity recognition [17, 8] and sequential event-based decision making [52, 12]. These fields are characterised by the identification and interpretation of behaviours as they occur and the optimisation of subsequent choices of reaction. They allow multiple applications ranging from monitoring and assisting in smart environments to enhance user convenience and safety [43, 13, 53], to automating cyber-security protocols for real time threat detection and response [15, 22, 18], and applying control in industrial settings to improve operational efficiency. However, the development of robust and reliable models for these domains has been challenged by the inherent complexity of their associated environments and the limited availability of suitable test-beds for evaluation [9, 33, 47].

Environments in these fields are typically characterised by multiple interconnected activities evolving concurrently over time, observable indirectly through sensors and detection mechanisms. They often exhibit complex dynamics [44, 25], where activities are defined as

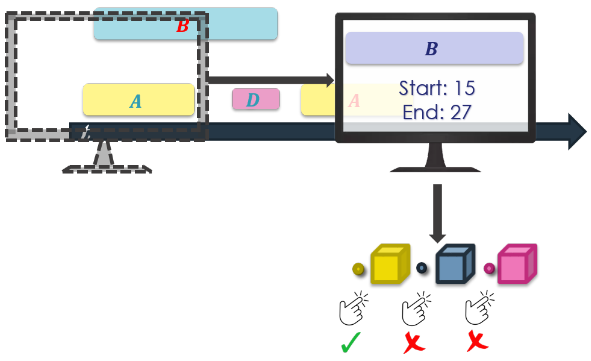


Figure 1 Overview of the `Open The Chests` environment. The figure shows *the event stream* being generated, the *observation of events* on the screen, and the corresponding *interactions with chests* based on significant sequence detection.

significant patterns of observations characterized by complex temporal relationships and inter-dependencies. In the case of the event-driven paradigm [11, 9], observations are processed in order to extract significant changes in the environment under the form of data instances, also referred to as events. These events arrive asynchronously and provide temporal and attribute information on the occurred changes, constituting an event stream. The goal of the system interacting with the environment is to identify activities by analysing event signatures and selecting adapted reactions.

A key challenge for solutions and simulations lies in capturing the contextual and history-dependent nature of behaviors present in the event stream [5, 37, 42]. The significance and interpretation of an event are highly dependent on its surrounding context and the sequence of prior events. Additional challenges include heterogeneous data sources, complex temporal inter-dependencies [40], and uncertainty [2], which make it difficult to develop robust and adaptable systems. These challenges also complicate the process of obtaining large-scale datasets and conducting controlled simulations, thereby hindering the advancement of solutions [25, 20, 55]. Collecting and annotating datasets is expensive, time-consuming, and often impractical due to the vast number of scenario configurations and complexity levels required. Capturing the full range of variability, uncertainty, and asynchrony in event-driven environments often necessitates multiple datasets, further complicating the process. On the other hand, simulators must balance application specificity and simplicity. While a highly detailed, application-specific simulator might closely mimic real-world conditions, it can also make it difficult to isolate and evaluate specific complexities objectively. Conversely, overly simplified simulations may miss critical nuances, leading to gaps in testing.

To address this issue, we introduce `Open The Chests` [1], illustrated in Figure 1, a novel reinforcement learning environment that simulates a gamified scenario of activity recognition and sequential decision-making. Modeled after popular Reinforcement learning environments [51], the environment is designed to represent the complexities of real-world scenarios, where the significance of observations and appropriate reactions are highly dependent on temporal context and the history of events. Its configurable nature allows for multiple levels of complexity, enabling users to isolate specific challenges and systematically evaluate various aspects of decision-making processes. This article details the core complexities of the environment, formalizes the main elements of the problem, and introduces `Open The`

[1] `https://github.com/ThalesGroup/open-the-chests`

`Chests` as a versatile tool for such evaluations.

2 Background and Related Work

Multiple domains developing solutions to the challenges of reactive decision-making could benefit from the usage of standardized benchmarks, particularly in event-driven environments. Complex Event Processing (CEP) and Complex Event Recognition (CER) systems [36, 14, 24] are methods designed to identify and recognize intricate patterns within streams of events using predefined rules. Examining complexities such as uncertainty, noise, and varying context lengths could be crucial in assessing the robustness and reliability of these systems in diverse scenarios. Similarly, when extracting rules and patterns for these domains [19, 32], exploring how the accuracy and interpretability of results vary with scenario complexity could provide valuable insights into the adaptability of these approaches. For the temporal interval pattern mining community [38, 41, 6], comparisons across algorithms could be enriched by considering their computational efficiency and limits, especially in the context of standardized benchmarks. Other approaches to monitoring activities involve using probabilistic methods, often combined with logic-driven techniques [26, 2]. These methods typically estimate the likelihood of events, and exploring their adaptability in less predictable contexts can provide insights into their ability to maintain accuracy under varying conditions. Finally, Reinforcement Learning [48], which is showing growing interest in real-world scenarios [20], would gain from frameworks that allow thorough testing of long-term, contextual dependencies and delayed rewards.

Current benchmarks and datasets for event-driven environments typically fall into two distinct categories, focusing either on the classification of activities [54] or sequential decision-making [51]. Classification methods, which involve identifying and categorizing specific patterns of behavior or events, are often driven by deep learning techniques or knowledge-based methods [12, 8]. Several datasets are available for these purposes [8, 4], offering a range of scenarios for recognition. However, while effective in controlled, static settings, these classification methods often struggle to adapt to the dynamic nature of realistic environments, as they typically do not account for the impact of interaction. Conversely, sequential decision-making frameworks, such as those in Reinforcement Learning, focus on optimizing decisions based on the system's current state, making them well-suited for dynamic environments. Despite their strengths, these methods often rely on simplified environmental models, which can limit their real-world applicability [9]. Specifically, many existing simulators fail to account for the complexities of history dependence, context dependence, and inter-dependencies within the environment. Active research is working to address these challenges by incorporating more complex environmental features into these frameworks [7, 34]. These benchmarks challenge agents with higher-level tasks, requiring memory and goal abstraction. However, to our knowledge, none of them specifically address the task of activity recognition, motivating our development of the `Open The Chests` environment.

3 Formalising Activities using Temporal Logic and Attribute Filters

The development of robust algorithms and reliable simulations for event-driven environments requires the precise formalization of events, activities, and interactions. This formalization provides a foundation for understanding system dynamics and guiding the construction of the environment. In particular, *activities* are critical constructs in reactive decision-making systems, serving as the basis for identifying relevant scenarios and triggering appropriate responses. By integrating both temporal and attribute constraints on the event stream, activities enable the precise encoding of complex semantics as high-level abstractions, capturing meaningful patterns of behavior over time.

3.1 Definition of Events

Events are the fundamental building blocks of the environment, representing significant changes that occur over time. Formally, an event e can be represented as a tuple:

$$e = (sym, Attr, t_{start}, t_{end},)$$

where:

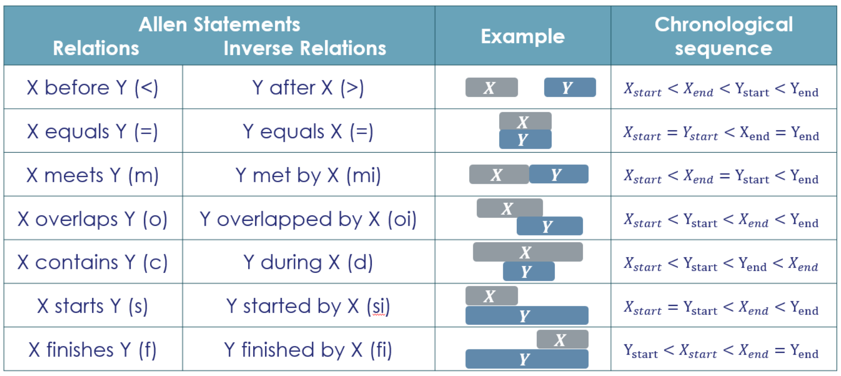
- sym is a symbolic identifier that categorizes the type of event, facilitating its recognition and interpretation within the system.
- $Attr = \{attr_1, \ldots attr_n\}$ represent a set of attributes that provide additional information about the event. These attributes can include, but are not limited to, spatial coordinates, intensity levels, source identifiers, and other domain-specific parameters.
- t_{start} and t_{end} denote the timestamps marking the initiation and conclusion of the event, respectively. These temporal markers are crucial for understanding the duration and sequence of events.

Events arrive asynchronously and are processed to form a history, or event stream. It is denoted as h_t, where t represents the current discrete time step, or the number of the last received event:

$$h_t = \langle e_1, e_2, \ldots, e_t \rangle$$

The temporal aspect of events allows for identifying relationships between them, enabling the construction of higher-level abstractions. Attributes enrich event representation by encoding domain-specific information, facilitating more sophisticated analysis and reasoning about the state of the environment.

3.2 Temporal Relations between Events



Allen Statements: Relations	Allen Statements: Inverse Relations	Example	Chronological sequence
X before Y (<)	Y after X (>)	X Y	$X_{start} < X_{end} < Y_{start} < Y_{end}$
X equals Y (=)	Y equals X (=)	X Y	$X_{start} = Y_{start} < X_{end} = Y_{end}$
X meets Y (m)	Y met by X (mi)	X Y	$X_{start} < X_{end} = Y_{start} < Y_{end}$
X overlaps Y (o)	Y overlapped by X (oi)	X Y	$X_{start} < Y_{start} < X_{end} < Y_{end}$
X contains Y (c)	Y during X (d)	X Y	$X_{start} < Y_{start} < Y_{end} < X_{end}$
X starts Y (s)	Y started by X (si)	X Y	$X_{start} = Y_{start} < X_{end} < Y_{end}$
X finishes Y (f)	Y finished by X (fi)	X Y	$Y_{start} < X_{start} < X_{end} = Y_{end}$

Figure 2 Allen's 13 temporal interval relations: *before, after, meets, met-by, overlaps, overlapped-by, starts, started-by, during, contains, finishes, finished-by, equal.*

Encoding temporal relationships between events is a key aspect of an activity's representation, capturing their relative order, duration, and overlaps. Various formalisms have been proposed for representing temporality, each with its own trade-offs in expressiveness and computational complexity [16, 27]. For the `Open The Chests` environment, we leverage Allen's Interval Algebra [3] due to its expressivity in capturing temporal relations and its established use in the activity recognition community [13, 39]. This algebra defines a set of thirteen mutually exhaustive temporal relationships between two time intervals. As depicted in Figure 2, each relation imposes a unique constraint on the start and end times of the two considered events. Leveraging these concepts, we establish the temporal relation $\mathcal{T}_{allen}$ between two events e_i and e_j:

$$\mathcal{T}_{allen}(e_i, e_j) = \begin{cases} \text{True,} & \text{if } e_i.t_{start}, e_i.t_{end}, e_j.t_{start}, e_j.t_{end} \text{ respect conditions.} \\ \text{False,} & \text{otherwise} \end{cases} \tag{1}$$

Each relation captures a specific temporal interaction between intervals, enabling the precise modeling of complex temporal sequences. Specific Allen relations are denoted by using subscripts such as $\mathcal{T}_{before}(e_i, e_j)$.

3.3 Attribute Filtering

In addition to temporal relationships, activities are characterized by dependencies between event attributes. Specifically, recognizing an activity relies on recognizing the specific attribute values associated with events. For example, in a smart-home scenario, the location attribute may be crucial for identifying activities such as "cooking" or "watching TV". Thus, the definition of an activity can be refined by imposing constraints on the attribute values of its constituent events, introducing the notion of filtering. To formalize this, we define an attribute filter function F_a that evaluates the relevance of an event's attributes:

$$Fa(e) = \begin{cases} \text{True,} & \text{if the attributes of } e \text{ satisfy the filter conditions.} \\ \text{False,} & \text{otherwise} \end{cases} \tag{2}$$

Comparing the attribute values of two events can also determine their relevance to the same activity, which is crucial for accurately linking contextually connected events. For instance, in a surveillance system, two events occurring in the same area might need to share the same location attribute to accurately recognize a security breach or suspicious behavior. To formalize this, we define a relative attribute filter function Fr that evaluates the relevance of a pair of events' attributes:

$$Fr(e_1, e_2) = \begin{cases} \text{True,} & \text{if the attributes of } e_1, e_2 \text{ jointly satisfy the filter conditions.} \\ \text{False,} & \text{otherwise} \end{cases} \tag{3}$$

3.4 Composition and Definition of Activities

Formally, an activity A is a temporally-structured sequence of m events which follow specific temporal and attribute relationships. The recognition of an activity is formalized by the function R_A, which combines these temporal and attribute relations to determine whether a given set of events constitutes a recognized activity. This can be expressed as:

$$R_A(e_1, \ldots, e_m) = \bigwedge_{i=1}^{m} Fa_i(e_i) \wedge \bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^{n} Fr_{i,j}(e_i, e_j) \wedge \bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^{m} \mathcal{T}_{allen_{i,j}}(e_i, e_j) \tag{4}$$

We assume that an environment is constituted of n activities $\{A_1, \ldots A_n\}$ with corresponding recognition functions $\{R_{A_1}, \ldots R_{A_n}\}$.

4 Defining Interaction within the Environment

The interaction between the system and its environment evolves continuously. A stream of events is presented to the agent, generated by an underlying set of activities which prompt interaction. For realism, it is crucial to model not only the activities themselves but also the impact of interactions on the environment. Specifically, the agent's decisions have meaningful effects on the state of the environment, creating a closed-loop dynamic interaction. Traditionally, this decision-making process is modeled as a Markov Decision Process (MDP) [23] or a Partially Observable Markov Decision Process (POMDP) [46]. However, these approaches often fall short in capturing the complexities of event-driven environments, particularly those that exhibit rich contextual and historical dependencies [44].

4.1 Challenges in Decision making and Modeling as an MDP

The challenges frameworks face can be categorized into three main areas:

- **State Space Complexity**: The asynchronous and concurrent nature of activities, along with their inter-dependencies, significantly expands the state space. Each activity, defined by its events, attributes, and temporal dependencies, has multiple states of advancement. With multiple activities occurring simultaneously in the environment, representing states as the compounded relations of these activities becomes complex. The coupling of events from different activities creates a multi-dimensional activity space, where the complexity increases due to their inter-dependencies and overlapping temporal dynamics. The number of possible event combinations and correlations grows exponentially with the number of actors, activities, and attributes, complicating the transition function and leading to modeling complexity [45, 35, 29].
- **Contextual and Historical Dependencies**: The state of activities is not directly observable by the agent, which must rely on the history of events to make inferences. Because individual events cannot be fully understood without their associated temporal and attribute relations, the agent is compelled to consider the entire event stream to accurately infer the underlying activities. This process underscores the critical role of context and history in determining the significance of events. The reliance on historical context and continuous event streams breaks the Markov assumption, which assumes that future states depend only on the current state and not on the sequence of past events [5, 50].
- **Temporally-Structured Nature of Activities**: Events and activities are inherently dependent on time and structured in complex temporal patterns, rather than being independent, instantaneous transitions. The temporal relations between events are crucial for recognizing activities and must be specifically captured. This temporal structuring is essential because activities often span multiple time steps and involve events that interact over time. Therefore, enriched state representations that incorporate these temporal dynamics are necessary to accurately reflect real-world scenarios and support effective decision-making processes [44, 49].

4.2 Defining an Activity-Based MDP

To address these limitations, we propose a novel formulation of the decision-making problem in event-driven environments by building on the fundamentals of activity recognition. We define the Activity-Based Markov Decision Process (AB-MDP), which integrates principles of Activity Recognition within the Markov Decision Process framework. Similar to Contextual Markov Decision Processes [28] and Dynamic Contextual Markov Decision Processes [50], the state space in the AB-MDP is expanded to include not only the observable state information but also additional contextual information. History dependence is captured by a contextual variable that indicates whether an activity has been completed, conditioning rewards and transitions. This contextual variable is latent to the agent, making it a special case of POMDP. In this version of the AB-MDP, illustrated in Figure 3, we assume that interventions are relevant only when activities have been completed. This simplification allows the model to focus on the state of recognized activities rather than the entire event sequence.

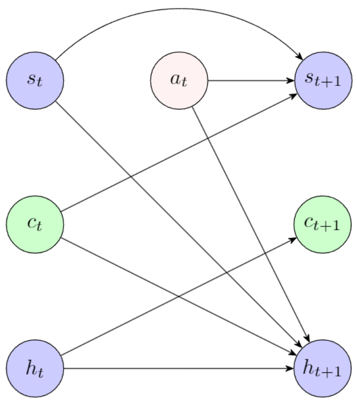


Figure 3 Causal diagram depicting the dependencies in an AB-MDP. Green circles represent unobserved variables. Here s_t and s_{t+1} are the current and next states, c_t and c_{t+1} are the current and next context variables, h_t and h_{t+1} are the current and next histories and a_t is the applied action.

▶ **Definition 1** (Activity-Based MDP). *Supposing an environment is defined by the presence of n activities, an AB-MDP is defined by the tuple $\langle S, E, C, A, T, \mathcal{R}\rangle$*

- *S is the observable space of the environment, constituted by any observable information outside of events.*
- *E is the space of events observations.*
- *C is a contextual vector of size n. Each element c_i indicates the completion status of the i-th activity, with $c_i \in \{True, False\}$. It is latent to the learning agent and must be inferred from the history of observations.*
- *A is the finite action space defined by the possible responses or reactions the system can take.*
- *T is a context dependent transition function defined as $T(s_t, a_t, c_t, s_{t+1}) = Pr(s_{t+1} \mid s_t, a_t, c_t)$, with $s_t, s_{t+1} \in S$, $a_t \in A$ and $c_t \in C$. The transition is influenced by the actions taken and the currently active activities.*
- *$\mathcal{R}$ is the reward function defined as $\mathcal{R}(s, a, c) = r$, which evaluates the success of actions in reacting to recognized activities.*

▶ **Example 2.** To illustrate the components of the AB-MDP, consider the example of an autonomous drone monitoring a restricted area. The state at each step s_t represents the drone's current position, battery level, and basic environmental data like wind speed. In contrast, events e_t capture instantaneous occurrences such as detecting movement near a perimeter fence, the activation of a door sensor, or an alarm signal triggered by unauthorized entry. The context variable c_t indicates whether specific activities, like unauthorized entry, have been completed, while the action a_t might involve the drone adjusting its position, zooming its camera, or sending an alert to security personnel.

At each time step t, the contextual variable c_i specifies whether there exists a subset of events $\{e_1, \ldots, e_m\}$ in the event history $h_t \in \mathcal{H}$ that satisfy the predicate conditions for activity i.

$$c_i = \begin{cases} \text{True}, & \{e_1, \ldots, e_m\} \subset h_t | R_{A_i}(e_1, \ldots, e_m) = \text{True} \\ \text{False}, & \text{otherwise} \end{cases} \tag{5}$$

This means that the next context occurs with a probability dependent on the history of events. Since this variable is latent to an observer of the environment, its values must be inferred from the observed events.

5 Description of the Open the Chests Environment

The `Open The Chests` environment implements an AB-MDP to simulate the complexities of real-world event-driven systems, providing a configurable platform for testing activity recognition and reactive decision-making algorithms. The environment models a scenario where an agent interacts with a series of chests that can be opened based on specific, unknown patterns of events. The agent's primary goal is to recognize these patterns by processing the observable event stream and deciding which buttons to press to open the appropriate chest. The environment's role is to generate events that respect the configured activities and allow interaction with their corresponding chests. This setup illustrates the dynamic and context-dependent nature of real-world systems, where recognizing event sequences allows the agent to modify the environment.

5.1 Game Mechanics

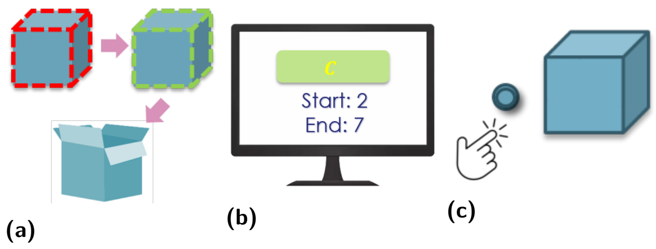


Figure 4 The main elements of the `Open The Chests` environment: (a) boxes and their respective states (`active`, `ready`, `open`); (b) event observations with their symbol, attributes (`background color`, `foreground color`) and temporal information (`start`, `end`); (c) buttons for interacting with chests to modify their state.

The `Open the Chests` environment consists of several key elements, each playing a critical role in the simulation: chests, observations, buttons, and rewards. Figure 4 illustrates how these elements are presented in the environment.

- **Event Observations**: Event observations are the *event symbols, attributes and temporal information* presented to the agent. These observations form the event stream that the agent must analyze to recognize patterns and make decisions. At each step, the environment emits *last occurred event* (see Sec.5.2), generated according to the activities present in the environment (see Sec.5.3).
- **Chests**: Chests represent the *activities* present in the environment that the agent needs to recognize and address. Each chest corresponds to a specific *pattern of events* within the observable event stream. The state of chests is partially observable: the information of whether a chest is *open* or *active* is known, while the information of a chest being *ready* is hidden (see Sec.5.4).
- **Buttons**: Buttons are the interactive elements associated with each chest representing *actions*. Pressing the corresponding button allows opening the chest if the associated pattern of events is present in the event history, i.e. if the chest is `ready` (see Sec.5.4).
- **Rewards**: Rewards are the *feedback* which the environment provides to the agent based on its actions. If the button is pressed at the correct time, the corresponding chest opens and a positive reward is given; otherwise, a negative or no reward is received (see Sec.5.5).

The environment continuously generates events based on predefined activities until all chests are recognized and opened. To solve this, the agent must monitor the event stream history, detect patterns, and infer the latent context. Upon recognizing a pattern corresponding to a chest activity, the agent must decide to press the appropriate button, linking the pattern, chest, and action.

5.2 Event Observation and Interaction Time

Events are displayed as symbols with varying values, colors, and background colors, representing different types of detections in the environment (Figure 5). Each event also carries continuous time information, indicating its start and end times. These events are presented to the agent one by one upon completion, allowing the agent to make decisions based on fully observed events. This approach ensures that the environment operates in discrete steps. While the timeline of events is not directly visible to the agent, it is implicitly understood through the sequence displayed on the observation screen. The environment allows for variation in event length during configuration, adding complexity and realism to the simulation. This variability challenges the agent to adapt to different event durations, enhancing the robustness of pattern recognition algorithms.

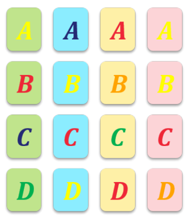


Figure 5 Possible events symbols and their attributes represented by symbols and colors.

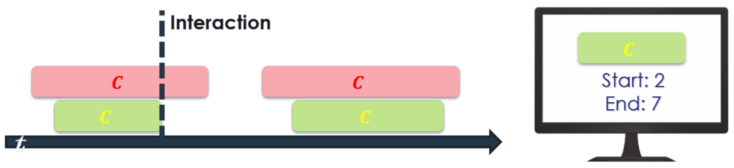


Figure 6 An event is communicated to the agent at its completion.

▶ **Example 3.** Consider an activity defined by two overlapping events, both marked with the symbol C, as shown in the Figure 6. The activity recognition rule $R_A(e_1, e_2) =$

$F_a(e_1, C, \textit{green}, \textit{yellow}) \wedge F_a(e_2, C, \textit{red}, \textit{red}) \wedge \mathcal{T}_{contains}(e_1, e_2)$ involves detecting event e_1 with the attributes C, *green*, and *yellow*, and event e_2 with the attributes C, *red*, and *red*, while maintaining the temporal relation $\mathcal{T}_{contains}(e_1, e_2)$. Since the *contains* relation requires that $t_{\mathrm{end}_{e_1}} < t_{\mathrm{end}_{e_2}}$, event e_1 will be generated and communicated first, along with its specified colors and timestamps.

5.3 Defining event patterns and generating events

When configuring event patterns for each chest, we use Temporal and Attribute relations as defined in Equation 4. To facilitate the management of these relations, we use Höppner's matrix form [30] to transform them into a structured format, which makes it easier to check for continuity and detect any contradictions. To handle event generalisation and simplify the complexity of multiple concurrent activities, we represent patterns as memory-enriched automata [31]. Their goal is to track both current and pending events, while storing relevant past events that are needed for generation. During execution, the next event to generate is thus selected with respect to all activities and their current execution step. Each activity is configured to begin after a certain delay, which can be specified during its definition. Additionally, Allen Relations can be parameterized to introduce varying delays between events, adding complexity and realism to the simulation. Finally, empty filters can be defined, allowing for variations in attributes during event generation.

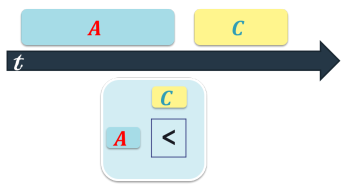


Figure 7 An illustration of a defined pattern and its associated matrix representation.

```
INSTANTIATE
   - name: e1
     type: A
     params:
       fg: red
       bg: blue
     duration:
       mu: 5
       sigma: 2
   - name: e2
     type: C
     params:
       fg: blue
       bg: yellow
     duration:
       mu: 6
       sigma: 2
```

Figure 8 Defining the events and attribute filters of a two-event pattern.

```
RELATIONSHIP:
   - type: after
     events:
       - e1
       - e2
     other:
       gap_dist:
           mu: 4
           sigma: 1
```

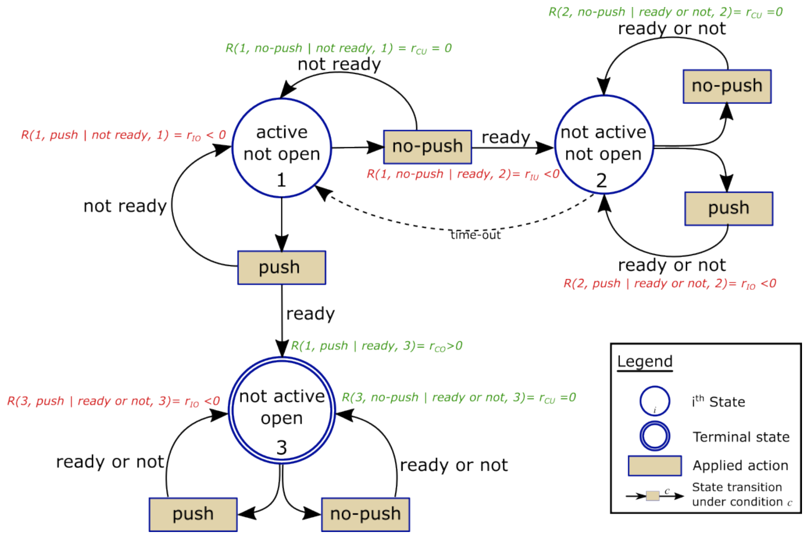
Figure 9 Defining the relations of a two-event pattern.

▶ **Example 4.** To define a pattern consisting of two events where one occurs before the other, and where the events are represented by symbols A and C, we start by specifying their attributes and relations, as shown in Figures 8 and 9. We use the parameters `mu` and `sigma` to define a normal distribution that will be used to sample the duration of events or the time between events. In this case, the matrix representation of this pattern (Figure 7) includes only one temporal relation between the two events.

Complexity is added to the environment by introducing noise and randomness into the event stream, simulating some of the inherent uncertainty of real-world systems. Specifically, patterns can be generated with varying degrees of noise, enabling the system to add events to the stream that are unrelated to the execution of current activities.

5.4 Chests and Buttons: Interaction and Box States

Chests represent activities in the environment, each with three binary state values: `active`, `ready`, and `open`. At the start of the game, all chests are closed (not `open`). A chest becomes `active` once it starts generating events that are observed in the event stream. When the sequence of events is fully generated and present in the stream, the corresponding chest becomes `ready` to open by pushing a button. If the button is pushed, the chest is marked as `open`, its pattern is removed from the environment, and it stops generating further events. If the button is not pushed, the pattern generation continues and eventually restarts. Until a chest is opened, its associated pattern continues to repeat, providing the agent with multiple opportunities to recognize and interact with the sequence. The `active` and `open` states represent the observable part of the environment and are communicated to the agent with each event observation. The `ready` state serves as the activity context, determining the outcome of button actions.



R(2, no-push | ready or not, 2)= r_{CU} =0
ready or not
R(1, no-push | not ready, 1) = r_{CU} = 0
not ready
no-push
R(1, push | not ready, 1) = r_{IO} < 0
active
not open
1
no-push
ready
not active
not open
2
R(1, no-push | ready, 2)= r_{IU} <0
push
not ready
time-out
push
ready or not
R(2, push | ready or not, 2)= r_{IO} <0
ready
R(1, push | ready, 3)= r_{CO}>0
R(3, push | ready or not, 3)= r_{IO} <0
not active
open
3
R(3, no-push | ready or not, 3)= r_{CU} =0
ready or not
ready or not
push
no-push
Legend
i
ith State
Terminal state
Applied action
c
State transition under condition c

Figure 10 The state transition graph of a single chest. A chest is initially `active` and `not open`. Its transitions are conditioned by the pushing buttons and on its associated `ready` value.

▶ **Example 5.** Suppose that only one chest is defined in the environment using the activity definitions in Example 3. Initially it is *active* and generates events one by one. Once both its associated events have been observed it will pass to the state *ready*, meaning its associated context variable $c_1 = True$ will indicate activity completion. If the button is correctly pressed, the chest will *open* and no further events will be generated. Otherwise the chests is deactivated and goes back to the *active* state after a delay. This transition is illustrated in Figure 10.

5.5 Rewards

Reward is defined in terms of the number of correctly opened chests N_{CO}, incorrectly opened chests N_{IO}, correctly unopened chests N_{CU}, and incorrectly unopened chests N_{IU}. These notions are equivalent to true positives, true negatives, false positives, and false negatives, respectively. The action $a_t \in \{\text{True}, \text{False}\}^n$ defines the chests the agent attempted to open and the time at which the agent pressed the button corresponding to each chest. We use the vector $c_t \in \{\text{True}, \text{False}\}^n$ to represent which of the chests were ready to open at time t. We can thus define the following:

- $N_{CO} = a_t \wedge c_t$ meaning all correctly opened chests
- $N_{IO} = a_t \wedge (\neg c_t)$ meaning all incorrectly attempted chests
- $N_{CU} = (\neg a_t) \wedge (\neg c_t)$ meaning all correctly unopened chests
- $N_{IU} = (\neg a_t) \wedge c_t$ meaning all incorrectly unattempted chests

We define a reward for each type of chest: r_{CO}, r_{IO}, r_{CU}, r_{IU}, allowing us to calculate the final reward r.

$$r = r_{CO} \cdot N_{CO} + r_{IO} \cdot N_{IO} + r_{CU} \cdot N_{CU} + r_{IU} \cdot N_{IU}$$

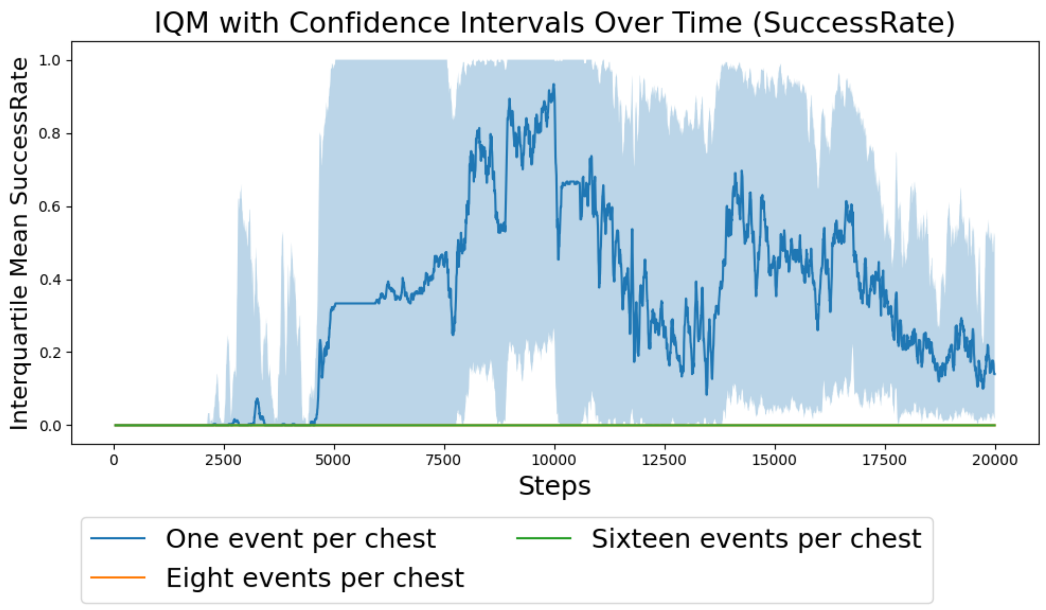
For the time being, we set the values $r_{CO} = 1$, $r_{IO} = -1$, $r_{CU} = 0$, and $r_{IU} = -1$.

6 Validation of the Open The Chests Environment

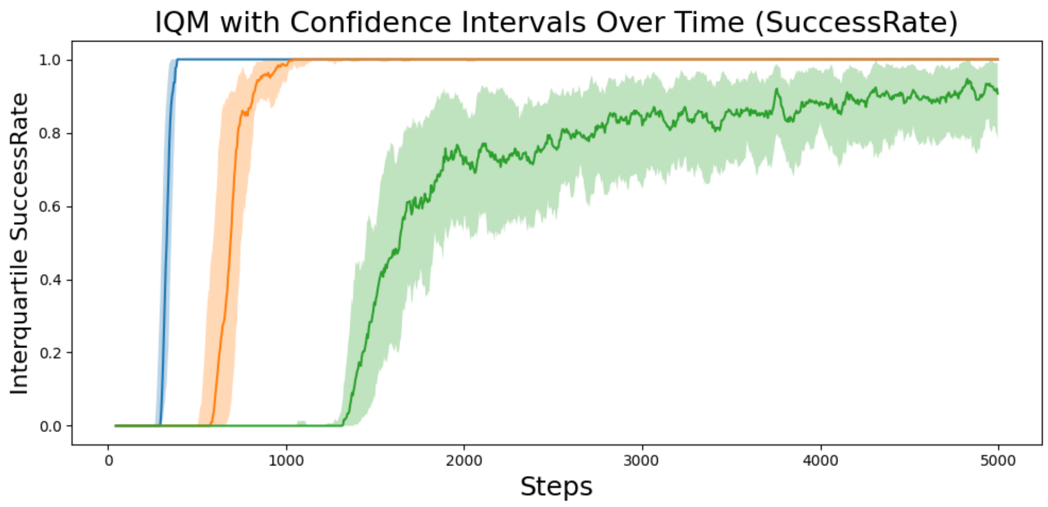
The goal of our validation was to ensure the proper execution of the OpenTheChests environment and to establish performance baselines using two algorithms: Deep Q-Network (DQN) and Deep Transformer Q-Network (DTQN) [21]. Successfully generating meaningful and consistent results across both algorithms confirms that the environment functions as intended, providing a reliable platform for testing and comparing different algorithms. This validation also demonstrates the environment's capability to effectively differentiate between the performance and behavior of distinct decision-making approaches, ensuring its integration with existing methods.

To achieve this, we configured three scenarios with varying levels of complexity. Each configuration contained 5 chests but differed in the number of events per activity: 1, 8, and 16 events. The simplest configuration associates one unique event per chest, meaning it doesn't require additional temporal or context dependencies. Once the event is observed, the expected response is to identify the appropriate chest and press its button. Conversely, the environments with eight and sixteen events per chest require identifying sequences of appropriate length as well as their respective attribute and temporal relations. To configure these patterns, we randomly selected the corresponding number of event filters and Allen temporal interval relations. Finally, the environment was configured with varying levels of noise per activity, ranging from 0.1 to 0.3, meaning that a proportionate amount of non-relevant events was generated alongside the activity patterns. Our evaluations is shown in Figure 11, where each learning curve presents the success rate of the agents during training across the three different environments. To asses the performance in both algorithms, we measured the success rate in each scenario, reflecting the ability of agents to correctly identify and interact with the appropriate chests. We utilized the `rliable` library [1] to calculate the Interquartile Mean (IQM), which provides a robust measure of central tendency, along with stratified bootstrap confidence intervals to capture the variability of the results across five random seeds. These metrics were plotted over the course of training, allowing us to observe the learning progress and stability of each algorithm in handling the varying levels of complexity within the environments.

Both DQN and DTQN algorithms were successfully integrated with the environment by leveraging the standardised `gym` framework. As expected, DQN performed poorly in scenarios requiring historical context and complex temporal dependencies. The algorithm struggled to effectively handle tasks that demanded memory of past events or intricate event interd-ependencies, which are critical in the `Open The Chests` environment. DQN's design,



(a) DQN



(b) DTQN

Figure 11 Success rates of DTQN and DQN across three environments during training with 1, 8, and 16 events per activity. Each pattern was configured with a noise value between 0.1 and 0.3. Values are measured across 5 random seeds.

optimized for simpler, state-based decisions, lacks the mechanisms necessary to process and utilize long-term dependencies, leading to suboptimal performance in these context-sensitive scenarios. Interestingly, we also observed struggles in simpler scenarios without history dependence, likely due to the challenges posed by multiple parallel activities and the multi-dimensional nature of the actions required. DTQN demonstrated superior performance, especially in more complex scenarios, due to its ability to incorporate past observations. The use of transformer architectures allows DTQN to maintain and utilize a memory of sequential events, enabling it to make more informed decisions based on the historical context. However, as the scenarios became increasingly complex with longer contexts, DTQN's performance began to decline, likely due to limitations in the model's parameter settings, such as the fixed context window size, which may not fully capture extended dependencies, as well as its capacity to effectively separate parallel activities. Further exploration of these methods, particularly through the use of custom configurations to isolate specific complexities, would allow for a better understanding of the algorithms' limitations. This underscores the value of the `Open The Chests` environment. Additionally, the lack of interpretability in both algorithms remains an important consideration for future development, as it impacts their practical applicability in real-world scenarios.

7 Conclusion and Perspectives

The `OpenTheChests` environment facilitates defining benchmarks of varying complexities, depending on the number of chests and the complexity of their corresponding patterns. Integrated with the `gym` [10] framework, it models dynamic, interactive scenarios where the agent must recognize patterns of events and make timely decisions, illustrating real-world system complexities. Through the integration of Activity-Based Markov Decision Processes (AB-MDP), `OpenTheChests` simulates context-dependent, sequential decision-making tasks, facilitating comprehensive testing and development of advanced algorithms.

Looking ahead, several enhancements are planned to further develop the capabilities of `OpenTheChests`. These include:

- **Dependence Between Activities and Model Expansion**: Introducing mechanisms where activities can influence each other, creating more complex inter-dependencies and richer scenarios for testing decision-making strategies. We aim to develop an advanced AB-MDP that considers intermediate activity states.
- **Event Sharing Between Activities**: Implementing various event consumption policies that govern how events are shared or partitioned among concurrent activities, enabling the exploration of different coordination strategies.
- **Advanced Attribute Relations**: Developing more sophisticated attribute relations that can span multiple events, enhancing the ability to model and recognize complex patterns and dependencies.

Future research will focus on expanding the complexity of the `OpenTheChests` environment and exploring its applications in various real-world scenarios. While initial results showed good performance, both DQN and DTQN lacked interpretability. Further goals include improving the interpretability of the algorithms, enabling clearer insights into decision-making processes, and enhancing the overall robustness of the environment.

References

1 Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

2 Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. Probabilistic complex event recognition: A survey. *ACM Computing Surveys (CSUR)*, 50(5):1–31, 2017.

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

4 Anindya Das Antar, Masud Ahmed, and Md Atiqur Rahman Ahad. Challenges in sensor-based human activity recognition and a comparative analysis of benchmark datasets: A review. In *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and*

2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), pages 134–139. IEEE, 2019.

5 Fahiem Bacchus, Craig Boutilier, and Adam Grove. Structured solution methods for non-markovian decision processes. In *AAAI/IAAI*, pages 112–117. Citeseer, 1997.

6 Iyad Batal, Gregory F Cooper, Dmitriy Fradkin, James Harrison, Fabian Moerchen, and Milos Hauskrecht. An efficient pattern mining approach for event detection in multivariate temporal data. *Knowledge and information systems*, 46:115–150, 2016.

7 Benjamin Beyret, José Hernández-Orallo, Lucy Cheke, Marta Halina, Murray Shanahan, and Matthew Crosby. The animal-ai environment: Training and testing animal-like artificial cognition. *arXiv preprint arXiv:1909.07483*, 2019.

8 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.

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

10 Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. `arXiv:arXiv:1606.01540`.

11 Christos G Cassandras and Stéphane Lafortune. *Introduction to discrete event systems*. Springer, 2008.

12 Wuhui Chen, Xiaoyu Qiu, Ting Cai, Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Deep reinforcement learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1659–1692, 2021.

13 Jean-René Coffi, Christophe Marsala, and Nicolas Museux. Adaptive complex event processing for harmful situation detection. *Evolving Systems*, 3:167–177, 2012.

14 Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):1–62, 2012.

15 Ala' Darabseh and Nikolaos M Freris. A software-defined architecture for control of iot cyberphysical systems. *Cluster Computing*, 22(4):1107–1122, 2019.

16 Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Interval temporal logics: a journey. *Bulletin of EATCS*, 3(105), 2013.

17 Florenc Demrozi, Graziano Pravadelli, Azra Bihorac, and Parisa Rashidi. Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey. *IEEE access*, 8:210816–210836, 2020.

18 Derui Ding, Qing-Long Han, Yang Xiang, Xiaohua Ge, and Xian-Ming Zhang. A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing*, 275:1674–1683, 2018.

19 Anton Dries and Luc De Raedt. Towards clausal discovery for stream mining. In *Inductive Logic Programming: 19th International Conference, ILP 2009, Leuven, Belgium, July 02-04, 2009. Revised Papers 19*, pages 9–16. Springer, 2010.

20 Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

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

22 Ido Finder, Eitam Sheetrit, and Nir Nissim. Time-interval temporal patterns can beat and explain the malware. *Knowledge-Based Systems*, 241:108266, 2022.

23 Frédérick Garcia and Emmanuel Rachelson. Markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 1–38, 2013.

24 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.

25 Ben Goertzel, Nil Geisweiller, Lucio Coelho, Predrag Janičić, and Cassio Pennachin. *Real-World Reasoning: Toward Scalable, Uncertain Spatiotemporal, Contextual and Causal Inference*, volume 2. Springer Science & Business Media, 2011.

26 Shaogang Gong and Tao Xiang. Recognition of group activities using dynamic probabilistic networks. In *Proceedings ninth IEEE international conference on computer vision*, pages 742–749. IEEE, 2003.

27 Valentin Goranko, Angelo Montanari, and Guido Sciavicco. A road map of interval temporal logics and duration calculi. *Journal of Applied Non-Classical Logics*, 14(1-2):9–54, 2004.

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

29 Derek Hao Hu, Sinno Jialin Pan, Vincent Wenchen Zheng, Nathan Nan Liu, and Qiang Yang. Real world activity recognition with multiple goals. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 30–39, 2008.

30 Frank Höppner. Learning temporal rules from state sequences. In *IJCAI Workshop on Learning from Temporal and Spatial Data*, volume 25. Citeseer, 2001.

31 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

32 Nikos Katzouris, Evangelos Michelioudakis, Alexander Artikis, and Georgios Paliouras. Online learning of weighted relational rules for complex event recognition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II 18*, pages 396–413. Springer, 2019.

33 Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

34 Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4501–4510, 2020.

35 Niels Landwehr. Modeling interleaved hidden processes. In *Proceedings of the 25th international conference on Machine learning*, pages 520–527, 2008.

36 D Luckham. The power of events: An introduction to complex event processing in distributed enterprise systems. additions-wesley. *Reading*, 2001.

37 Sultan Javed Majeed and Marcus Hutter. On q-learning convergence for non-markov decision processes. In *IJCAI*, volume 18, pages 2546–2552, 2018.

38 Nijat Mehdiyev, Julian Krumeich, David Enke, Dirk Werth, and Peter Loos. Determination of rule patterns in complex event processing using machine learning techniques. *Procedia Computer Science*, 61:395–401, 2015.

39 Vlad I Morariu and Larry S Davis. Multi-agent event recognition in structured scenarios. In *CVPR 2011*, pages 3289–3296. IEEE, 2011.

40 Robert Moskovitch. Multivariate temporal data analysis-a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(1):e1430, 2022.

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

42 Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

43 Carlos F Pfeiffer, Veralia Gabriela Sánchez, and Nils-Olav Skeie. A discrete event oriented framework for a smart house behavior monitor system. In *2016 12th International Conference on Intelligent Environments (IE)*, pages 119–123. IEEE, 2016.

44 Emmanuel Rachelson. *Temporal markov decision problems.* PhD thesis, Citeseer, 2009.

45 Emmanuel Rachelson, Gauthier Quesnel, Frédérick Garcia, and Patrick Fabiani. A simulation-based approach for solving generalized semi-markov decision processes. In *ECAI 2008*, pages 583–587. IOS Press, 2008.

46 Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 387–414. Springer, 2012.

47 Richard S Sutton. The quest for a common model of the intelligent decision maker. *arXiv preprint arXiv:2202.13252*, 2022.

48 Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

49 Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1250–1257. IEEE, 2012.

50 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.

51 Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

52 Aashma Uprety and Danda B Rawat. Reinforcement learning for iot security: A comprehensive survey. *IEEE Internet of Things Journal*, 8(11):8693–8706, 2020.

53 Antonio Vitale, Alpha Renner, Celine Nauer, Davide Scaramuzza, and Yulia Sandamirskaya. Event-driven vision and control for uavs on a neuromorphic chip. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 103–109. IEEE, 2021.

54 Michalis Vrigkas, Christophoros Nikou, and Ioannis A Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28, 2015.

55 Cuebong Wong, Erfu Yang, Xiu-Tian Yan, and Dongbing Gu. Autonomous robots for harsh environments: a holistic overview of current solutions and ongoing challenges. *Systems Science & Control Engineering*, 6(1):213–219, 2018.

A Example configuration

This appendix provides an example configuration for the `Open The Chests` environment, illustrating how to set up chests with patterns of multiple events. The first step to configuring the environment is to define a set of event types along with their foreground and background colors. This is done by specifying the sets of all possible event types along with all possible foreground and background colors, as follows:

$$\begin{aligned}\text{Types} &= \{A, B, C, D, E, F, G, H, I, J\}\\ \text{Background Colors} &= \{\text{red}, \text{blue}, \text{green}, \text{orange}, \text{pink}\}\\ \text{Foreground Colors} &= \{\text{red}, \text{blue}, \text{green}, \text{orange}, \text{pink}\}\end{aligned}$$

The next step would be to define activities using filters and Allen Relations. Below is an example for one chest, showing how to specify the events and their temporal relations.

$$\begin{aligned}
R_{activity}(e_1, \ldots, e_8) = &\mathit{Fa}_1(e_1, \mathrm{B}, \mathrm{pink}, \mathrm{orange}) \\
&\wedge \mathit{Fa}(e_2, \mathrm{D}, \mathrm{red}, \mathrm{green}) \\
&\wedge \mathit{Fa}(e_3, \mathrm{E}, \mathrm{orange}, \mathrm{blue}) \\
&\wedge \mathit{Fa}(e_4, \mathrm{G}, \mathrm{blue}, \mathrm{pink}) \\
&\wedge \mathit{Fa}(e_5, \mathrm{H}, \mathrm{green}, \mathrm{red}) \\
&\wedge \mathit{Fa}(e_6, \mathrm{I}, \mathrm{pink}, \mathrm{orange}) \\
&\wedge \mathit{Fa}(e_7, \mathrm{J}, \mathrm{green}, \mathrm{blue}) \\
&\wedge \mathit{Fa}(e_8, \mathrm{C}, \mathrm{orange}, \mathrm{pink}) \\
&\wedge \mathit{Fr}(e_2, e_3, \{4, 1\}) \\
&\wedge \mathit{Fr}(e_4, e_5, \{5, 2\}) \\
&\wedge \mathit{Fr}(e_7, e_8, \{3, 1\}) \\
&\wedge \mathcal{T}_{during}(e_1, e_2) \\
&\wedge \mathcal{T}_{after}(e_2, e_3) \\
&\wedge \mathcal{T}_{metBy}(e_3, e_4) \\
&\wedge \mathcal{T}_{after}(e_4, e_5) \\
&\wedge \mathcal{T}_{during}(e_5, e_6) \\
&\wedge \mathcal{T}_{metBy}(e_6, e_7) \\
&\wedge \mathcal{T}_{after}(e_7, e_8)
\end{aligned}$$

In this example, $\mathit{Fa}_i(e_i, \mathrm{type}, \mathrm{fg}, \mathrm{bg})$ specifies the type, foreground, and background colors for each event e_i. This means that during generation, only events that satisfy the provided symbols and attributes will be selected. Relations like $\mathcal{T}_{during}(e_1, e_2)$ and $\mathcal{T}_{metBy}(e_6, e_7)$ guide the generation of the start and end times for the events e_1 through e_8, ensuring that these events occur in a sequence that adheres to the specified temporal constraints. Finally relative filters, like $\mathit{Fr}(e_2, e_3, \{4, 1\}$, allow us to define the relative distance between events in relations like "before" and "after," which imply a gap between the events, with this gap being determined by sampling from a normal distribution characterized by a mean of μ and standard deviation σ. The temporal relations between events in this configuration can also be summarized using a matrix of interval relations, as shown in Table 1.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_1	=	d						
e_2		=	>					
e_3			=	mi				
e_4				=	>			
e_5					=	d		
e_6						=	mi	
e_7							=	>
e_8								=

Table 1 Interval relation matrix representing the temporal relations between events e_1 to e_8 as specified in the configuration.

The defined pattern is then transformed into a memory-enriched automaton, where each state corresponds to the generation of an event. Relevant temporal data, such as the initialization time of the activity and past generated events, is stored in memory, ensuring the consistent generation of complex event sequences.