Open the Chests

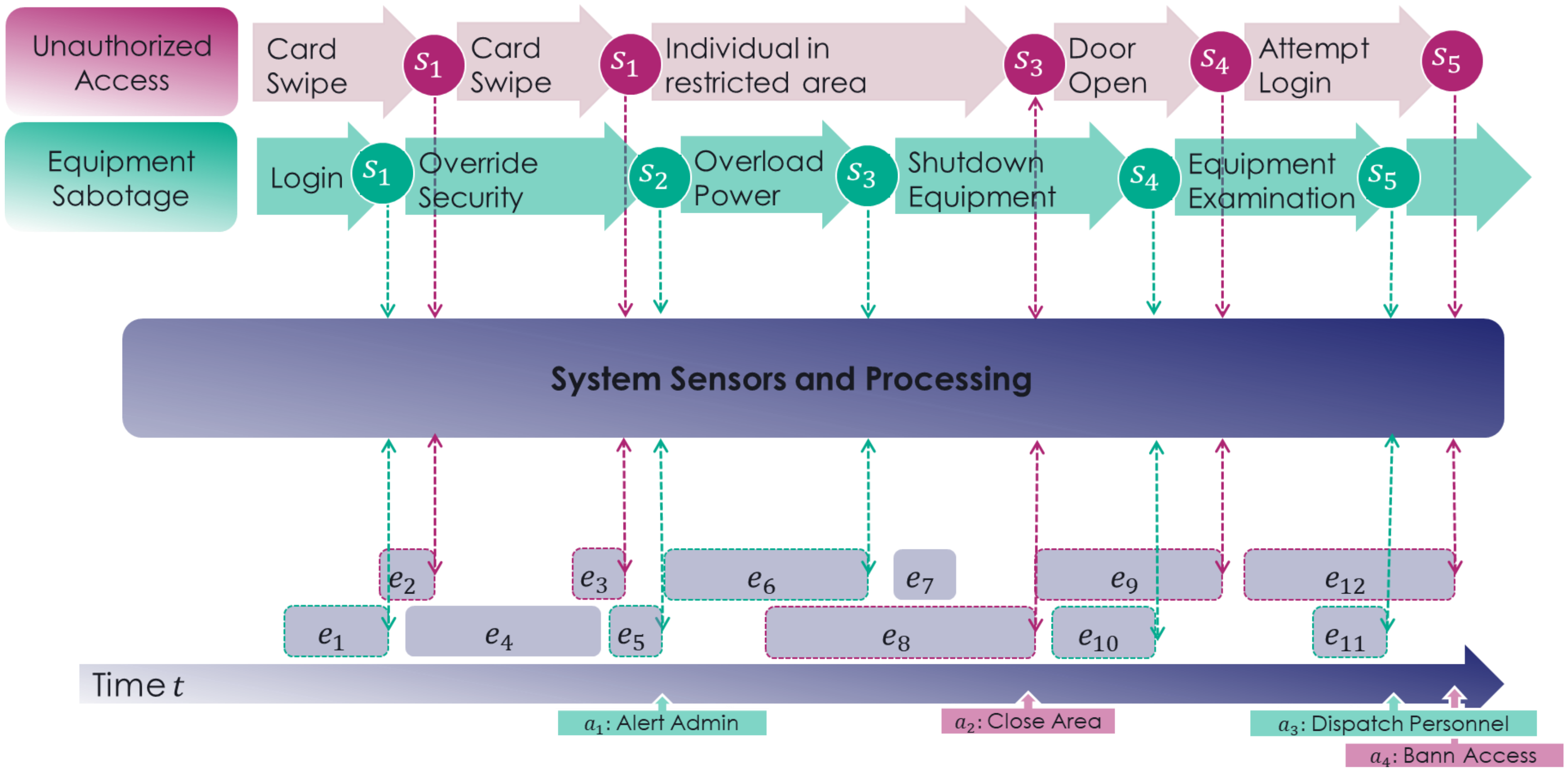
An Environment for Activity Recognition and Sequential Decision Problems Using Temporal Logic

I. Stoyanova[1,2], N. Museux[2],
Sao Mai Nguyen[1], David Filliat[1]
(1)ENSTA Paris (2)Thales
www.thalesgroup.com

REACTIVE ACTIVITY RECOGNITION BRINGS INSIGHT TO VARIOUS FIELDS

ALL SHARE THE NEED TO UNDERSTAND AND ACT UPON THE CURRENT SITUATION

Activities can be recognized in the stream through Pattern Conditions and reacted to using Reaction Rules



Activity : A process or behavior that unfolds over time.

$$\mathcal{A} = (A_1, A_2)$$

Pattern Condition : A signature of events allowing the recognition of an *Activity*

$$C_{A_1}(e_1, \dots, e_n) = \begin{cases} True \text{ if } e_1, \dots, e_n \text{ is } A_1 \\ False \text{ otherwise} \end{cases}$$

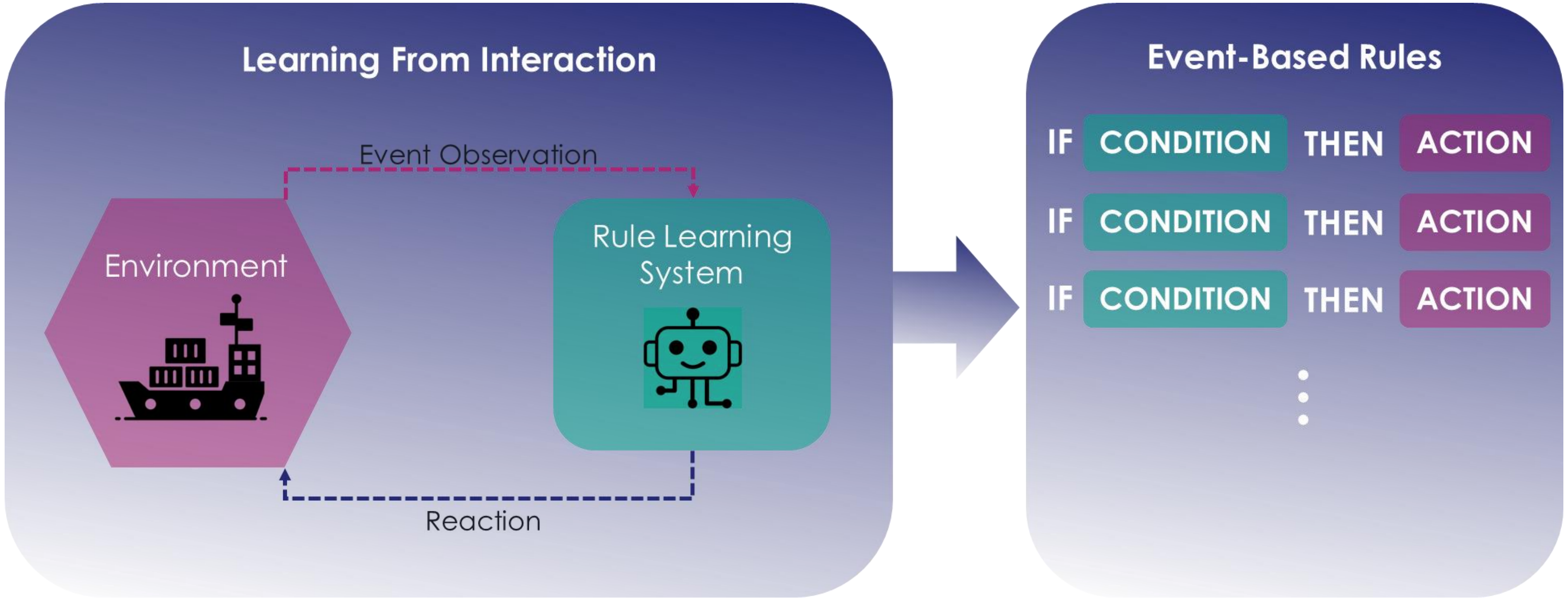
Reaction Rules : If a signature of events is recognized, respond with the appropriate action.

If $\exists \{e_1, \dots, e_n\} \in h_t$

Such that $C_{A_1}(e_1, \dots, e_n) = True$

Apply action a_1

AUTOMATING REACTIVE ACTIVITY RECOGNITION REQUIRES A REINFORCEMENT LEARNING ENVIRONMENT



> Objectives

- Automating Activity Recognition
- Extracting Patterns Automatically
- Optimising Action Choice

> Challenges

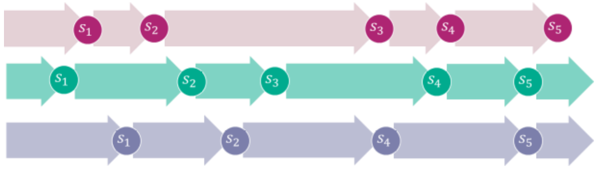
- Lack of Datasets and Simulators
- Unknown Activity Patterns
- Need for dynamic interaction

The underlying Decision Model is not a standard Markovian Process

No existing training environment that covers all these constraints

State Space Complexity

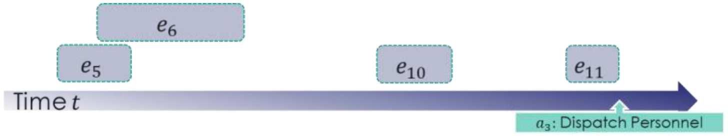
- **Asynchronous** and **concurrent** activities
- **Inter-dependencies** between events



(Landwehr, 2008)

Contextual and Historical Dependency

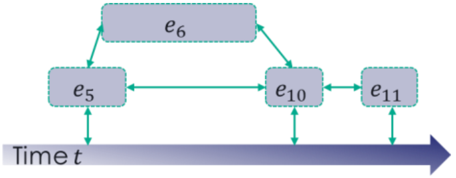
- Activities deduced from the **history of events**
- Activity states are **not observable**



(Tennenholtz et al., 2023)

Temporally-Structured Nature of Activities

- Events and activities are **time-dependent**
- Presence of **complex temporal relations**



(Rachelson, 2006)

Requirement of a training environment to tackle that CLASS of problems

- Simple
- Representative
- Easy to configure
- Standard API
- Evaluation friendly

Formalizing events as detections occurring over an interval of time

Detection : attributes representing signal interpretation

$$sym \in D_{sym} \; Attr = \{attr_1, attr_2, \dots\} \in D_{Attr}$$

$$\boldsymbol{sym_{alarm} = Alarm}$$
$$\boldsymbol{Attr_{alarm} = \{room_{id}, sensor_{id}, val\_db\}}$$

Interval : beginning and end of detection

$$I = (t_{start}, t_{end})$$
$$t_{start} \leq t_{end}$$

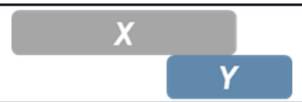
Event: Allows associating a detection to an interval

$$e = (sym, Attr, I)$$

Event Trace: An ordered sequence of k temporally positioned events.

$$h_k = (e_1, e_2, \dots, e_k)$$
where $\forall e_i, e_j \in h_k, i < j \equiv e_i.I.t_{end} \leq e_j.I.t_{end}$

Temporal Relations are Expressed through Allen's Interval Algebra

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

Allen, J. F., & Ferguson, G. (1994). Actions and events in interval temporal logic. Journal of logic and computation, 4(5), 531-579.

Formalizing Pattern Conditions and using them for Environment Transitions

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

> Pattern Condition: Specify signatures allowing the identification of an activity of interest

Absolute content filtering: Select events that have specific Types and Attributes
$Fa(e) \coloneqq (e.symbol = "alarm")$
$\wedge\, (e.room = 7)$

Relative content filtering: Select events that have comparable Types and Attributes
$Fr(e_1, e_2) \coloneqq (e_1.symbol = e_2.symbol)$

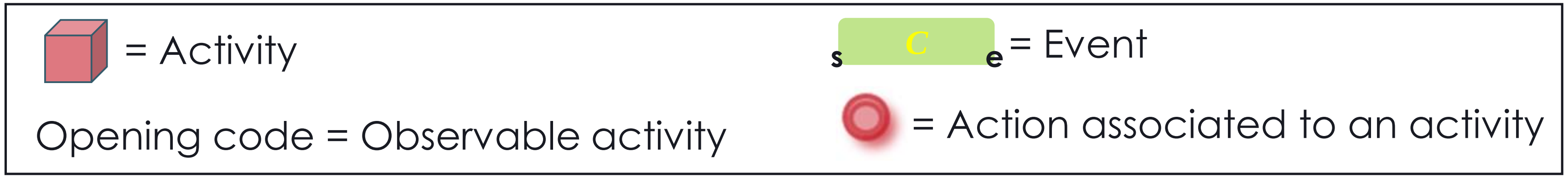
Temporal Relationship: Select events whose temporal intervals respect a certain condition
$R_{metby}(e_1, e_2) \coloneqq$
$e_1.t_{start} \geq e_1.t_{end} = e_2.t_{start} \geq e_2.t_{end}$

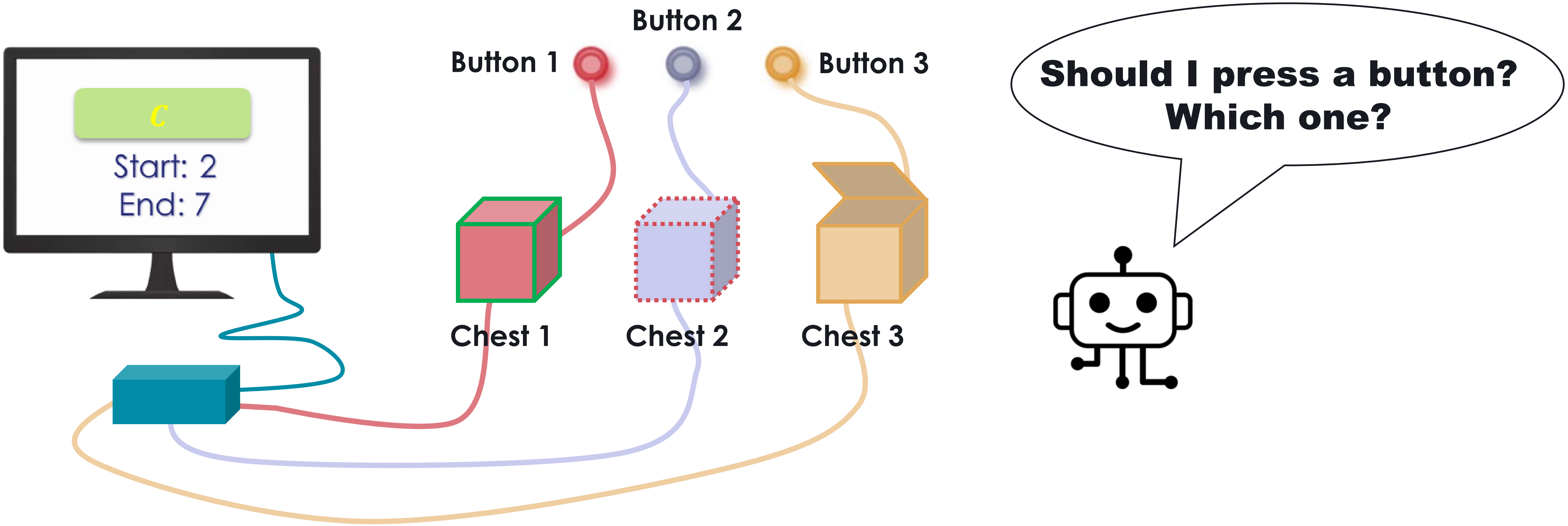
> Environment Evolution: Conditioned on observable information and activity completion

- **AB-MDP** defines a **context dependent transition function** defined as $T(s_t, a_t, c_t, s_{t+1}) = \Pr(s_{t+1} | s_t, a_t, c_t)$

with $c_i = \begin{cases} True, & \exists\{e_1, \ldots, e_n\} \subset h_t \mid C_{A_i}(e_1, \ldots, e_n) = True \\ False, & otherwise \end{cases}$

The game: find what combination of symbols opens which chest





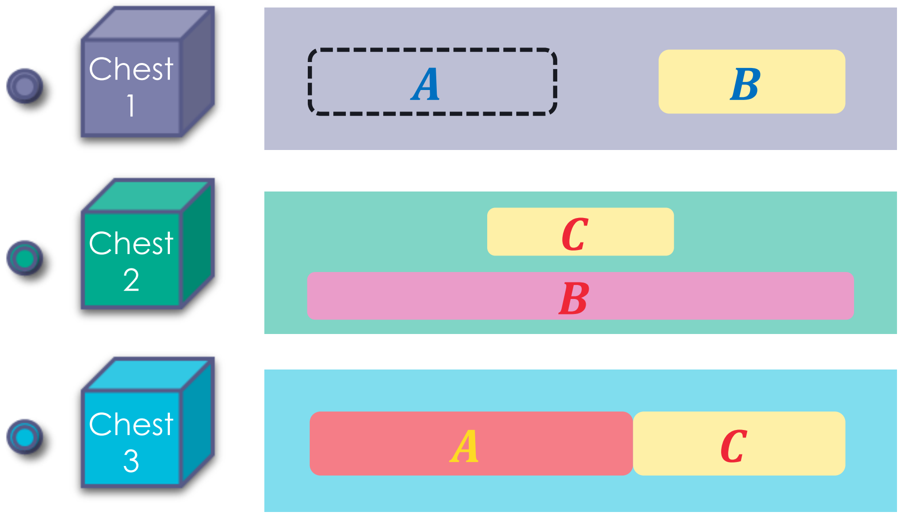
The game: A symbol with its attributes models an event

> Classes of event

- A, B, …, Z
- Could be any printable character

> Event information

- Start time, end time, so that start <= end
- Attributes
  - Foreground colour, background colour
  - Could also be: font, emphasis, size, etc.



> An opening code is a pattern of events

- **Chest 1** is **ready** to open when
  - “**A**(**fg:** BLUE, **bg:** *, **start:** s, **end:** s+4) FOLLOWED-BY **B**(**fg:** BLUE, **bg:** YELLOW, **start:** s’, **end:** s’+2)”
- **Chest 2** is **ready** to open when
  - “**B**(**fg:** RED, **bg:** PINK, **start:** s, **end:** s+3) OVERLAPPED-BY **C**(**fg:** RED, **bg:** YELLOW, **start:** s’, **end:** s’+7)”
- **Chest 3** is **ready** to open when
  - “**A**(**fg:** RED, **bg:** YELLOW, **start:** s, **end:** s+5) MET-BY **C**(**fg:** RED, **bg:** YELLOW, **start:** s’, **end:** s’+4)”

Game setup

> Overall game setting file

- Vocabulary (Implicit grammar): Types, attributes, values
- Pointers to the chest setup files

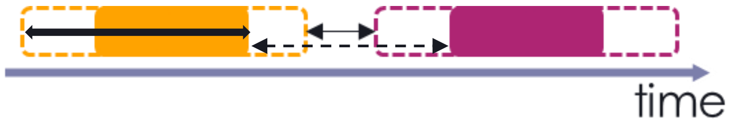
```yaml
EVENT_TYPES:
  NORMAL:
    -A
    -B
    -C
    -D
    -E
    -F
    -G
    -H
    -I
    -J
  NOISE:
    -K
    -L
    -M
```

```yaml
INSTRUCTIONS:
  -two_per_box/chest_1.yaml
  -two_per_box/chest_2.yaml
  -two_per_box/chest_3.yaml
```

```yaml
EVENT_ATTRIBUTES:
  NORMAL:
    fg:
      -red
      -blue
      -green
      -orange
      -pink
    bg:
      -red
      -blue
      -green
      -orange
      -pink
  NOISE:
    fg:
      -yellow
      -purple
      -black
    bg:
      -yellow
      -purple
      -black
```

> Chest setting file

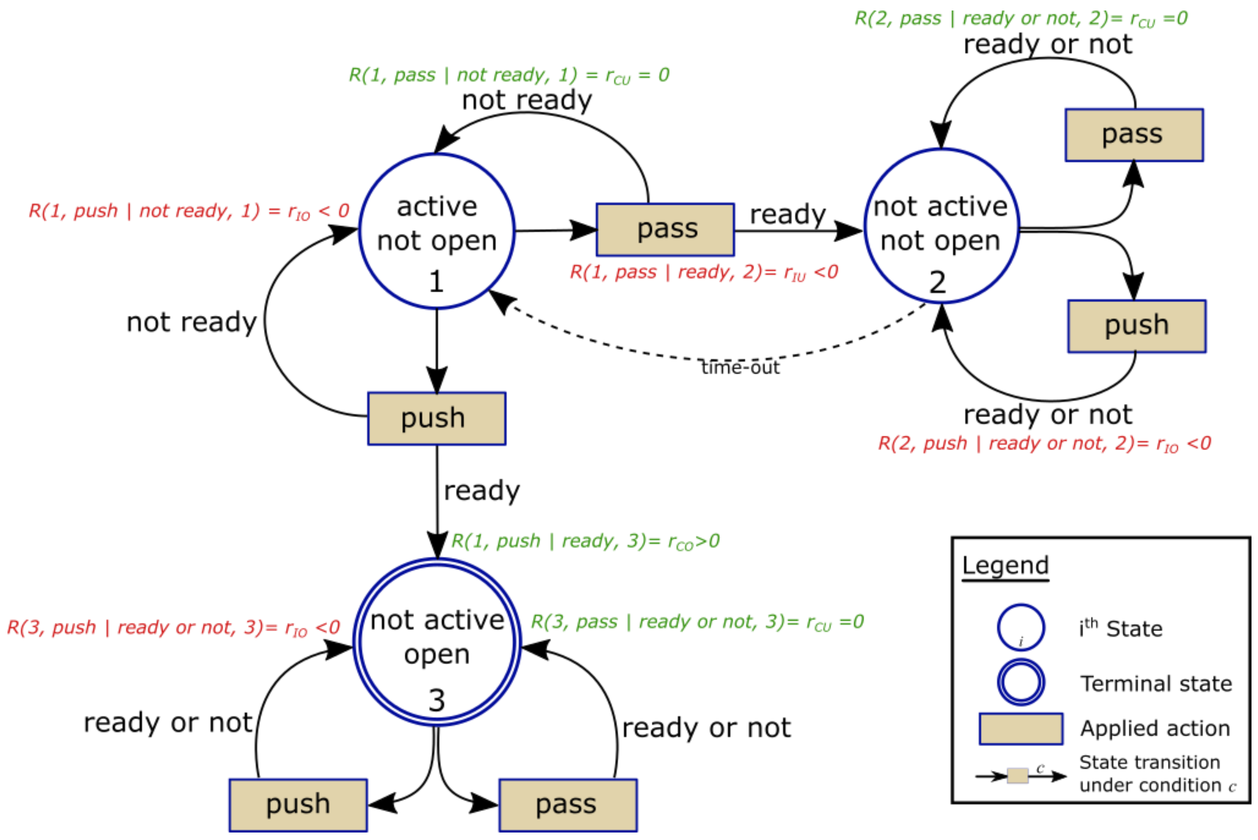
- Opening event-pattern
- Noise
  - Creation of event not associated to any chest
- Variability parameters
  - In event’s duration
  - In duration between events




```yaml
GENERAL:
  delay: 10
  noise: 0.3
INSTANTIATE:
  - name: e1
    type: A
    params:
      bg: blue
      fg: red
    duration:
      mu: 5
      sigma: 2
  - name: e2
    type: B
    params:
      bg: *
      fg: *
    duration:
      mu: 6
      sigma: 2

RELATION:
  -type: met_by
   events:
     -e1
     -e2
```

The Game: One Activity Based Markov Decision Process per chest



> A chest is or is not:

- ***Active***: The symbols of its code are visible
- ***Open***: The associated button must not to be pushed
- ***Ready***: The associated button has to be pushed

> MDP state is a triplet

- Observable Part(***activeness***, ***openness***)
- Context (***ready***)

> Actions

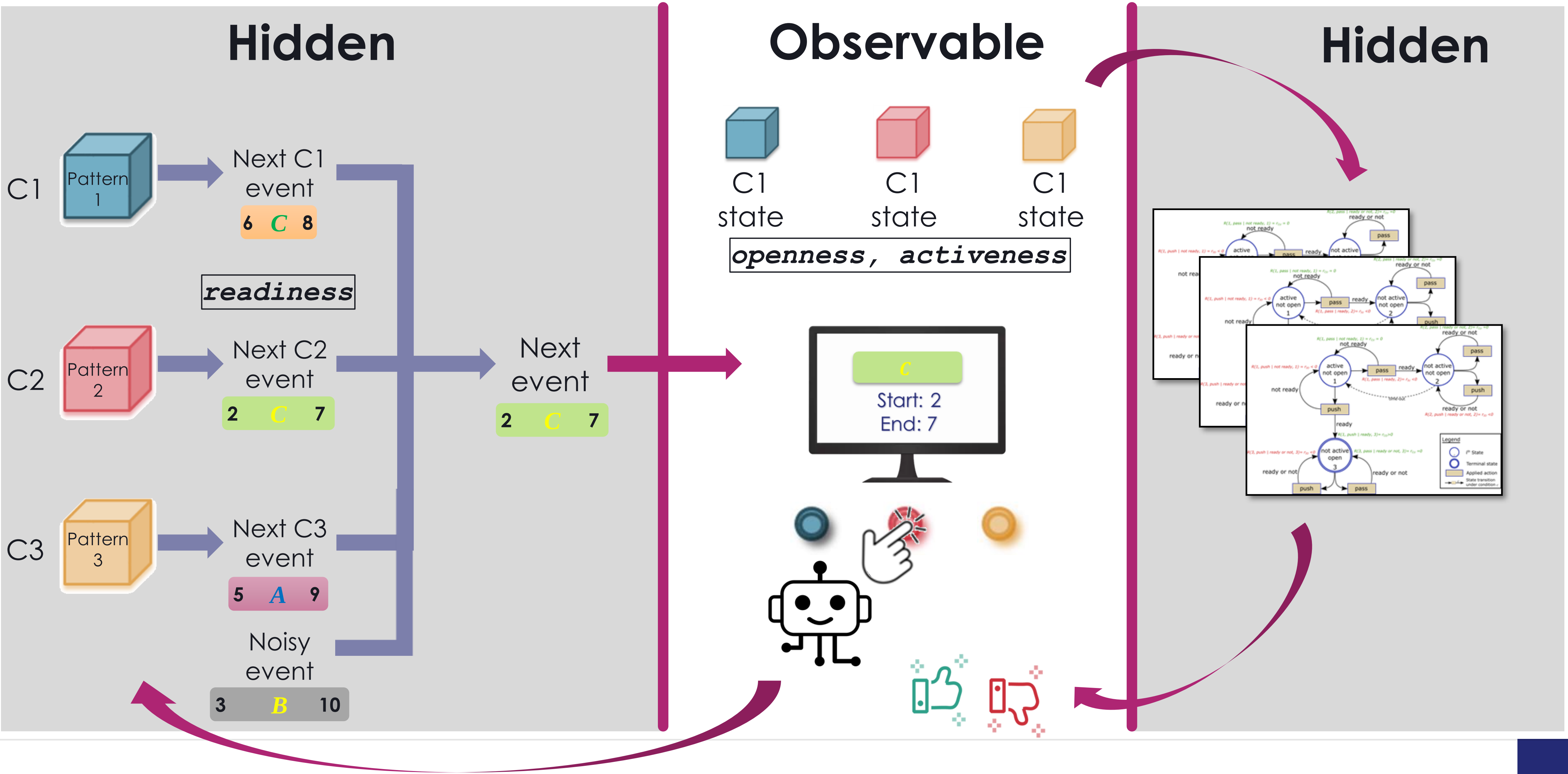
- **Push / Pass**

> But, it is partially observable

- ***Ready*** is latent
- Events history defines the context suitable to infer its value

The decision policy should come from the context, a.k.a. the history of events, NOT from the observable state of the MDP

The Game: Full view of the transition loop



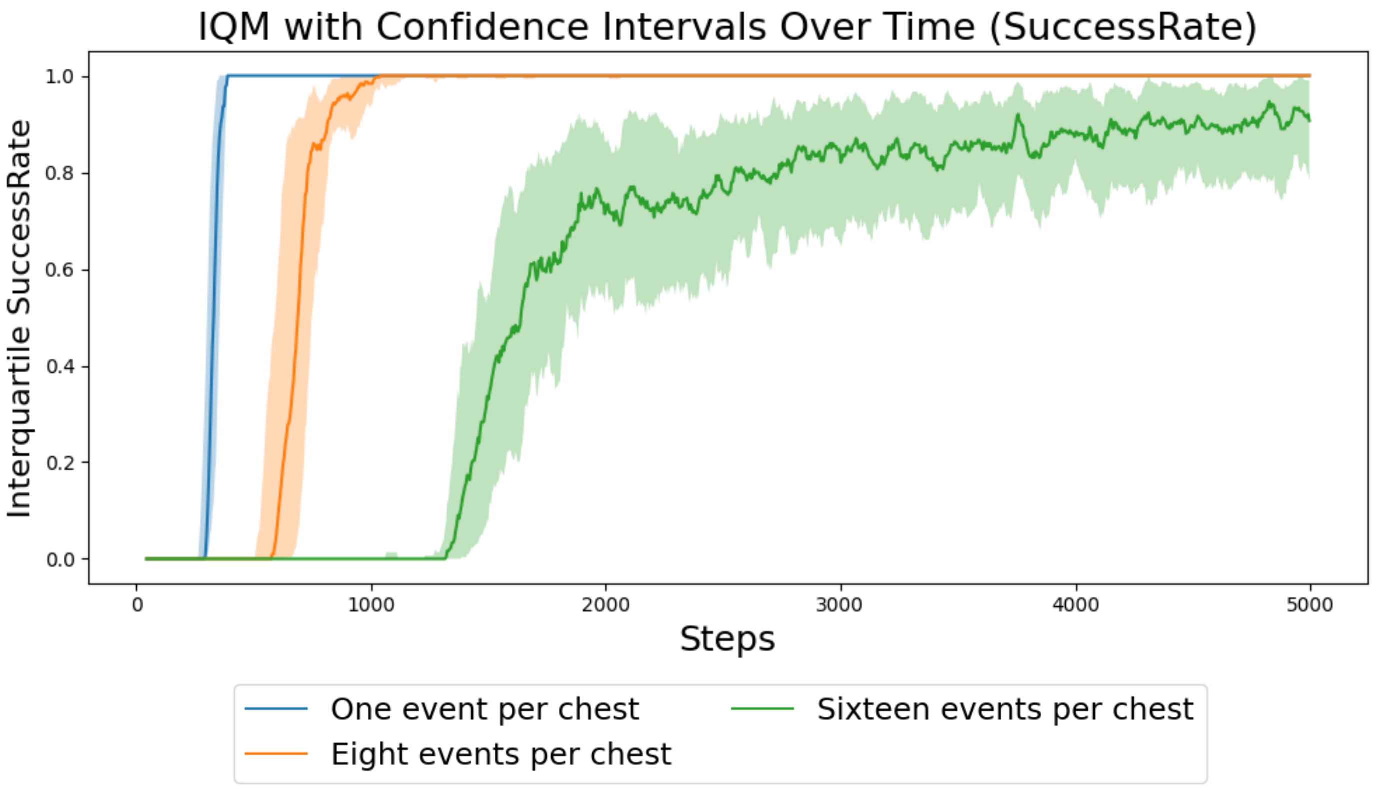
Validating environments of varying complexity using DTQN (Esslinger et al., 2022)

> Three environments for testing

- ***Each one has 5 chests***
- ***Varying number of events*** *per chest for each environment*
  - ***One event****: no history dependency*
  - ***Eight events:*** *medium history dependency*
  - ***Sixteen events:*** *high history dependency*
- ***Varying levels of noise*** *for each activity*
  - *from 0.1 to 0.3*

> *Results validate usage and adaptability*

- *Easily integrated with DTQN thanks to gym API*
- *Varying complexity show variation in success*
- *Longer activities and challenging temporal patterns lead to decline in performance*
- *Lack of interpretability impacts real-world applicability*



Next functionalities to come

> Integration of a discrete-event simulation framework

- Next event generation with discrete event systems simulation
- Handle more complex chests combination

> More heterogeneity in events

> Sequential dependency of chests

- Open a chest if and only if another one has been opened before

> More complex actions

- Multi-actions (more that one action at a time)
- Actions with attributes (e.g., pressing a button with a different duration or a different strength would change the impact)

THALES

Building a future we can all trust

OTC

OPEN THE CHESTS

https://github.com/ThalesGroup/open-the-chests

https://pypi.org/project/openthechests/

www.thalesgroup.com

References

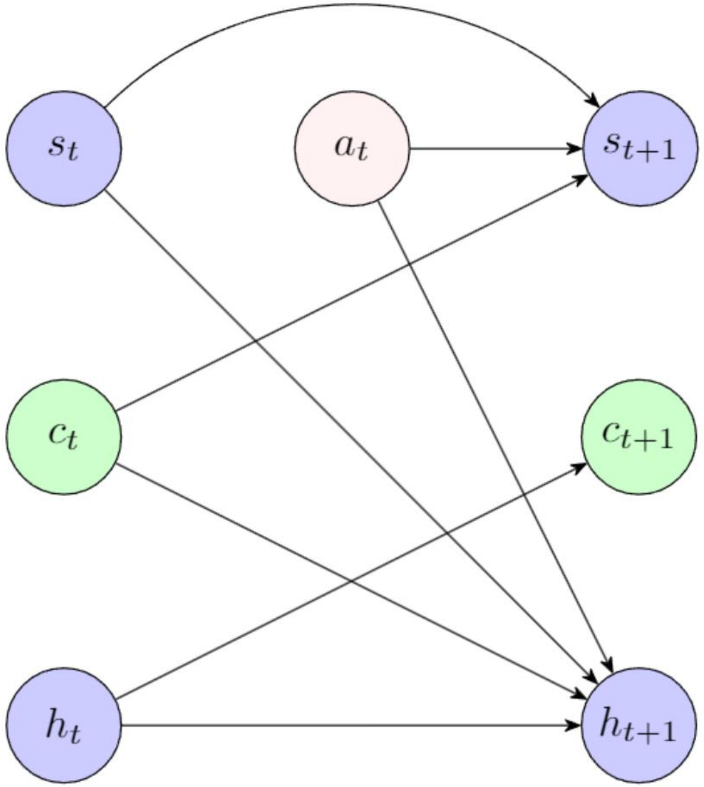
Landwehr, N. (2008, July). Modeling interleaved hidden processes. In Proceedings of the 25th international conference on Machine learning (pp. 520-527).
Tennenholtz, G., Merlis, N., Shani, L., Mladenov, M., & Boutilier, C. (2023, July). Reinforcement learning with history dependent dynamic contexts. In International Conference on Machine Learning (pp. 34011-34053). PMLR.
Boutilier, C., Dean, T., & Hanks, S. (1995, September). Planning under uncertainty: Structural assumptions and computational leverage. In Proceedings of the Second European Workshop on Planning (pp. 157-171).
Rachelson, E. (2009). Temporal markov decision problems (Doctoral dissertation, Université Paris 6).
Esslinger, K., Platt, R., & Amato, C. (2022). Deep transformer q-networks for partially observable reinforcement learning. arXiv preprint arXiv:2206.01078.

Representing Interaction through the AB-MDP (Activity-Based MDP)

> Interaction: Intervening with the environment at key moments when activities are present

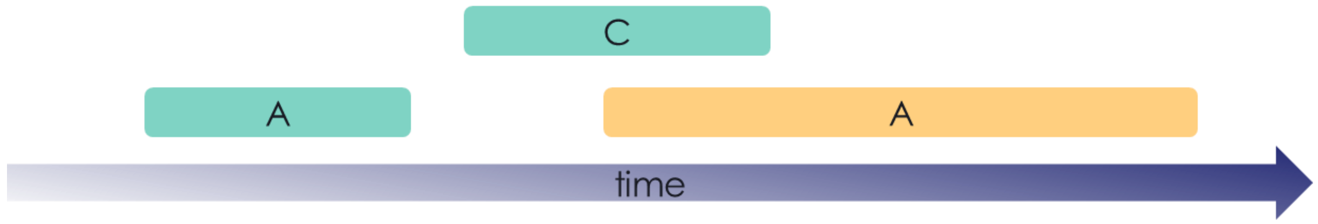
- $\boldsymbol{A}$ is the finite **action space**.
- $\boldsymbol{T}$ is a **context dependent transition function** defined as $T(s_t, a_t, c_t, s_{t+1}) = \Pr(s_{t+1} | s_t, a_t, c_t)$, influenced by the actions taken and the currently active activities.
- $\boldsymbol{\mathcal{R}}$ is the **reward function** defined as $\mathcal{R}(s,a,c) = r$, which evaluates the **success of actions in reacting to recognized activities**.
- $\boldsymbol{S}$ is the **observable space** of the environment, constituted by any **observable information outside of events**.
- $\boldsymbol{E}$ is the space of **event observations**.
- $\boldsymbol{C}$ is a **contextual vector** of size n. Each element $\boldsymbol{c_i}$ indicates the **completion status of the $\boldsymbol{i}$-th activity**, with

$$\boldsymbol{c_i} = \begin{cases} True, & \exists\{e_1, \dots, e_n\} \subset h_t \mid C_{A_i}(e_1, \dots, e_n) = True \\ False, & otherwise \end{cases}$$



Causal diagram depicting the dependencies in an AB-MDP. Green circles represent unobserved variables.

Generating Events using Activities as Memory Automata



Register

Allows to access and save value instances of emitted events.
$\{e_1 : val_{e_1}, e_2 : val_{e_2}, e_3 : val_{e\_3}\}$

Generate

At each step: First constrain all values for next event (symbol, attributes, interval) using filters and relations, then generate event using uniform distribution on these values.

1 → 2: $e_1 \sim Fa_1(e_1) \wedge e_1.t_{start} \geq t$ — Emit and Register

2 → 3: $e_2 \sim Fa_2(e_2) \wedge Fr_{1,2}(e_1, e_2) \wedge \mathcal{T}_{allen_{1,2}}(e_1, e_2)$ — Emit and Register

3 → 4: $e_3 \sim Fa_3(e_3) \wedge Fr_{1,3}(e_1, e_3) \wedge Fr_{2,3}(e_2, e_3) \wedge \mathcal{T}_{allen_{1,3}}(e_1, e_3) \wedge \mathcal{T}_{allen_{2,3}}(e_2, e_3)$ — Emit and Register

4 → 1: Wait for delay of t', empty Registers and save current time t

Representing Activity Recognition Conditions through filters and Temporal Relations

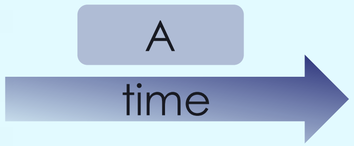
$$C_A(e_1, \dots, e_n) = \bigwedge_{i=1}^{n} Fa_i(e_i) \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} R_{temp}(e_i, e_j)$$

One Event

$$\begin{array}{cc} & e_1 \\ e_1 & \left(\begin{array}{c} F_A \\ = \end{array}\right) \end{array}$$

$$\Leftrightarrow$$

$$F_A(e_1)$$



Two Events

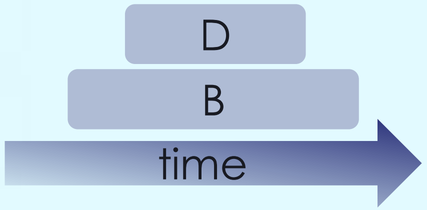
$$\begin{array}{cc} & \begin{array}{cc} e_1 & e_2 \end{array} \\ \begin{array}{c} e_1 \\ e_2 \end{array} & \left(\begin{array}{cc} F_B & F_D \\ = & d \\ \# & = \end{array}\right) \end{array}$$

$$\Leftrightarrow$$

$$F_B(e_1) \wedge F_D(e_2)$$

$$\wedge$$

$$R_d(e_1, e_2)$$



Three Events

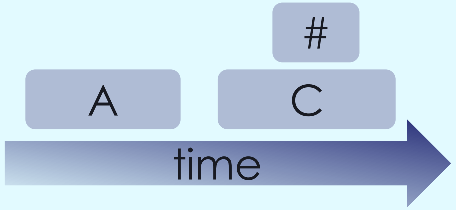
$$\begin{array}{cc} & \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{array}{c} e_1 \\ e_2 \\ e_3 \end{array} & \left(\begin{array}{ccc} F_A & \# & F_C \\ = & < & \# \\ \# & = & d \\ \# & \# & = \end{array}\right) \end{array}$$

$$\Leftrightarrow$$

$$F_A(e_2) \wedge F_C(e_3)$$

$$\wedge$$

$$R_<(e_1, e_2) \wedge R_d(e_2, e_3)$$



Example with two chests and 0.3 noise