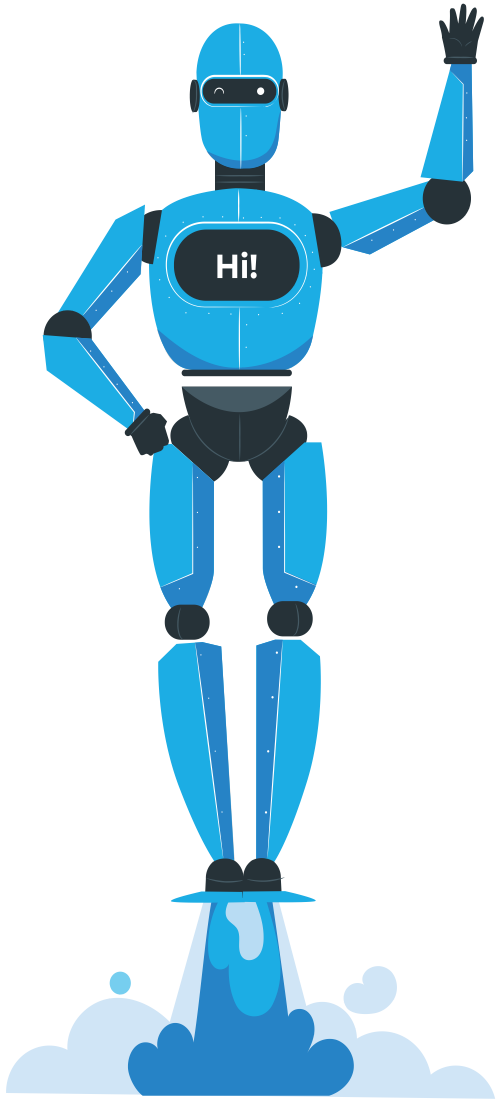
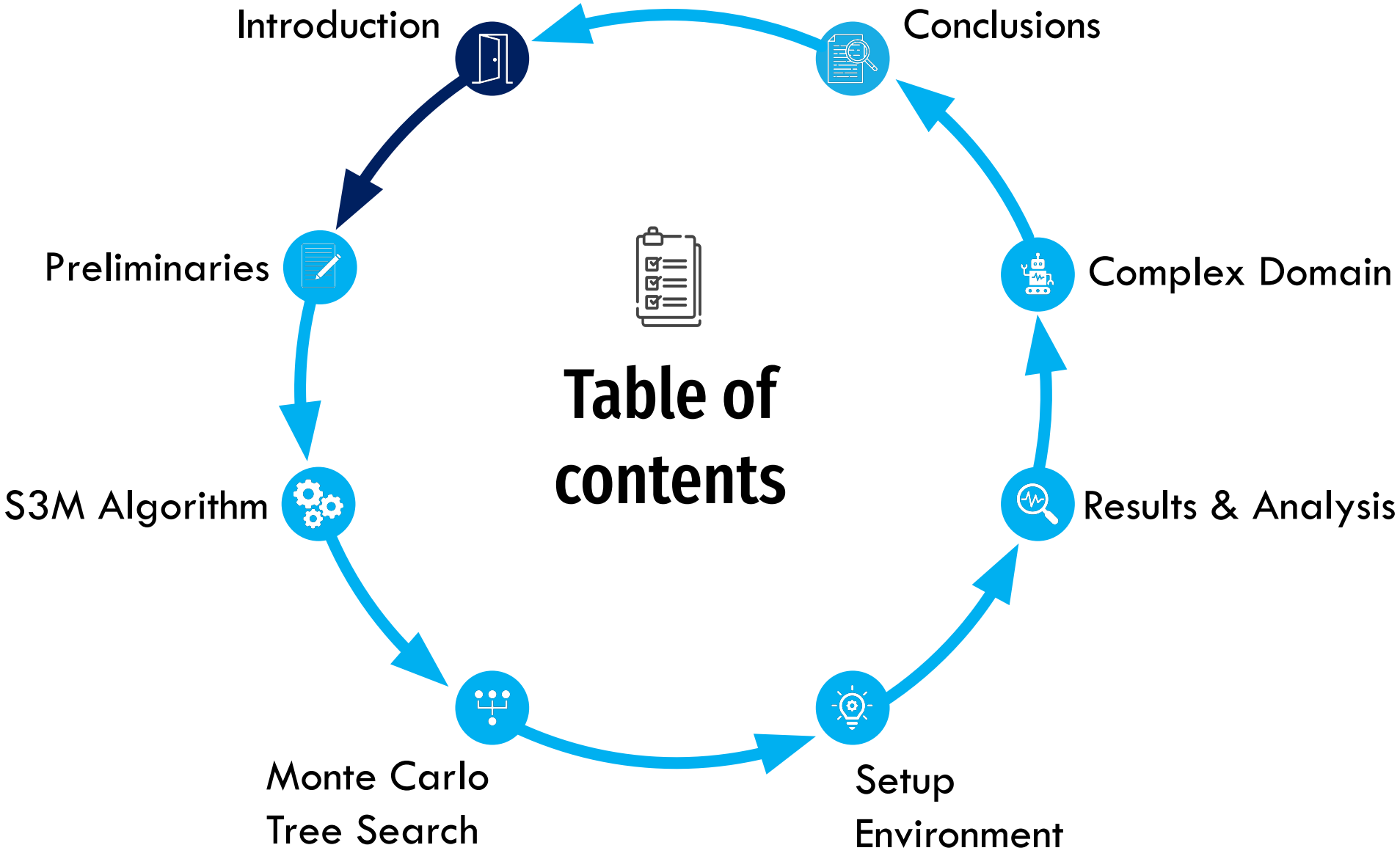
LEARNING AND SOLVING REGULAR DECISION PROCESSES

Tania Sari Bonaventura, Christian Marinoni,
Sveva Pepe & Simone Tedeschi

13TH SEPTEMBER
REASONING AGENTS




Introduction
Conclusions
Preliminaries
Complex Domain
Table of contents
S3M Algorithm
Results & Analysis
Monte Carlo Tree Search
Setup Environment

INTRODUCTION:
WHAT ARE RDPS?

Non-Markovian extension of **MDPs**

Fully observable, **non-Markovian** model

Next state and reward are stochastic functions of the entire **history** of the system

Dependence restricted to **regular functions**

WHAT ARE THE RECENT WORKS ON RDP?

Learning and Solving Regular Decision Processes
(Abadi and Brafman, 2020)

- Usage of a deterministic **Mealy Machine** to describe the RDP

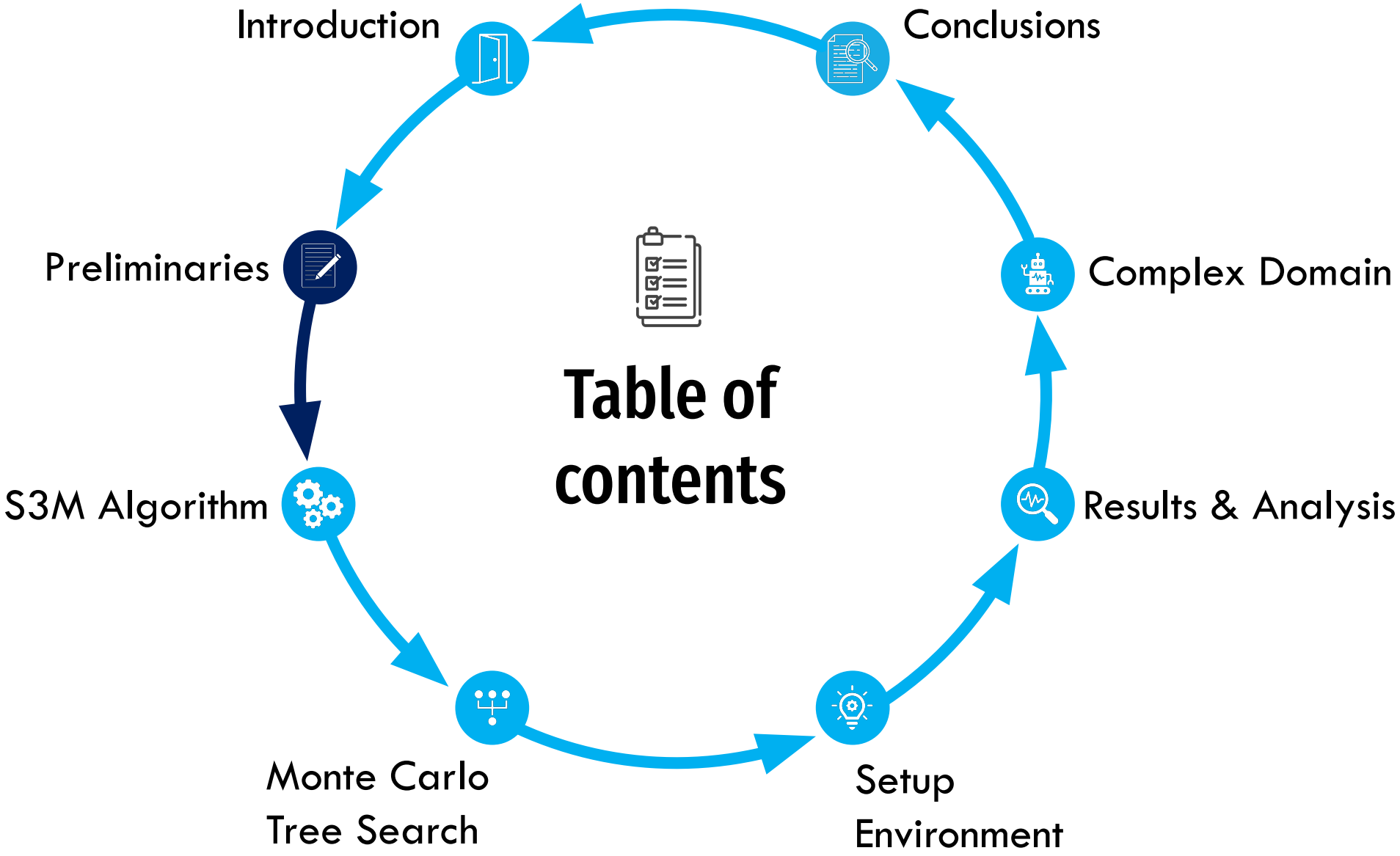
- First method for **learning RDPs from data**
  - combines **histories** with similar distributions
  - builds the **best model** from which the Mealy Machine is generated
  - uses Mealy Machine to generate the distributions needed to run **MCTS**

- Test the method on four **domains**:
  Rotating MAB, Malfunction MAB, Cheat MAB and RotatingMAZE

WHAT ARE THE RECENT WORKS ON RDP?

Efficient PAC Reinforcement Learning in Regular Decision Processes
(Ronca and De Giacomo, 2021)

- Learns a probabilistic deterministic finite automaton (**PDFA**) that represents RDP
- Computes an **intermediate policy** by solving the MDP obtained from the PDFA
  - final policy obtained by composing it with the transition function of the PDFA
- Algorithm ensures to reach an **ϵ-optimal policy** in **polynomial time**


Introduction
Conclusions
Preliminaries
Complex Domain
Table of contents
S3M Algorithm
Results & Analysis
Monte Carlo Tree Search
Setup Environment

PRELIMINARIES: MDP

$$< A, S, R, T, \gamma >$$

- A: set of **actions**
- S: **state** space
- R: $S \times A \times S \rightarrow \mathbb{R}$: **reward** function
- T: $S \times A \times S \rightarrow [0,1]$: **transition** function
- γ: **discount** factor

Markov assumption:

The outcome of an action is solely determined by the state in which it is performed

PRELIMINARIES: NMDP

$$< \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{R}, \boldsymbol{T}, \boldsymbol{\gamma} >$$

- Does not verify the Markov assumption
- ⇒ The outcome of the current action depends on the whole **history.**
- $\boldsymbol{T}: S^* \times A \times S \rightarrow [0,1]$: **transition** function
- $\boldsymbol{R}: S^* \times A \times S \rightarrow \mathbb{R}$: **reward** function
- $\boldsymbol{D}: S^* \times A \times S \times R \rightarrow [0,1]$: **dynamics** function

PRELIMINARIES: RDP

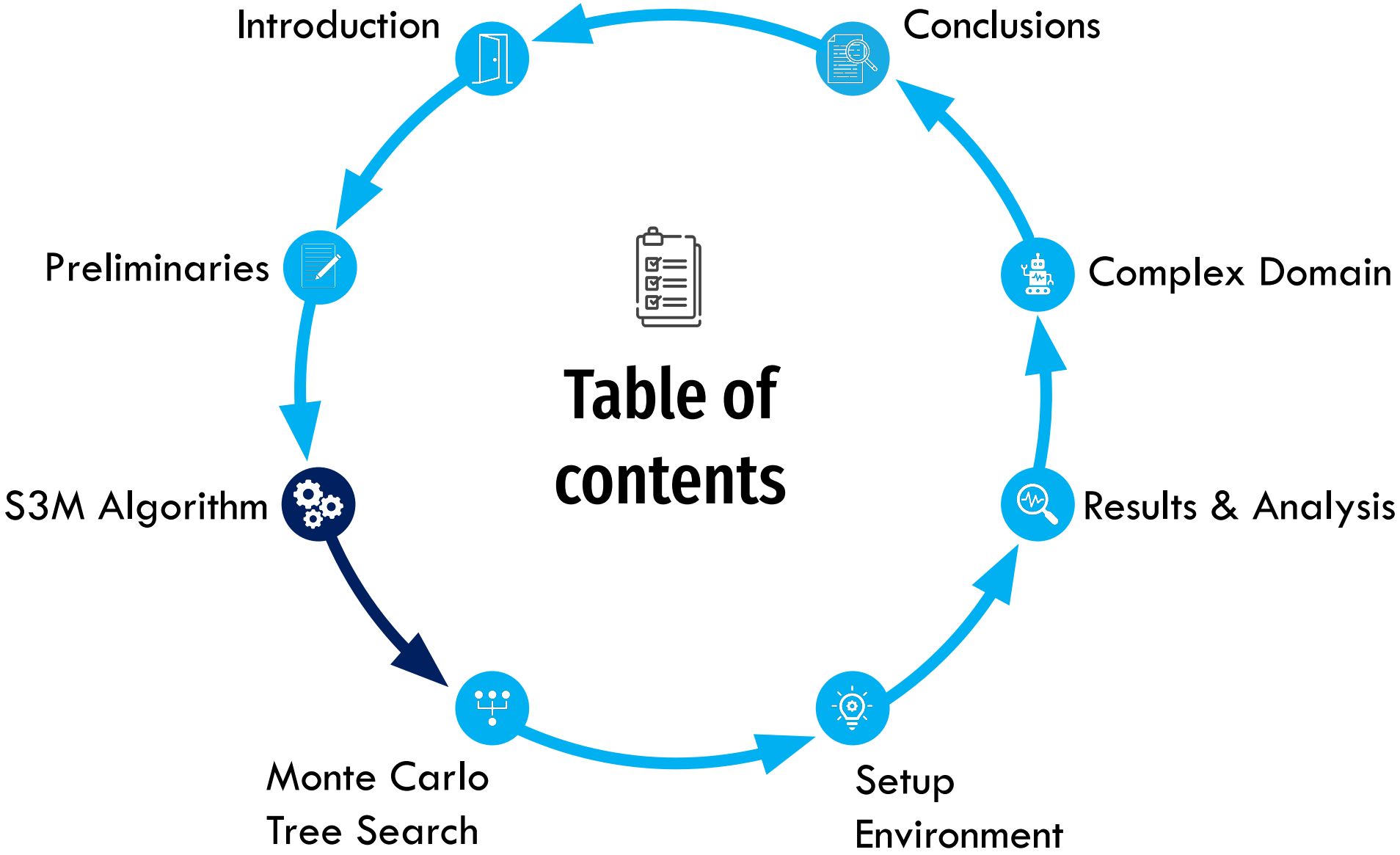
$$< \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{R}, \boldsymbol{T}, \boldsymbol{\gamma} >$$

- NMDP: dependence on **history** restricted to **RF**
- Transition and reward functions represented as **finite transducers:**
  - $\boldsymbol{T_h}: A \times S \rightarrow [0,1]$: transducer induced by *T* when its first argument is *h*
  - $\boldsymbol{R_h}: A \times S \times R \rightarrow [0,1]$: transducer induced by *R* when its first argument is *h*

PRELIMINARIES: MEALY MACHINE

$$< \boldsymbol{S}, \boldsymbol{s_0}, \Sigma, \Lambda, \boldsymbol{T}, \boldsymbol{G} >$$

- $\boldsymbol{S}$: **state** space
- $\boldsymbol{s_0}$: **starting** state
- Σ: **input** alphabet
- Λ: **output** alphabet
- $\boldsymbol{T}: Q \times \Sigma \rightarrow Q$: deterministic **transition** function
- $\boldsymbol{G}: Q \times \Sigma \rightarrow \Lambda$: **output** function


Introduction
Conclusions
Preliminaries
Complex Domain
Table of contents
S3M Algorithm
Results & Analysis
Monte Carlo Tree Search
Setup Environment

SAMPLE MERGE MEALY MODEL (S3M)

- ***Sampling***: generates traces from the RDP by interacting with the environment
- ***Base distribution***: it associates a distribution over the states to each history derived from the traces
- ***Merger***: it merges the distributions of multiple histories based on the Kullback-Leibler divergence
- ***Loss function***: it allows to select the best model, penalizing the ones with the few merges and too high support
- ***Mealy Machine***: it generates the Mealy Machine from the computed model

Algorithm 1 Sample Merge Mealy Model (S3M)

Input: $domain$
Parameter: $min\_samples$
Output: M

1: Initialize state space of M to RDP state space S_{RDP}
2: **repeat**
3: Set $S = sample(domain)$.
4: Set $Tr = base\_distributions(S, min\_samples)$
5: $best\_loss = \infty$
6: **for** ϵ *in* $possible\_epsilons$ **do**
7: $Tr' = merger(Tr, \epsilon)$
8: $loss = calc\_loss(Tr', S)$
9: **if** $loss < best\_loss$ **then**
10: $Tr = Tr'$
11: $best\_loss = loss$
12: **end if**
13: **end for**
14: $Me = mealy\_generator$
15: Set $M = \langle P, A, S_{RDP} \times S_{Me}, Tr, R, (s_{0_M}, s_{0_{Me}}) \rangle$
16: **until** Max_Iterations
17: **return** M

SAMPLING

🎯 Generate **sample traces**

Pure Exploration

- Encourage the exploration towards unseen regions of the environment
- $P(a|s) = \frac{f(a,s)}{\sum_a f(a,s)}$
- $f(a,s) = 1 - \frac{n(a,s)}{\sum_a n(a,s)}$

Smart Sampling

- ϵ-greedy approach which selects the greedy action through Q-learning with probability 1- ϵ
- performs pure exploration as explained above with the remaining probability

BASE DISTRIBUTION

Given a trace, we just associate to its history a distribution over the observations P(o|h)

- **History**: $h=(o_1\ a_1,o_2\ a_2,\ldots,o_n\ a_n)$
- **Trace**: ho'
- **Set Propositions** affected by action a: Ph,(o,a)
- Compute the empirical post-action distribution over for history h and observation o
- ⟶ our work does not use any proposition

MERGER

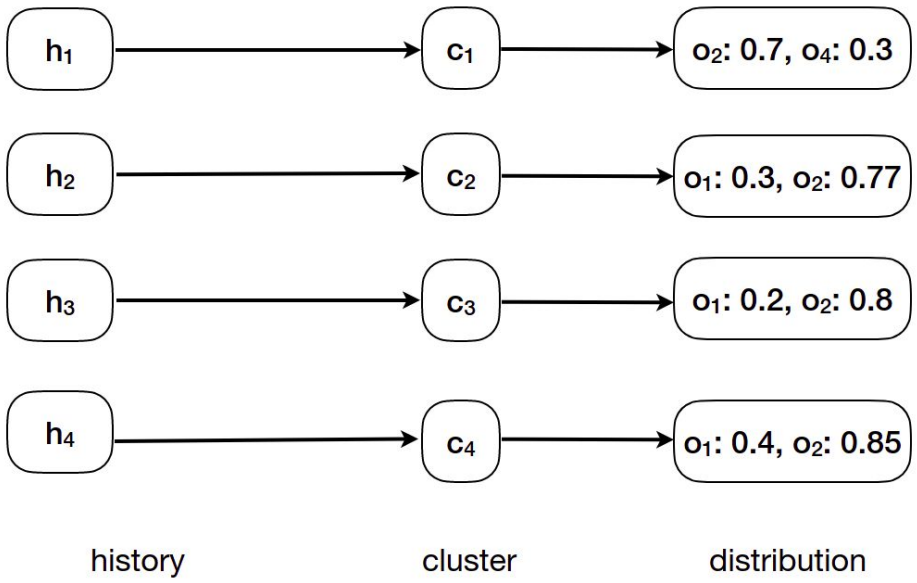
Merge histories with **similar distributions** and **same support**

Cluster histories with **Kullback-Leibler (KL)** divergence

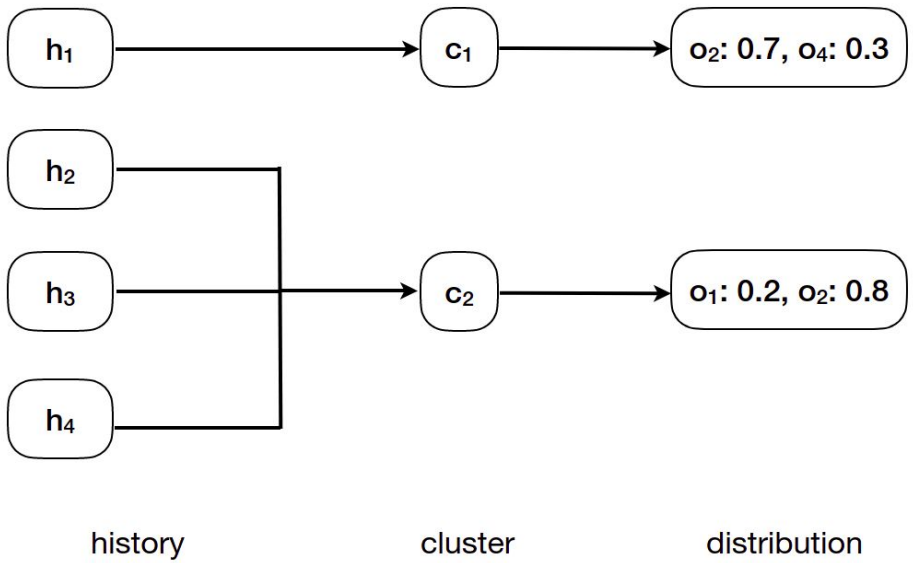
- Each history is associated to a cluster which encapsulates its distribution
- Several clusters will be merged after the merging operation
  - Several histories referring to the same distribution
  - The number of clusters will be fewer than initial one

MERGER

Before



After



MERGER

- Each cluster has a weight w: the number of samples used to create it.
  - Considering two clusters with distributions P_1 and P_2 and weights w_1 and w_2, the merging procedure will follow two different paths:

$w_1 \geq w_2 \geq min\_sample$
$D_{KL}(P_1, P_2) \leq \epsilon$
$w = w_1 + w_2$
$P(\cdot) = (1/w)[w_1 \cdot P_1(\cdot) + w_2 \cdot P_2(\cdot)]$

$w < min\_sample$
Same support: same set of observations
Smallest KL divergence

LOSS FUNCTION

🎯 Select the model that has the **best performance**

$$loss = -\sum_{h \in H} log\left(P(h|Tr(h)) + \lambda \cdot log\left(\sum_{\pi \in \Pi} |supp(\pi)|\right)\right)$$

$$P(h|Tr(h)) = \prod_{i=1}^{n} P(o_i|o_1 a_1 \dots o_{i-1} a_{i-1}; Tr(h))$$

- First term: likelihood of the histories given the given model Tr(h)
- Second term depends on the support of the observations' distributions

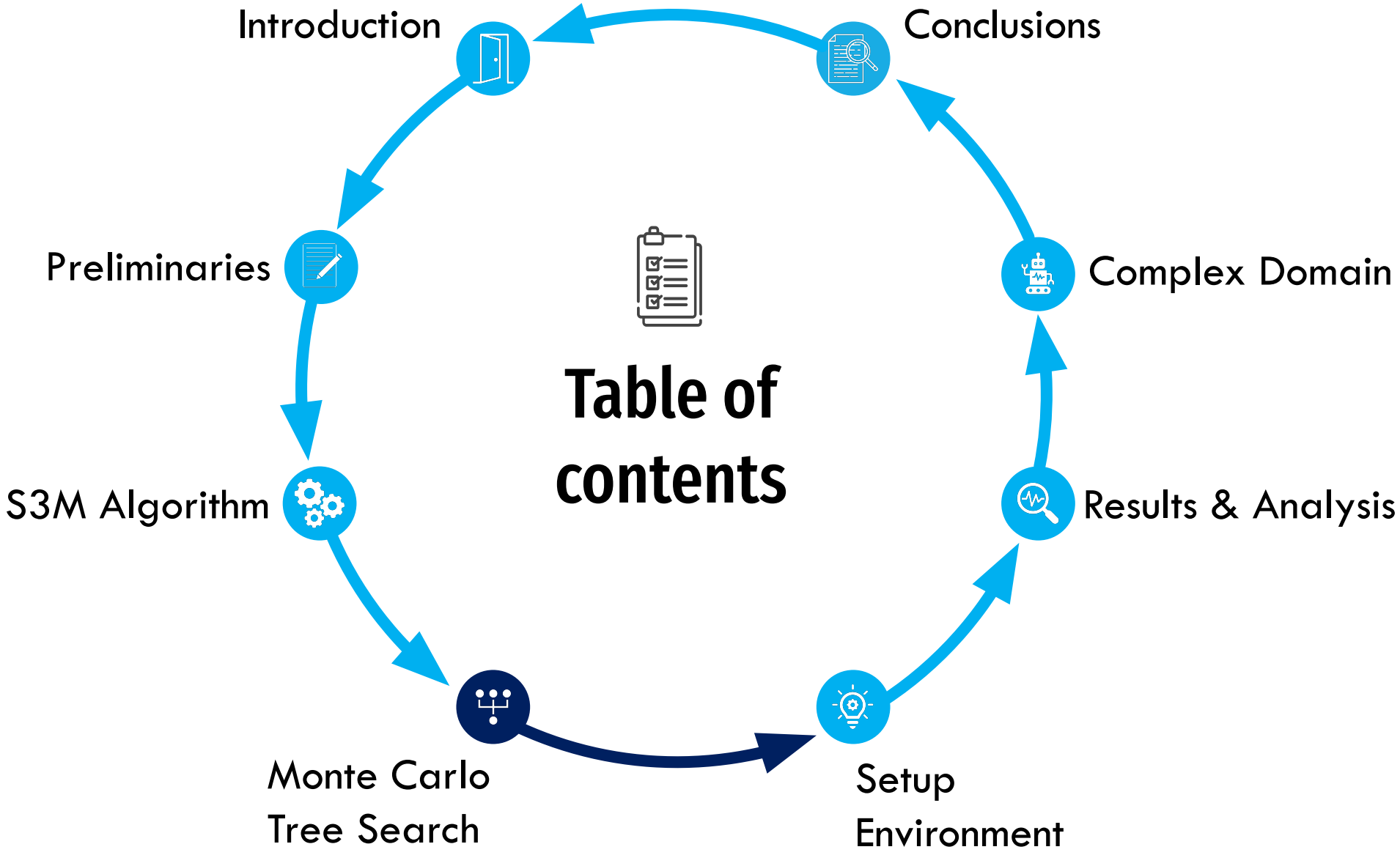
💡 Penalize models with **few merges** and too **high support**

MEALY MACHINE = <S, s_0, Σ, Λ, T, G>

🎯 Achieve a MDP = <SRDP x SMe, A, Tr, R, (s_0,s_0Me) >
that can be solved using UCT, an MCTS algorithm

```
19 4
1 1 o2a2/0
1 2 o2a2/0 o1a1/1
1 3 o2a2/0 o1a1/1 o2a2/0
1 2 o2a2/0 o1a2/0
1 3 o2a2/0 o1a2/0 o1a1/1
1 4 o2a2/0 o1a2/0 o1a1/1 o2a2/0
1 5 o2a2/0 o1a2/0 o1a1/1 o2a2/0 o1a1/1
1 4 o2a2/0 o1a1/1 o2a2/0 o1a1/1
1 5 o2a2/0 o1a1/1 o2a2/0 o1a1/1 o2a2/0
1 3 o2a2/0 o1a2/0 o1a2/0
1 4 o2a2/0 o1a2/0 o1a2/0 o1a2/0
1 5 o2a2/0 o1a2/0 o1a2/0 o1a2/0 o1a2/0
1 4 o2a2/0 o1a2/0 o1a2/0 o1a1/1
1 5 o2a2/0 o1a2/0 o1a2/0 o1a1/1 o2a2/0
1 5 o2a2/0 o1a2/0 o1a1/1 o2a2/0 o1a2/0
1 4 o2a2/0 o1a1/1 o2a2/0 o1a2/0
1 5 o2a2/0 o1a1/1 o2a2/0 o1a2/0 o1a2/0
1 5 o2a2/0 o1a1/1 o2a2/0 o1a2/0 o1a1/1
1 5 o2a2/0 o1a2/0 o1a2/0 o1a2/0 o1a1/1
```

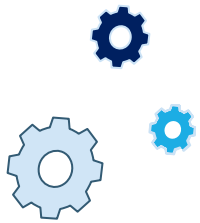
- *flexfringe* library to learn the Mealy Machine from data
- Every history of our sample is associated with a cluster index to which it belongs
- **First line:** the number of traces and the alphabet size
- From second row:
  - digit sample accept, length of the trace and pair obs-action/cluster_id

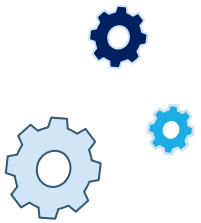

Introduction
Conclusions
Preliminaries
Complex Domain
S3M Algorithm
Results & Analysis
Monte Carlo
Tree Search
Setup
Environment
Table of contents

MONTE CARLO TREE SEARCH (MCTS)

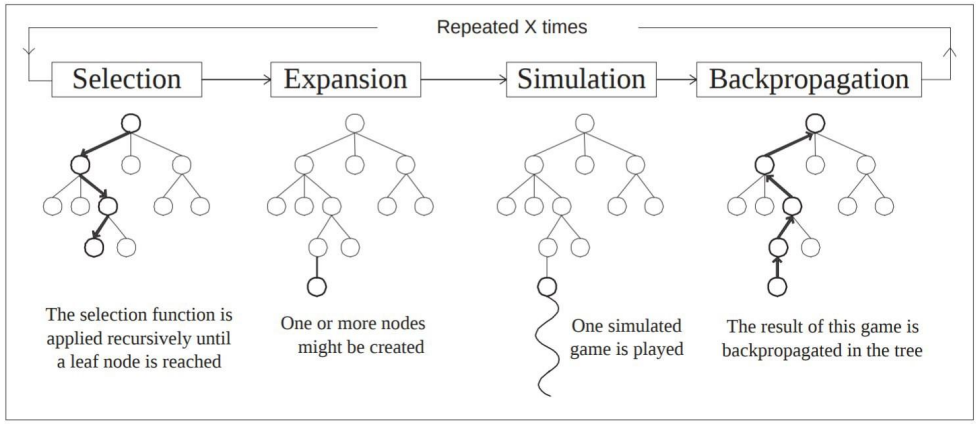
Solve Markov Decision Process

- Uses Monte Carlo simulation
- Pays more attention to nodes that are more promising





MONTE CARLO TREE SEARCH (MCTS)



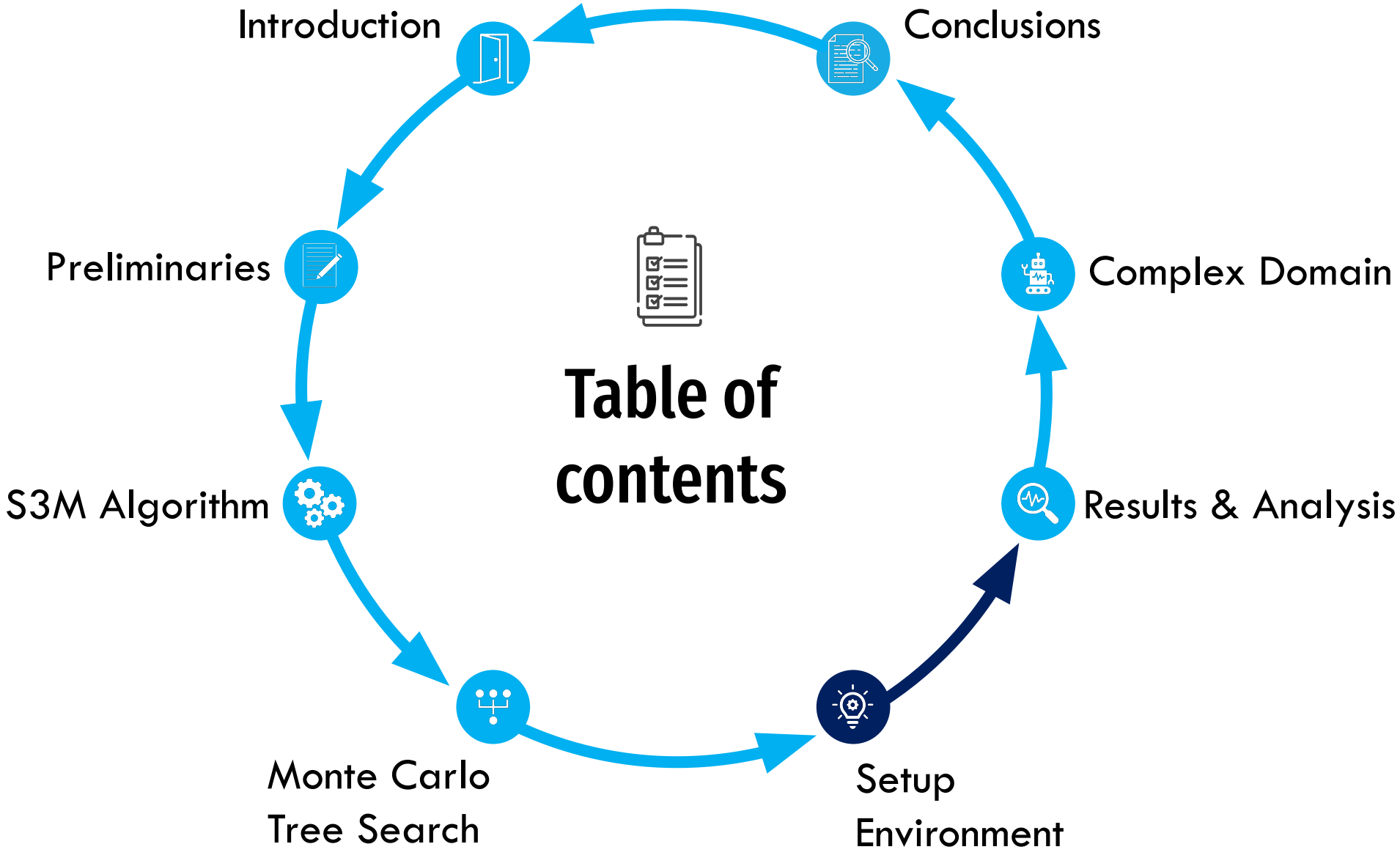
1. **Selection:** keeps selecting the *best* child nodes until leaf node is reached
2. **Expansion:** randomly picks an unexplored node from the leaf node
3. **Simulation:** simulates an entire episode from the selected node by picking a sequence of random actions
4. **Backpropagation:** evaluates the reached state and updates the win score of each node based on the final state

MONTE CARLO TREE SEARCH (MCTS)

- use *Upper Confidence bounds* applied to Trees (UCT) which uses the UCB1 criterion to select the best child

$$UCB1 = \frac{w_i}{n_i} + c \cdot \sqrt{\frac{2 \cdot \log N}{n_i}}$$

- *wi* the number of wins of node i, *ni* the number of visits, c is the exploration parameter and *N* the is the number of visits of parent
- First component: **exploitation** ⟶ as it is high for moves with high average win ratio
- Second component: **exploration** ⟶ since it is high for moves with few simulations
- MCTS estimates
  - be *unreliable* at the start of a search
  - converge to more reliable estimates given sufficient time
  - perfect estimates given infinite time


Introduction
Conclusions
Preliminaries
Complex Domain
Table of contents
S3M Algorithm
Results & Analysis
Monte Carlo Tree Search
Setup Environment

SETUP ENVIRONMENT

- **Implementation:** Python
- **Library:** flexfringe ⟶ generate MM

- Evaluation on 2 domains families:
  - *Multi-Arm Bandit (MAB) Domains:* **Rotating MAB, Malfunction MAB** and **Cheat MAB**
    - Simplest class of Reinforcement Learning (RL) domains -- standard MAB state-less
    - Extends Non-Markovian MAB by making the probability of receiving a (fixed-size) reward depend on the entire history of previous actions
    - n=2, a two-state RDP, where each state tells whether a reward was received or not
  - *Maze:* **Rotating Maze**
    - N x N grid, where the agent starts in a fixed position and needs to reach a designated location to receive a final reward

MAB

Rotating MAB: each action has a probability of winning a reward. When the agent receives a reward, this probability shifts right ⟶ the winning probability of each arm depends on the entire history, via a regular function.

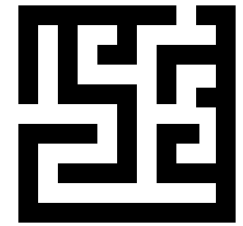
Malfunction MAB: after one action is executed k times, the corresponding arm becomes out of service ⟶ its winning probability lowers to zero for one turn.

Cheat MAB: there exists a sequence of actions that will guarantee to reach the reward with probability 1, regardless of the chosen action.

MAZE

Rotating Maze:

- Agent can do four actions (↑,↓,←,→)
- Action succeed 90% of the time, while in the rest 10 % it moves in the opposite direction, if feasible
- Effects of the actions are made a regular function of the history by changing the agent's orientation by 90 degrees every three actions ⟶ difficult to solve



PARAMETERS

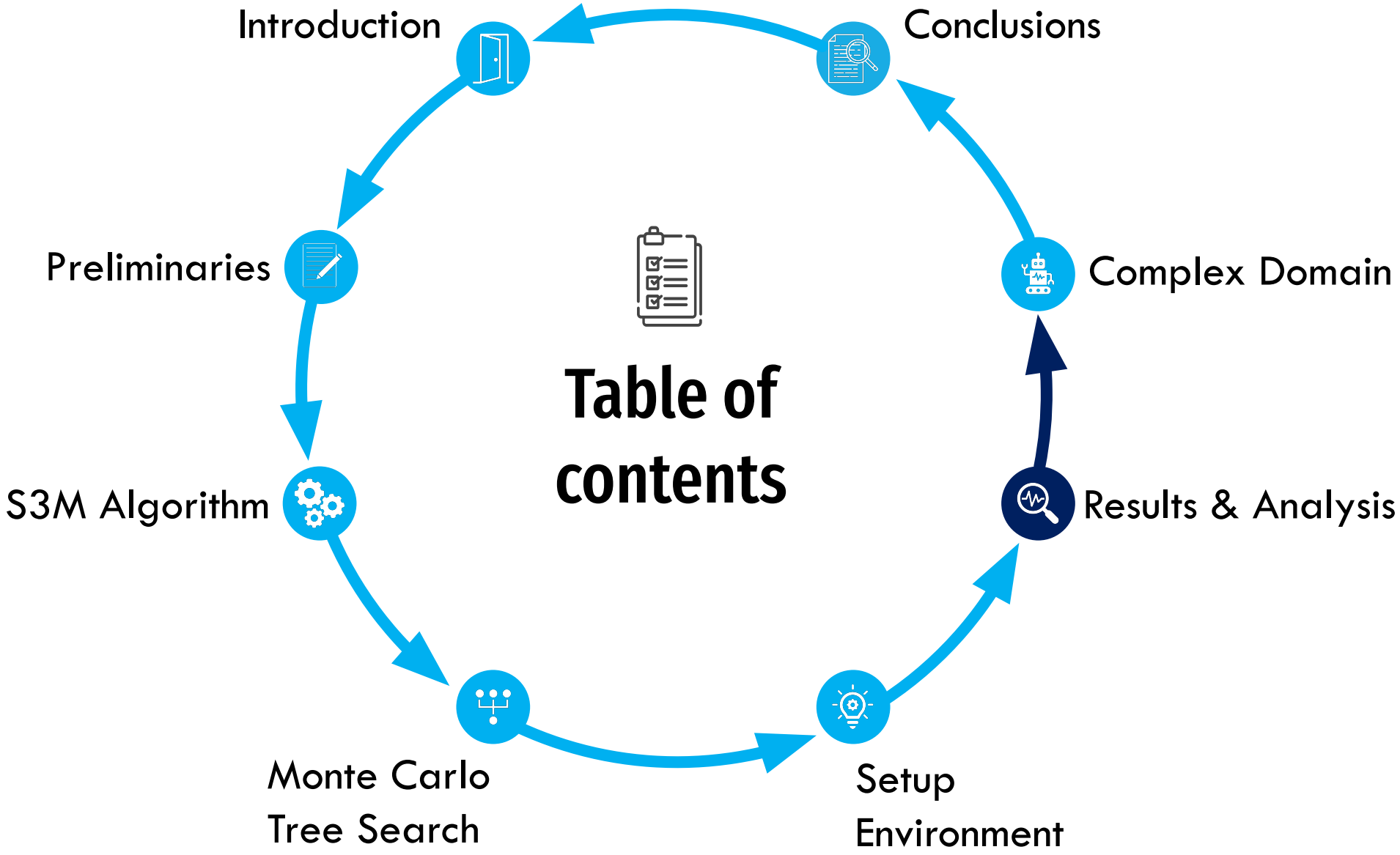
MAB

- 2 arms/actions
- Winning probabilities
  - Rotating MAB: (0.9,0.2)
  - Malfunction MAB: (0.2,0.2)
  - Cheat MAB: (0.8,0.2)

MAZE

- 4x4 maze
- Goal 5 steps away from initial position

Domain	min_sample	max_iterations	λ	ϵ	UCT iterations	UCT trials	episodes
RotatingMAB	100	500	100	[0.52, 1.04, 1.56, 2.08]	160000	30	10
MalfunctionMAB	10	100	100	[0.52, 1.04, 1.56, 2.08]	160000	30	10
CheatMAB	5	100	100	[0.52, 1.04, 1.56, 2.08]	160000	30	10
RotatingMaze	3	30	100	[0.52, 1.04, 1.56, 2.08]	140000	30	15


Introduction
Conclusions
Preliminaries
Complex Domain
Table of contents
S3M Algorithm
Results & Analysis
Monte Carlo Tree Search
Setup Environment

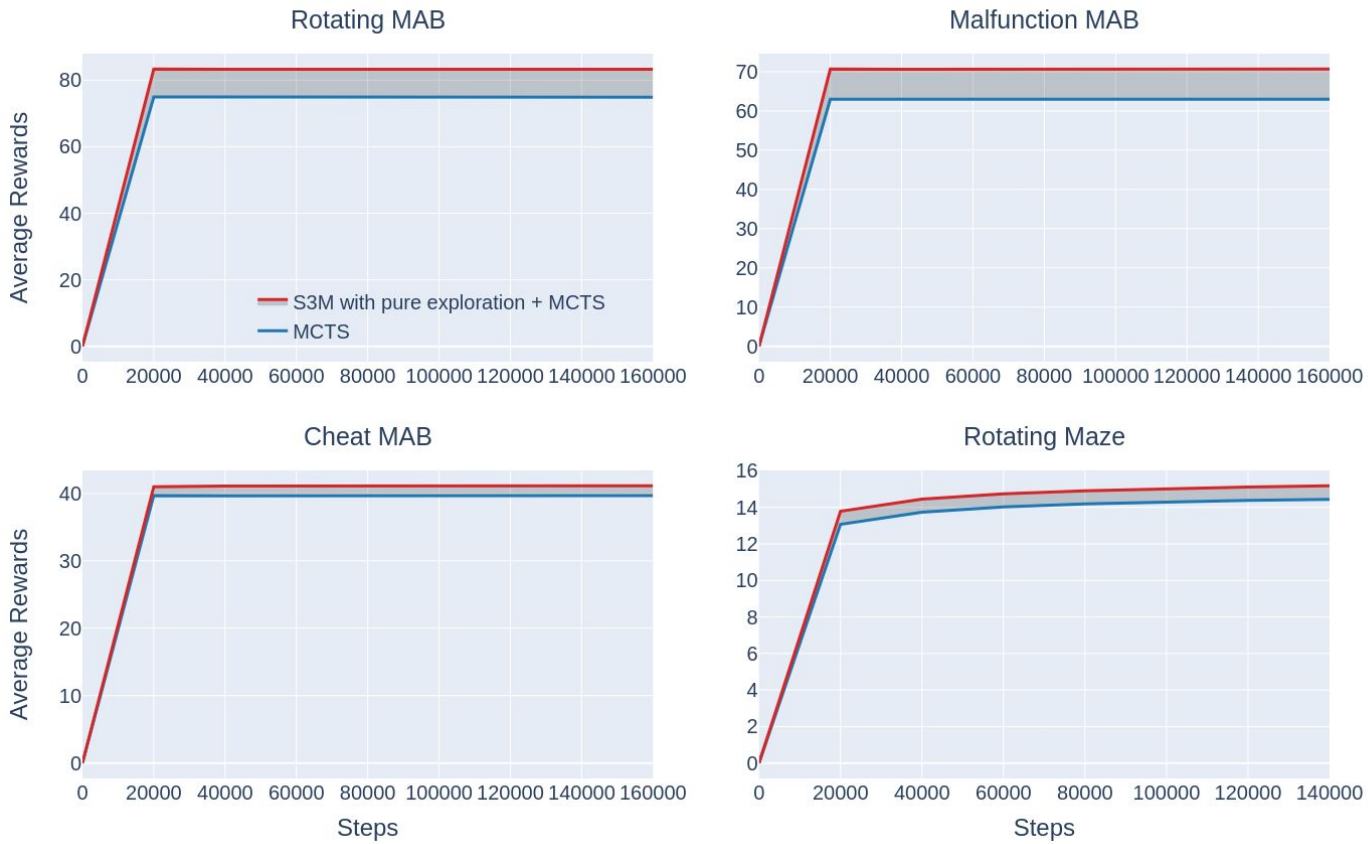
RESULTS

Compare two setups for each domain:

I. only MCTS

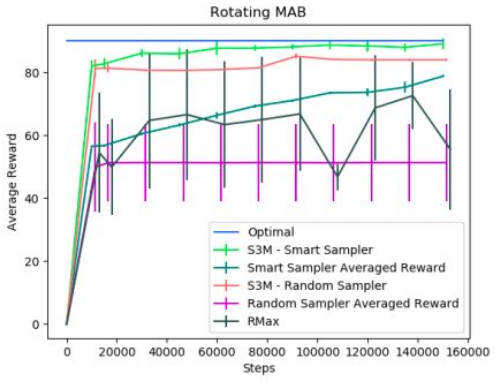
II. S3M algorithm with pure exploration policy + MCTS ⟶ verify the effectiveness of the S3M algorithm

- **30 trials**
- **MAB:**10 steps long
- **Maze:**15 steps long

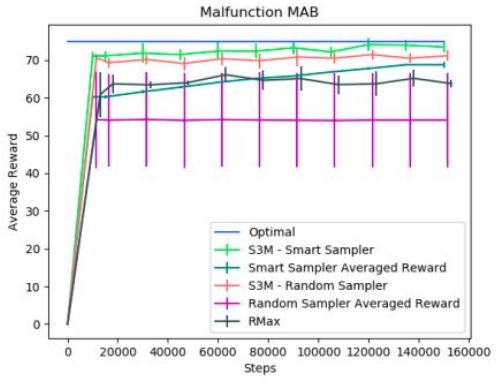


COMPARISON

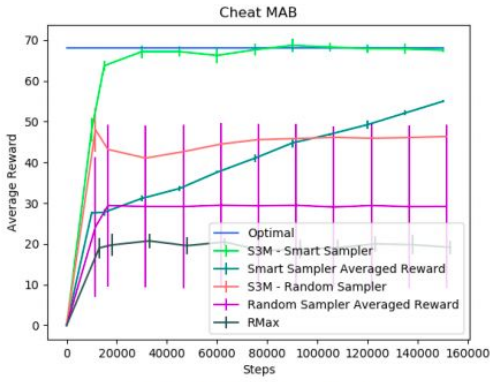
Results by Abadi and Brafman



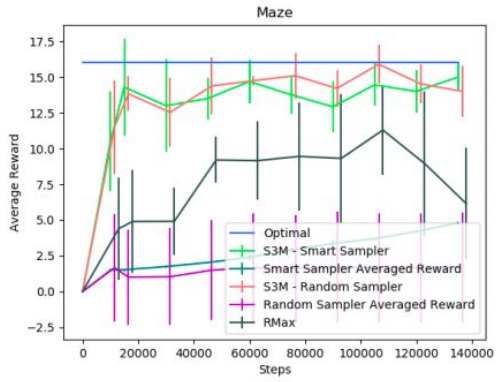
(a) Rotating MAB Domain Results



(b) Malfunction MAB Domain Results

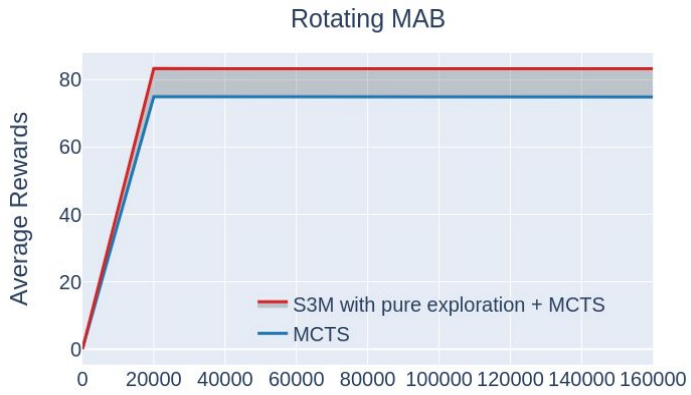


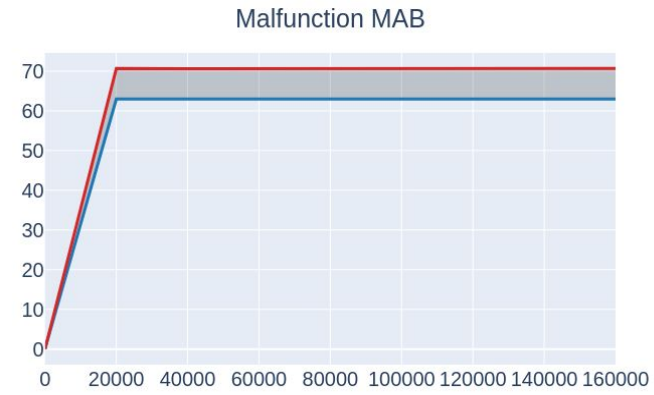
(c) Cheat MAB Domain Results

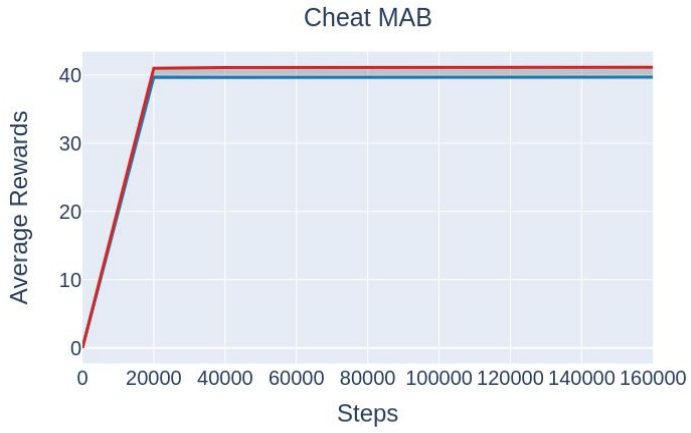


(d) Maze Domain Results

Our results

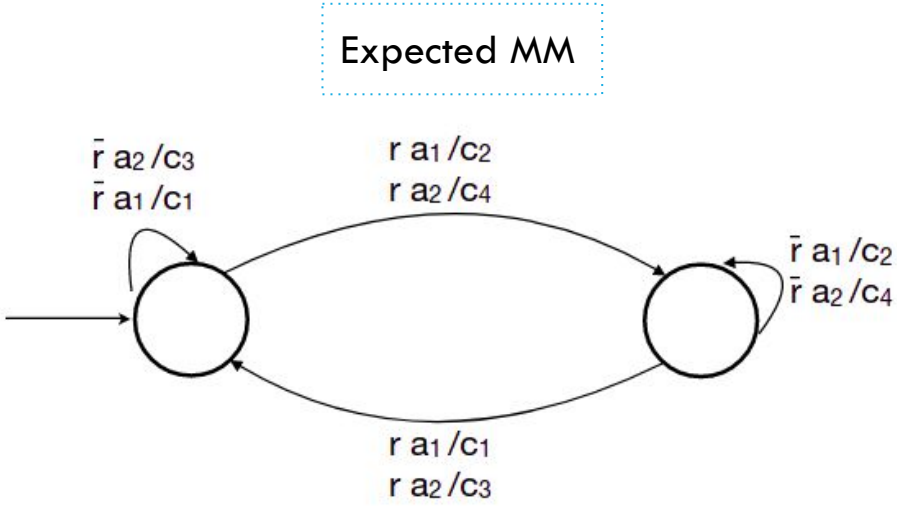






ANALYSIS ROTATING MAB (1/3)

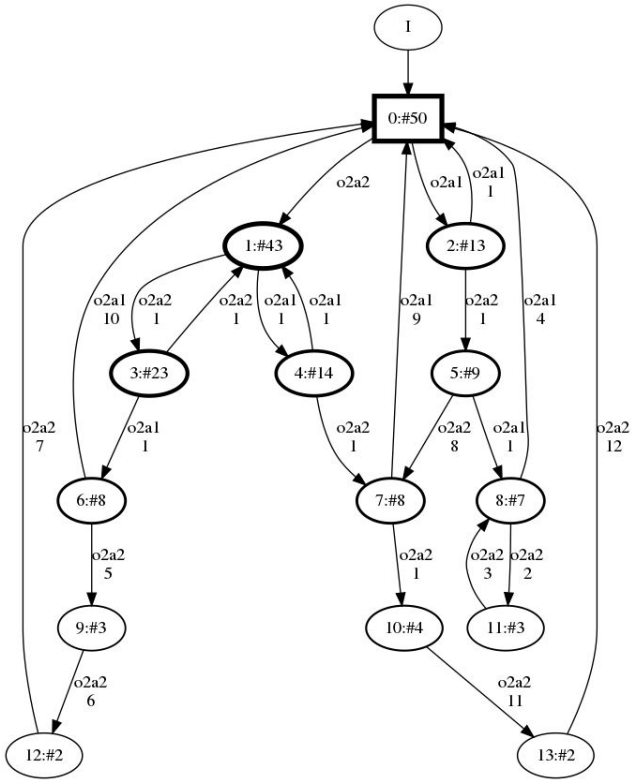
- Focus on the Rotating MAB
  - Two states: (0,) and (1,)
  - Expected MM: two states connected by edges
    - each edges ⟶ an observation, action chosen and cluster (output)
    - $\bar{r}$ ⟶ Agent did not receive a reward
    - r ⟶ Agent did receive a reward
      movement to the other state

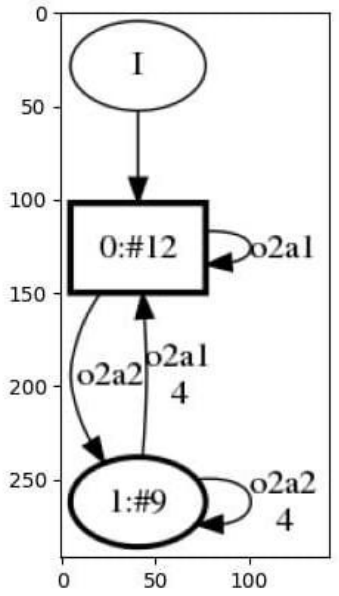


- Four clusters:
  1. c1: probability of observing a reward is 0.9
  2. c2: probability of observing a reward is 0.1
  3. c3: probability of observing a reward is 0.2
  4. c4: probability of observing a reward is 0.8

ANALYSIS ROTATING MAB (2/3)

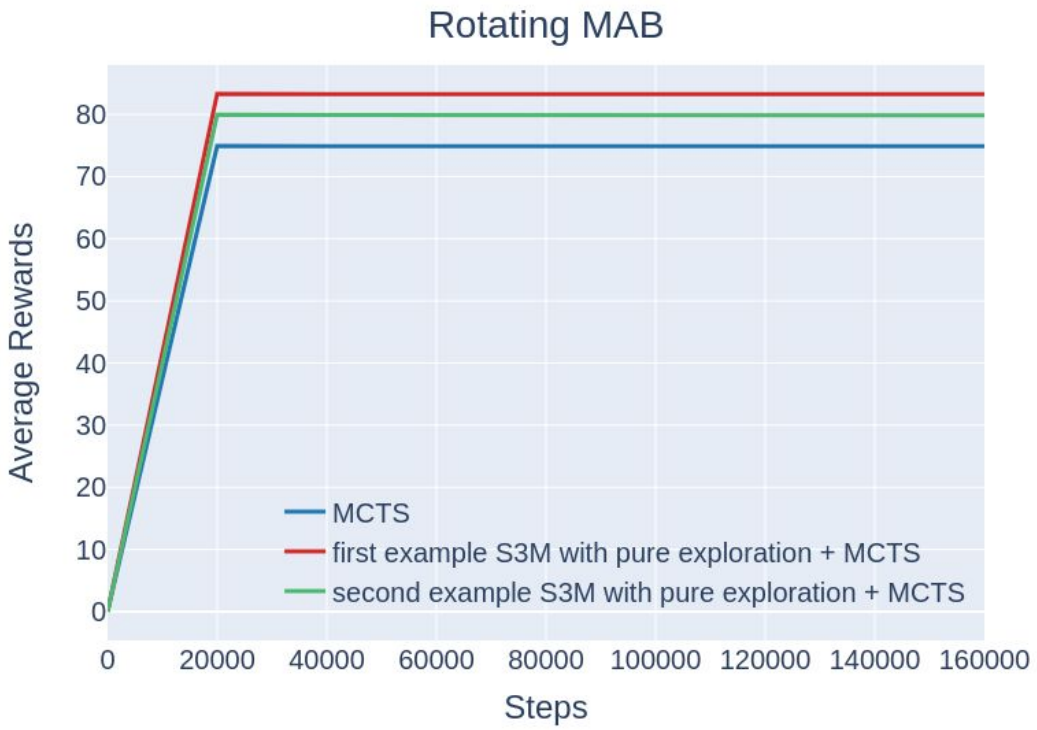
- S3M algorithms
  - Large variability in the MM' structure, ⟶ the merging between clusters
- Figure on the right shows a Mealy Machine very different from the ideal one
- A non-ideal MM > MCTS alone
  - MCTS learns better than alone with Cartesian product of the states of the RDP and the MM

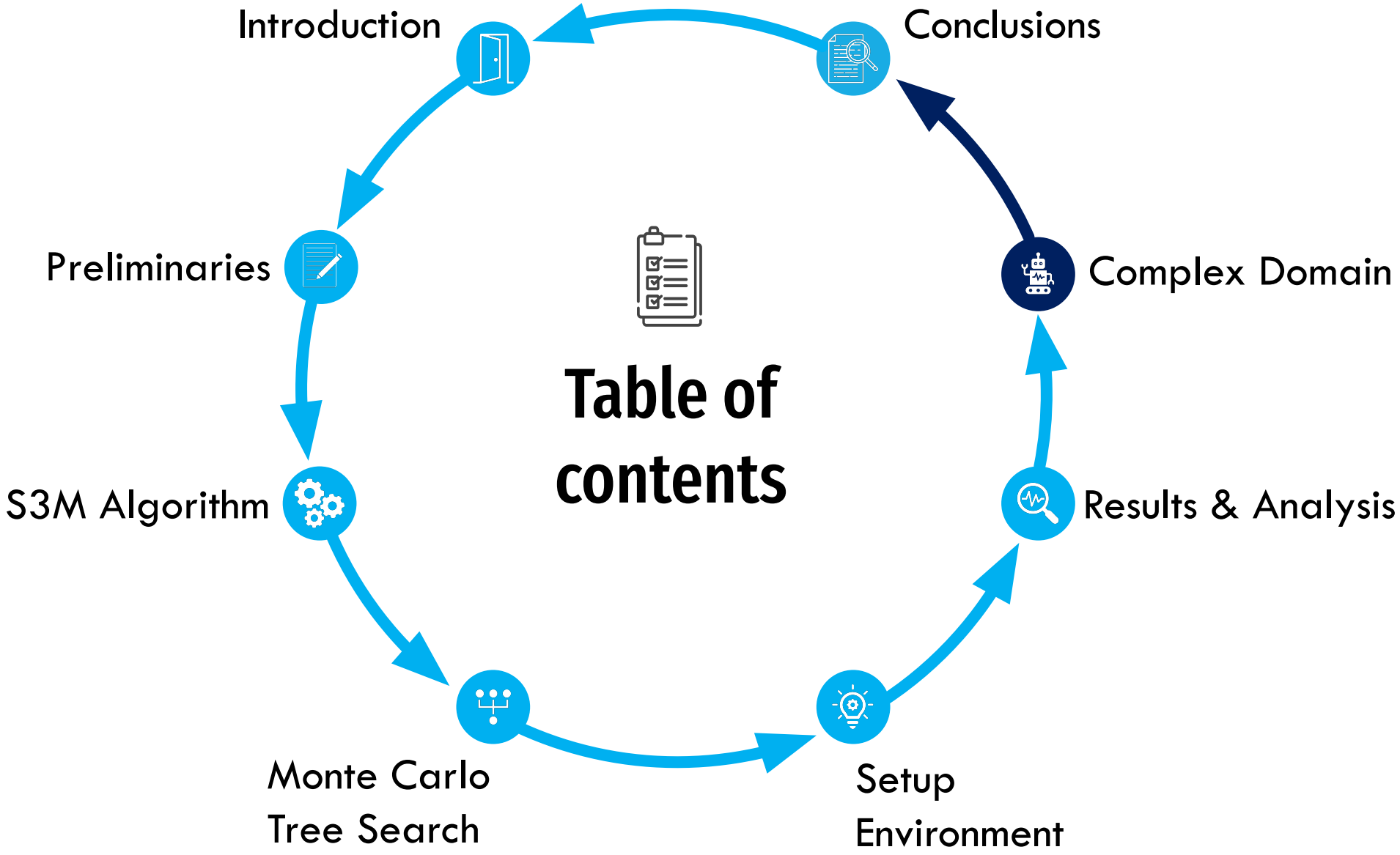




ANALYSIS ROTATING MAB (3/3)

- Results obtained with the MM closer to the ideal one and those with the MM furthest from ideal
- Complex to understand factors for this variability
  - Parameters:
    - **€:** significant in merging step ⟶ influence the structure of the MM
    - **min_sample:** larger ⟶ MM with more states
    - **Episode' length:** shorter episodes produce similar traces and tend to merge more ⟶ influence MM




Introduction
Conclusions
Preliminaries
Complex Domain
S3M Algorithm
Results & Analysis
Monte Carlo
Tree Search
Setup
Environment
Table of contents

OVERVIEW PAC-RL

Make the agent discover near-optimal policies in the shortest amount of time, without strictly maximizing the reward gathered during learning.
⟶ expect agent to return policies that improve with time

OVERVIEW PAC-RL

Make the agent discover near-optimal policies in the shortest amount of time, without strictly maximizing the reward gathered during learning.
⟶ expect agent to return policies that improve with time

- RL: an agent learning a policy for an unknown RDP $\mathcal{P}$ by obtaining observation states and rewards based on transition and reward functions

OVERVIEW PAC-RL

🎯 Make the agent discover near-optimal policies in the shortest amount of time, without strictly maximizing the reward gathered during learning.
⟶ expect agent to return policies that improve with time

- RL: an agent learning a policy for an unknown RDP $\mathcal{P}$ by obtaining observation states and rewards based on transition and reward functions
  - Agent
    - performs actions
    - produces episodes ⟶ experience ⟶ useful for learning a policy
    - receives policies that define its behaviour, which will then improve as it gets better policies

OVERVIEW PAC-RL

How quickly an agent can reach a point from which it exclusively produces favorable policies?

OVERVIEW PAC-RL

How quickly an agent can reach a point from which it exclusively produces favorable policies?

- Based on the PAC framework
  - Exact learning is not possible
  - Introduces two parameters:
    - $\epsilon > 0 \longrightarrow$ required accuracy
    - $\delta \in (0, 1) \longrightarrow$ confidence of success

OVERVIEW PAC-RL

How quickly an agent can reach a point from which it exclusively produces favorable policies?

- Based on the PAC framework
  - Exact learning is not possible
  - Introduces two parameters:
    - $\epsilon > 0 \longrightarrow$ required accuracy
    - $\delta \in (0, 1) \longrightarrow$ confidence of success
  - $\longrightarrow$ translate in the RL context into search for policies that are ϵ-optimal with a probability of at least 1-δ

OVERVIEW PAC-RL

How quickly an agent can reach a point from which it exclusively produces favorable policies?

- Based on the PAC framework
  - Exact learning is not possible
  - Introduces two parameters:
    - $\epsilon > 0 \longrightarrow$ required accuracy
    - $\delta \in (0, 1) \longrightarrow$ confidence of success
  - $\longrightarrow$ translate in the RL context into search for policies that are ϵ-optimal with a probability of at least 1-δ
  - RL is feasible if these policies can be discovered in a number of steps which is **polynomial** in $1/\epsilon$ and $\ln(1/\delta)$, and in other RDP-related factors

OVERVIEW PAC-RL

- RDP =<A, S, R, T, R, γ> is defined by a set of parameters:

$$\mathbf{d}_{\mathcal{P}} = (|A|, \frac{1}{1-\gamma}, R_{max}, n, \frac{1}{\rho}, \frac{1}{\mu}, \frac{1}{\eta})$$

- **|A|**: number of actions
- γ: discount factor
- **Rmax**: the maximum reward value
- **n**: number of states of the minimal transducer
- **ρ**: reachability ⟶ how simple it is to reach states of the dynamics transducer T within *n* steps by taking random actions
- **μ**: distinguishability ⟶ how easy it is to distinguish states of the dynamics transducer T
- η: degree of determinism ⟶ how straightforward it is to find transitions
  - If η is tiny, there are transitions that, although possible, are unlikely to be detected

WEAKNESSES OF THE ALGORITHMS

- RDP =<A, S, R, T, R, γ> is defined by a set of parameters:

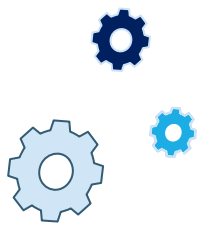
$$\mathbf{d}_{\mathcal{P}} = (|A|, \frac{1}{1-\gamma}, R_{max}, n, \frac{1}{\rho}, \frac{1}{\mu}, \frac{1}{\eta})$$

RL is feasible if these policies can be discovered in a number of steps which is **polynomial** in these parameters

  - Unfavorable parameters will make it difficult to converge to near-optimal policies

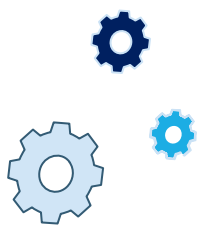
WEAKNESSES OF THE ALGORITHMS

- S3M algorithm:
  - in Rotating Maze domain the Smart Sampling technique is less effective
    - Pure exploration more efficient in this domain



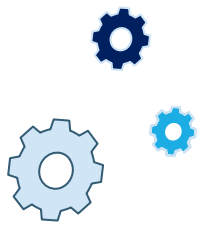
WEAKNESSES OF THE ALGORITHMS

- S3M algorithm:
  - in Rotating Maze domain the Smart Sampling technique is less effective
    - Pure exploration more efficient in this domain
    - Effectiveness of Pure exploration can be attributed to two factors:
      1. Many states and actions
         a. better domain exploration ⟶ selection of the less sampled action is encouraged
         b. Produced traces are more varied ⟶ higher accurate statics on various states
      2. Simple Regular Function



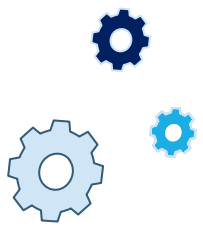
WEAKNESSES OF THE ALGORITHMS

- S3M algorithm:
  - in Rotating Maze domain the Smart Sampling technique is less effective
  - however, in a grid domain, the effectiveness of Pure exploration can be saturated
    - bigger grid
    - regular function that governs the dynamics more complex
    - exploration ≠ informative clusters



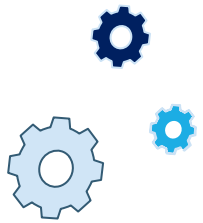
WEAKNESSES OF THE ALGORITHMS

- S3M algorithm:
  - in Rotating Maze domain the Smart Sampling technique is less effective
  - however, in a grid domain, the effectiveness of Pure exploration can be saturated
    - bigger grid
    - regular function that governs the dynamics more complex
    - exploration ≠ informative clusters
  - In a large grid domain with obstacles, Smart Sampling does not overcome the difficulties

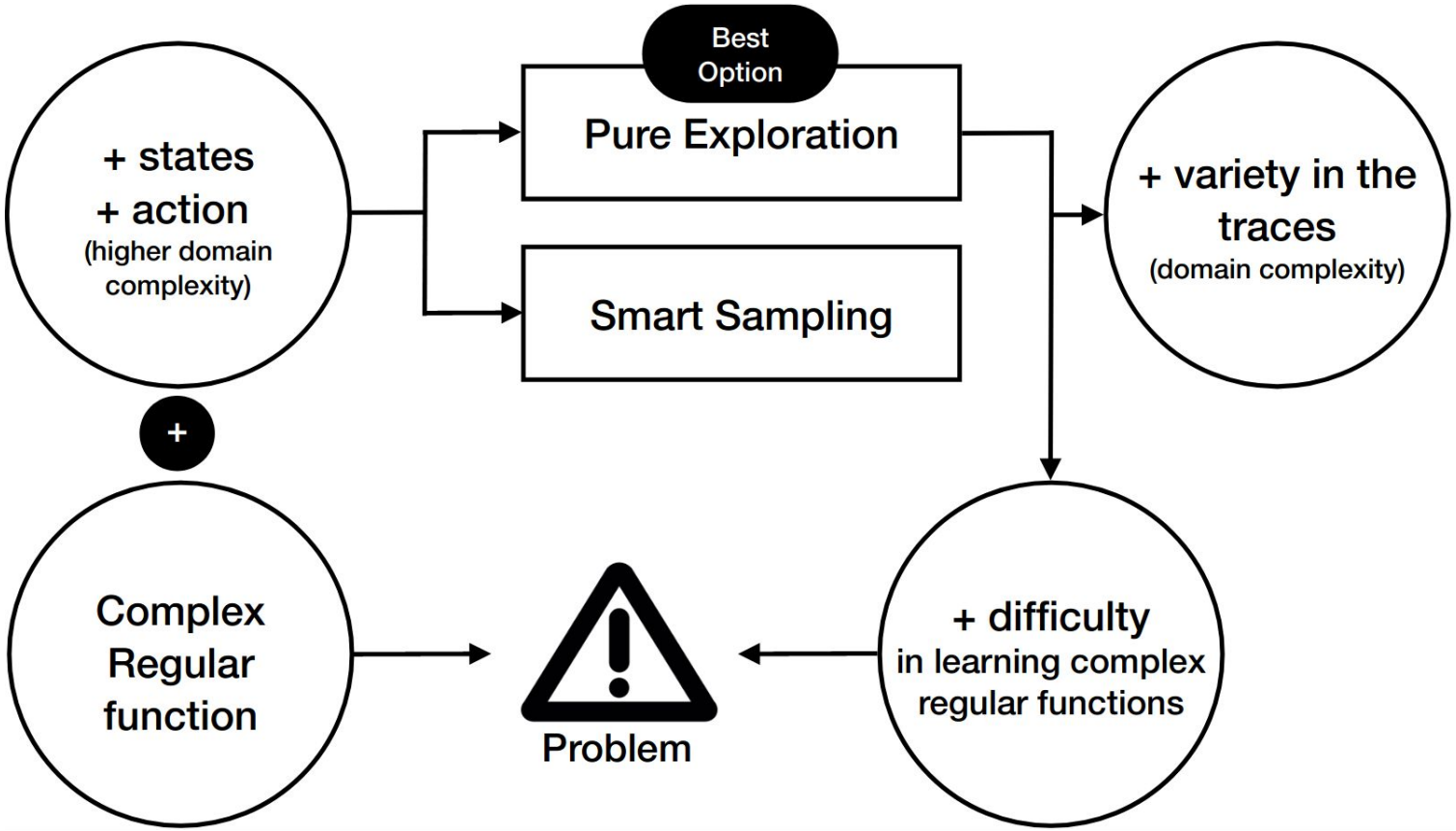


WEAKNESSES OF THE ALGORITHMS

- S3M algorithm:
  - in Rotating Maze domain the Smart Sampling technique is less effective
  - however, in a grid domain, the effectiveness of Pure exploration can be saturated
    - bigger grid
    - regular function that governs the dynamics more complex
    - exploration ≠ informative clusters
  - In a large grid domain with obstacles, Smart Sampling does not overcome the difficulties
    - when doing Q-learning it will be biased in the choice of pretty limited sequences of actions
    - even with an ε-greedy procedure, it will still not produce a varied set of good-quality traces and clusters

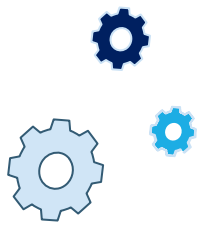


WEAKNESSES OF THE ALGORITHMS



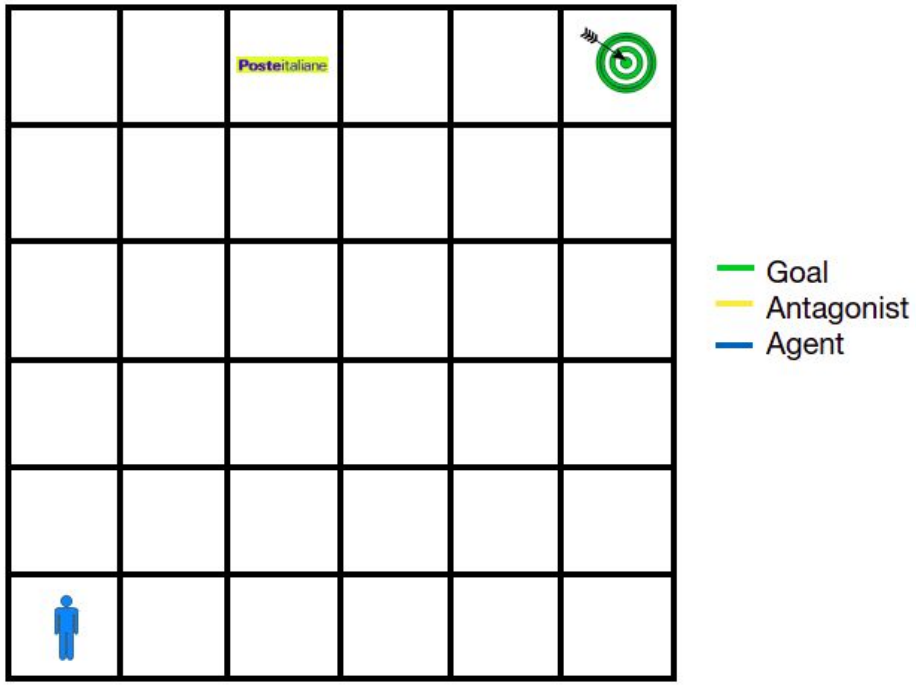
WEAKNESSES OF THE ALGORITHMS

- PAC-RL
  - sufficiently large grid domain
  - enemy/enemies or non-viable moving blocks
  - the reachability of a grid is likely to be significantly reduced
  - in general, all parameters can be made unfavorable
    ⟶ preventing the algorithm from converging in polynomial time



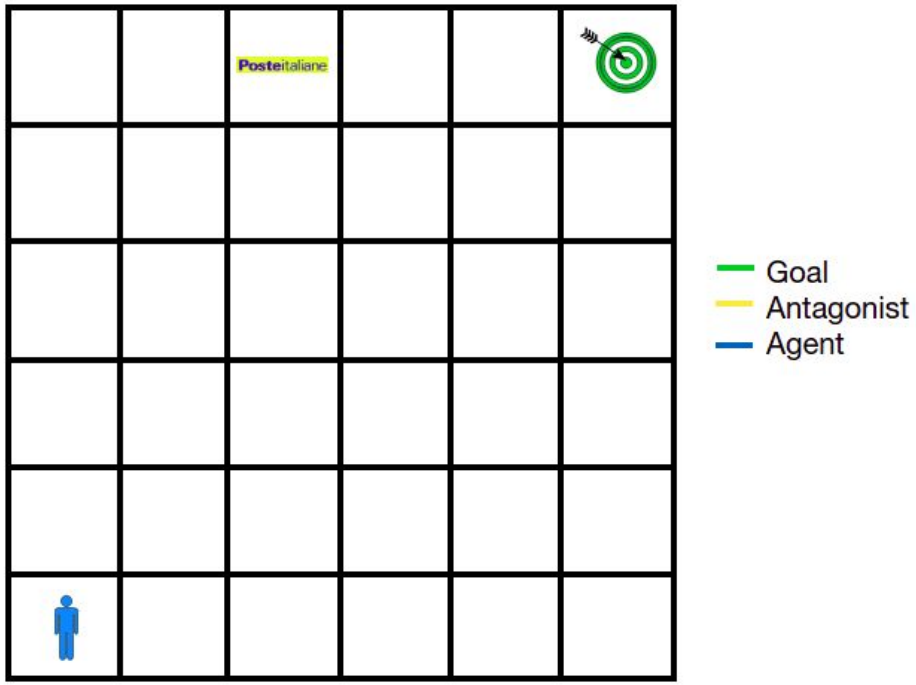
GRID DOMAIN

- Domain:
  - Agent
  - Antagonist
  - Goal



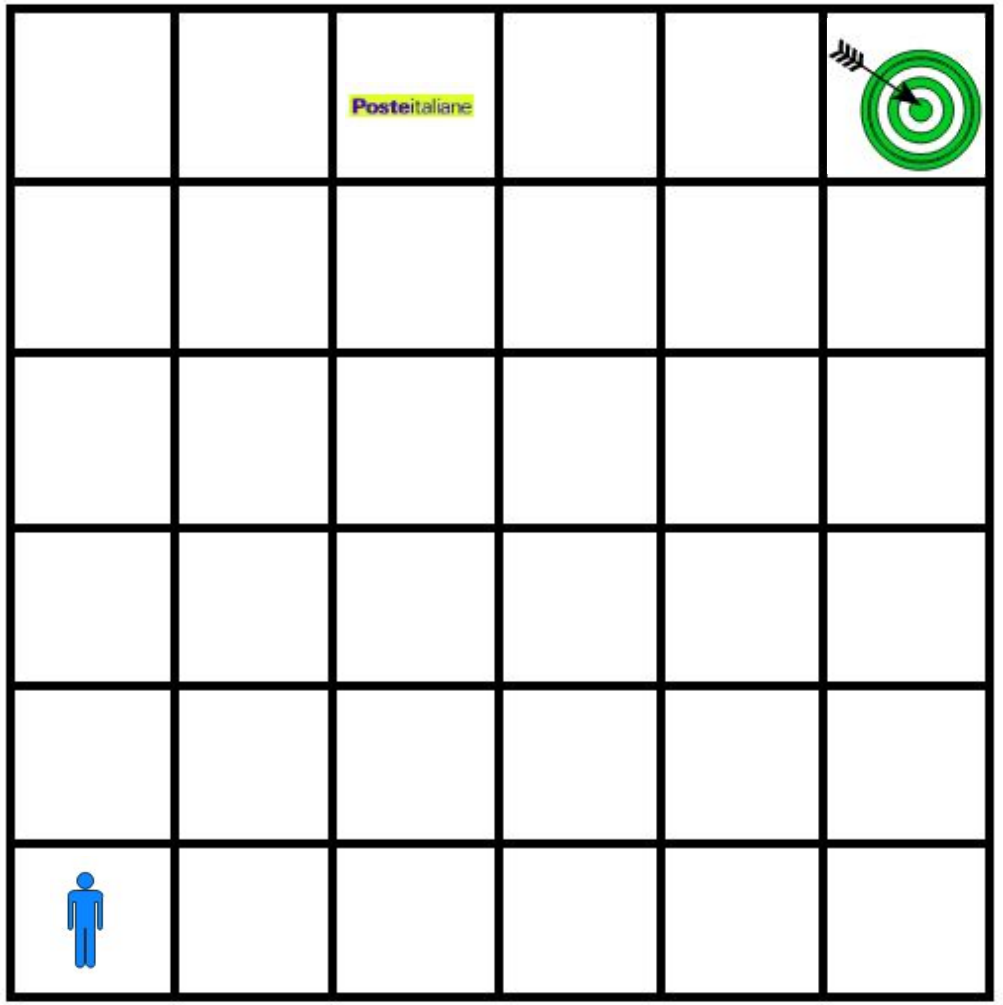
GRID DOMAIN

- Domain:
  - Agent
  - Antagonist
  - Goal
- Antagonist moves at each agent's action toward one of the four surrounding cells according to a distribution
- Grid rotate every *n* actions
- If the agent is in the same cell as his opponent, the algorithm will start over



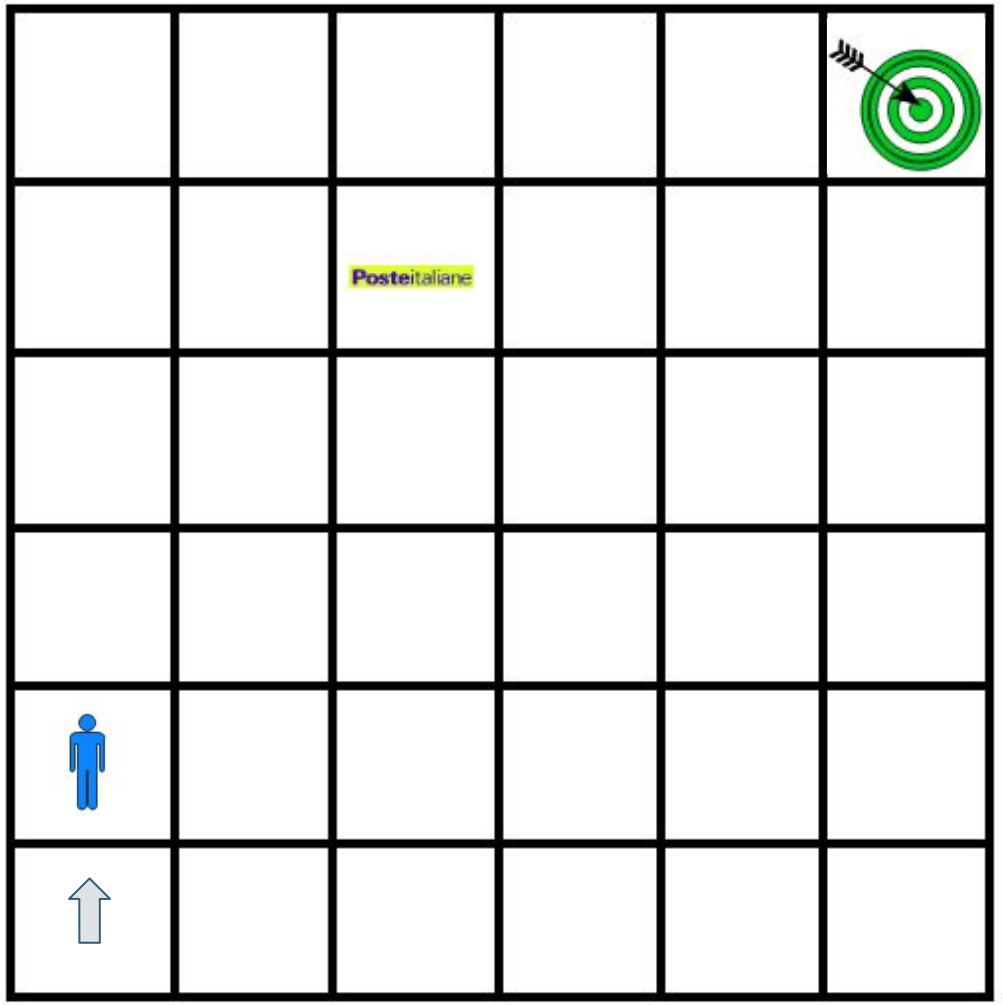
EXAMPLE

➔ The agent takes a sequence of actions for each episode



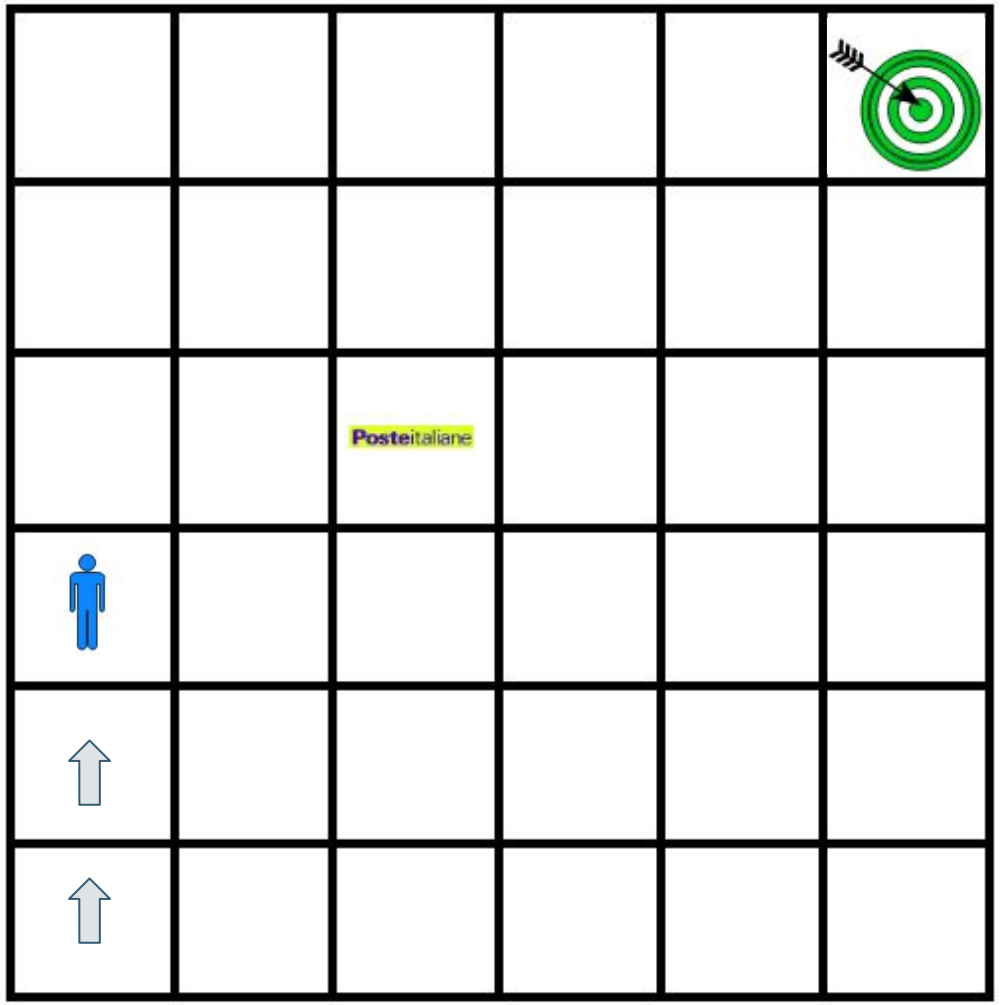
EXAMPLE

➔ The agent takes a sequence of actions for each episode



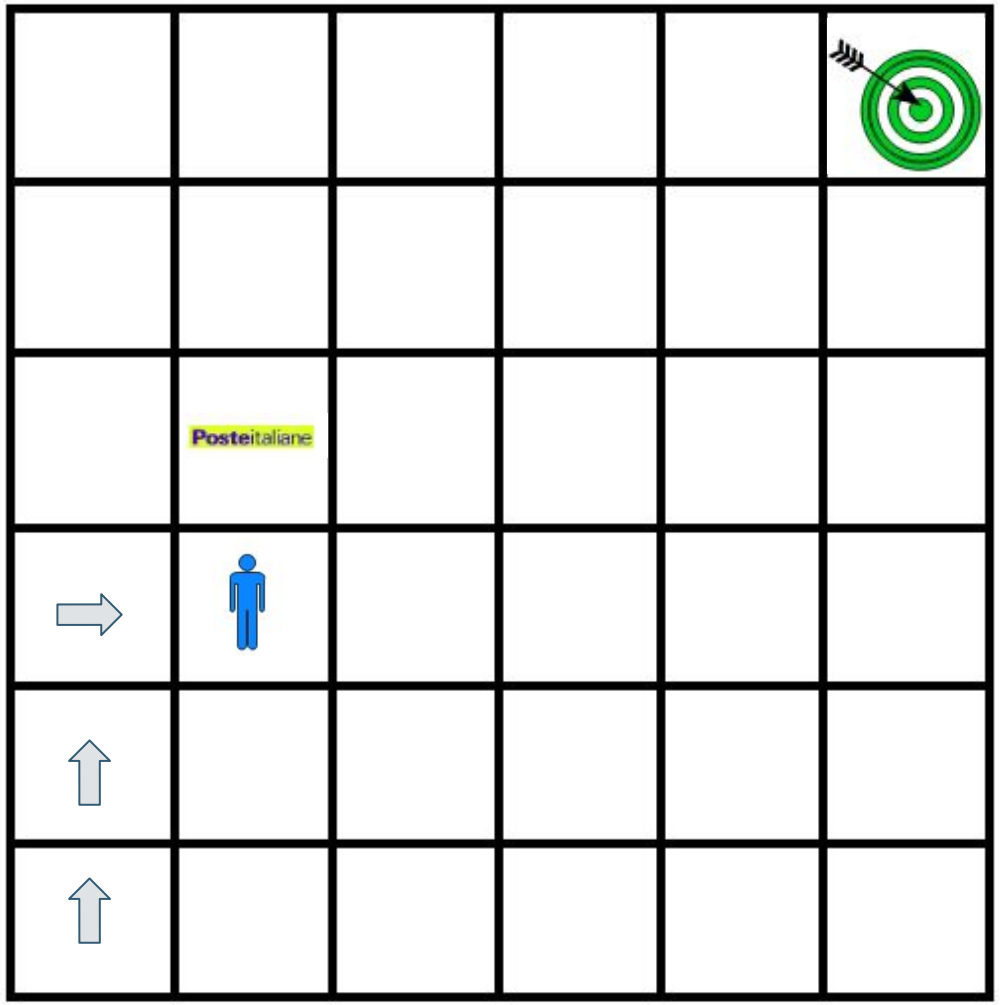
EXAMPLE

➔ The agent takes a sequence of actions for each episode



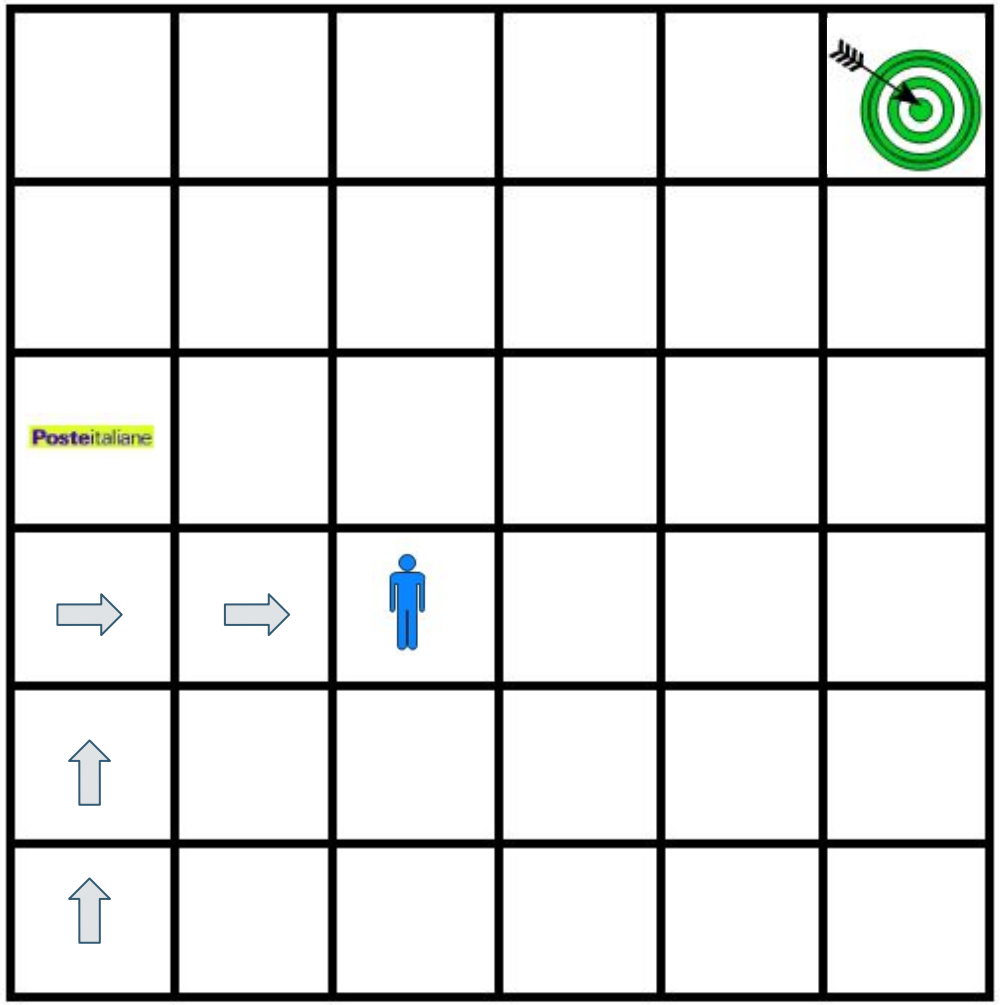
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



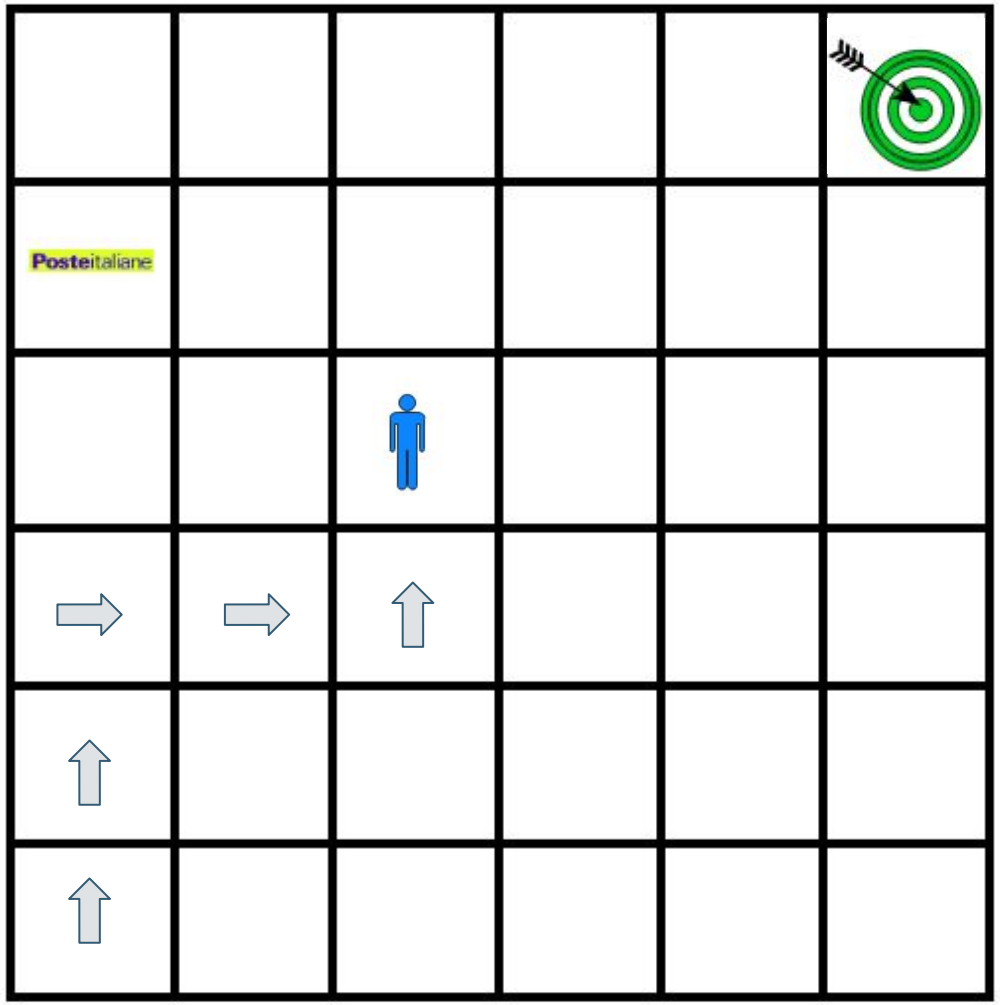
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



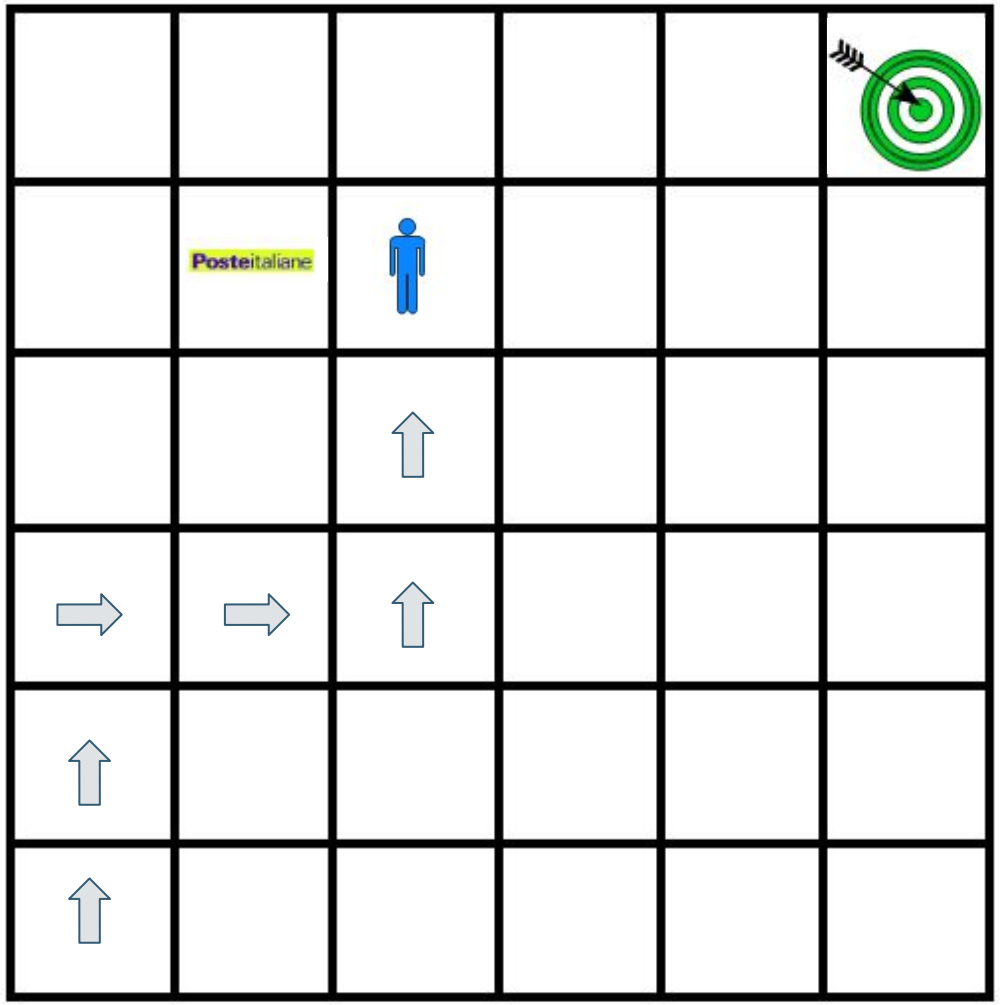
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



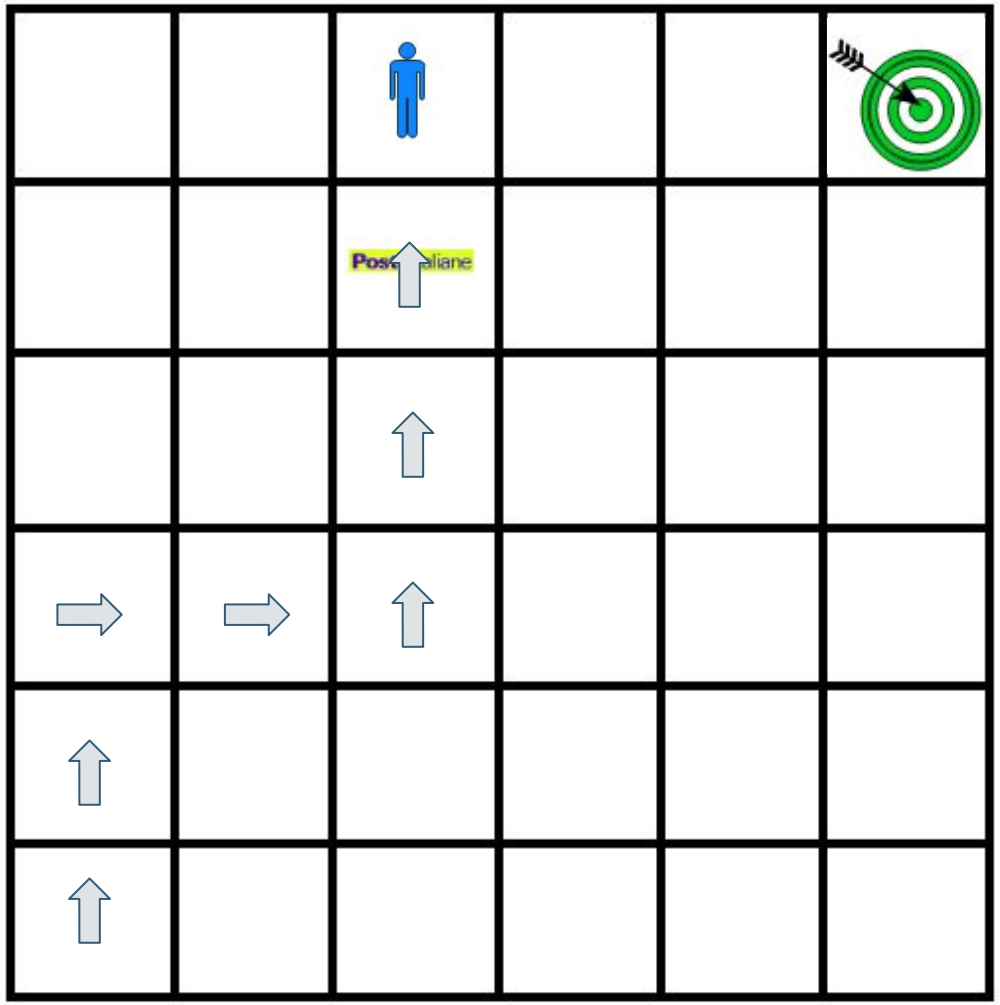
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



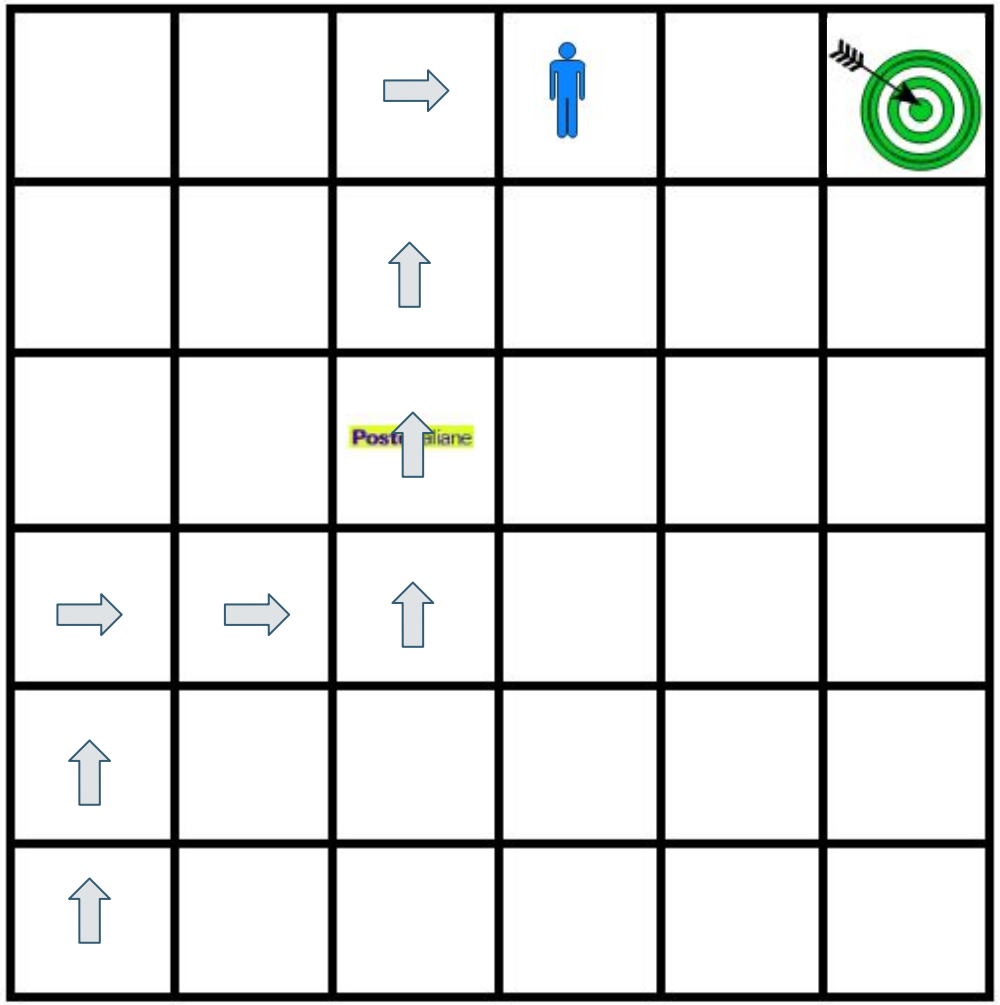
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



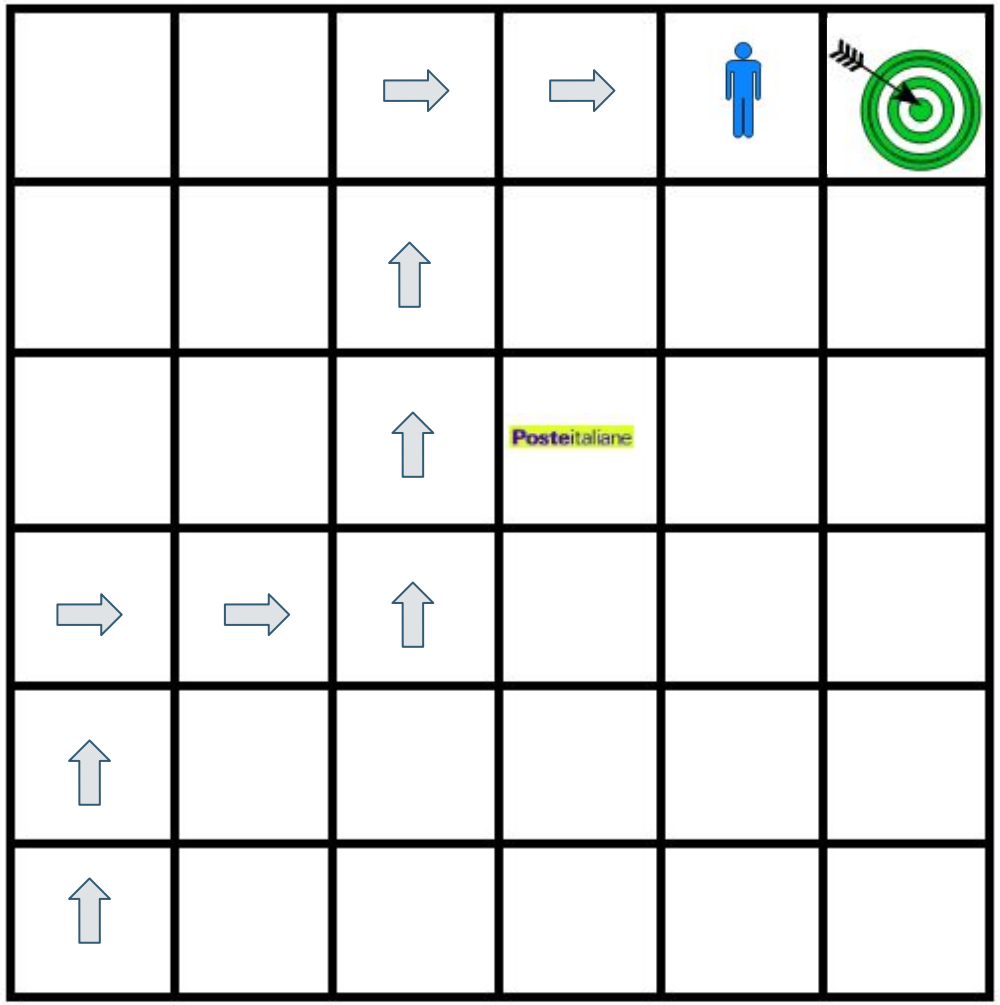
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



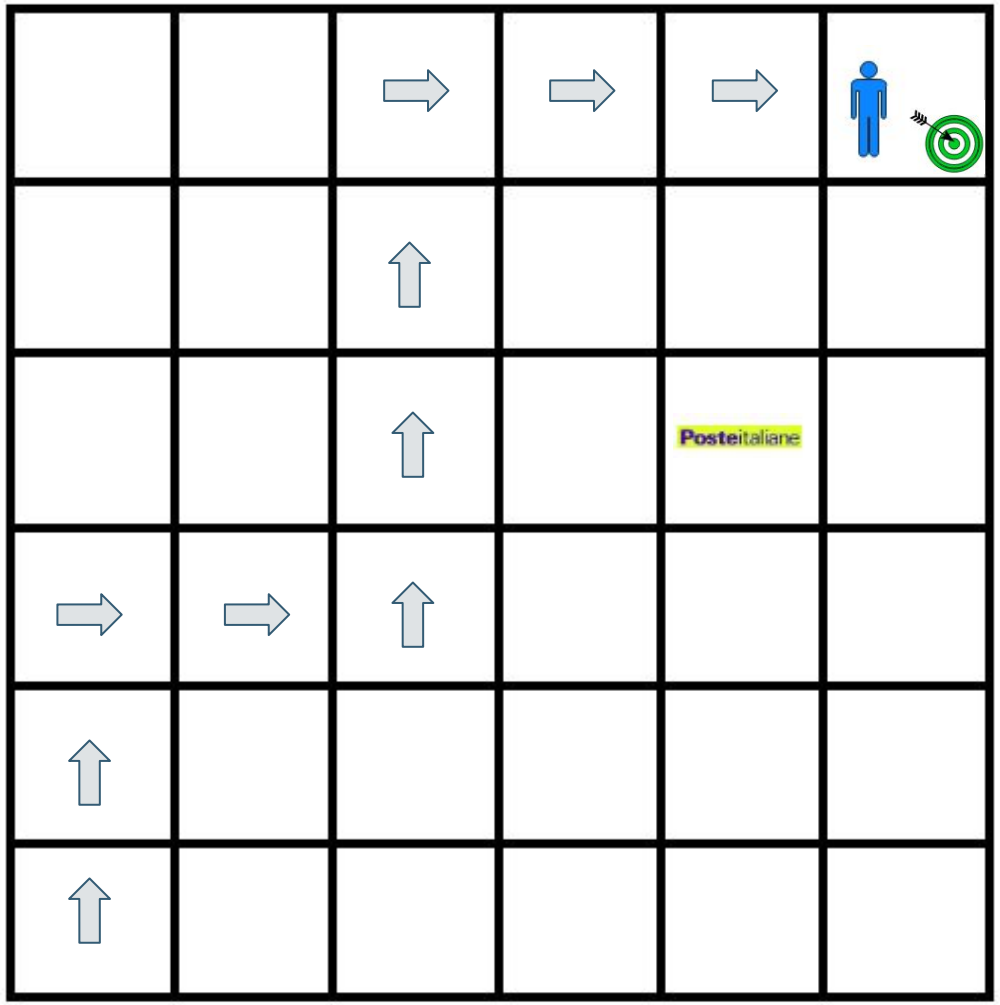
EXAMPLE

- ➔ The agent takes a sequence of actions for each episode
- ➔ The antagonist is an enemy but not an RL agent



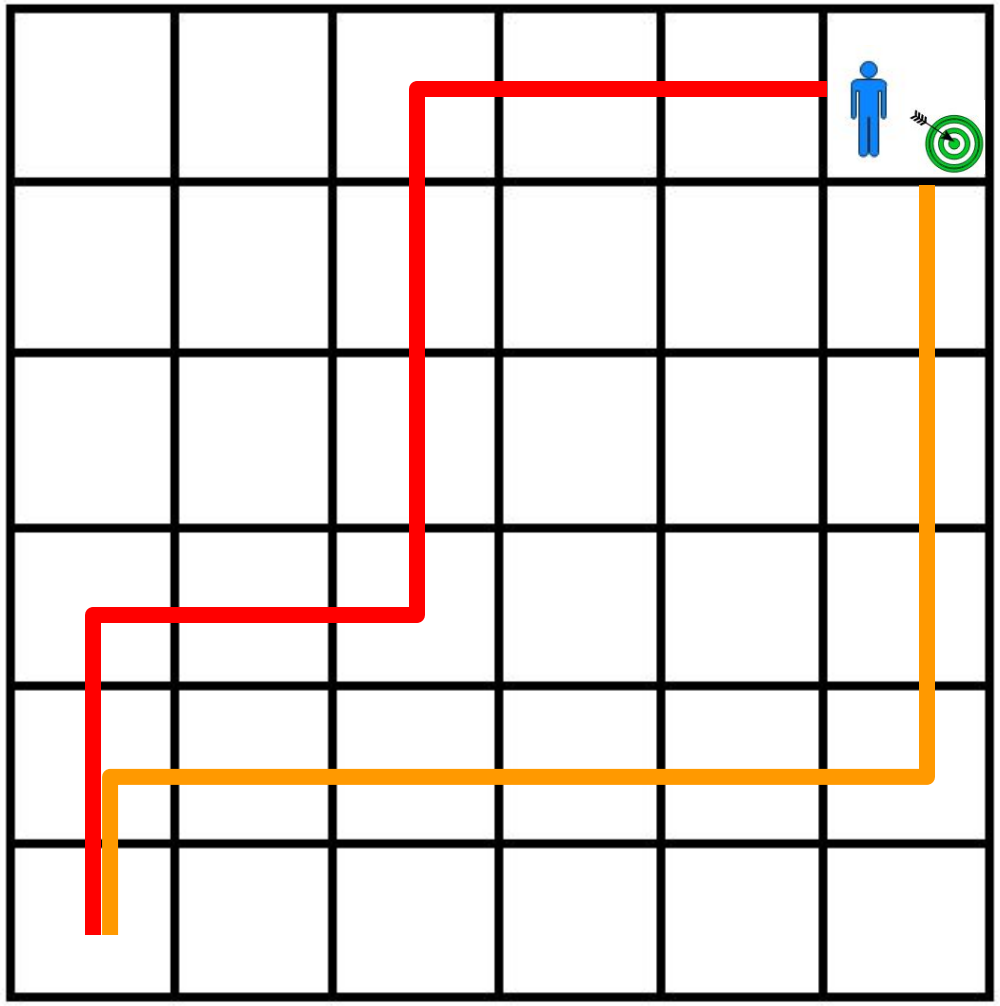
EXAMPLE

- The agent takes a sequence of actions for each episode
- The antagonist is an enemy but not an RL agent
- When using Smart Sampling, reaching the goal will result in the updating of the Q-values



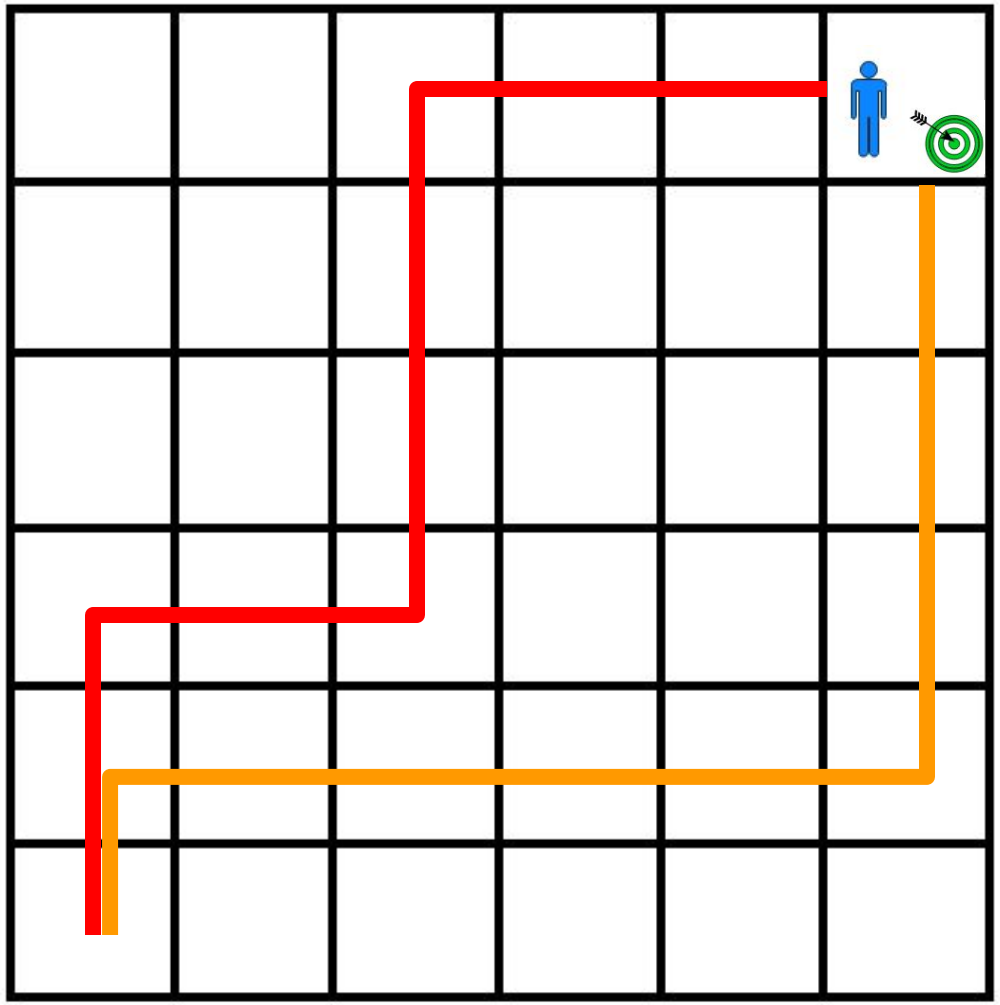
EXAMPLE

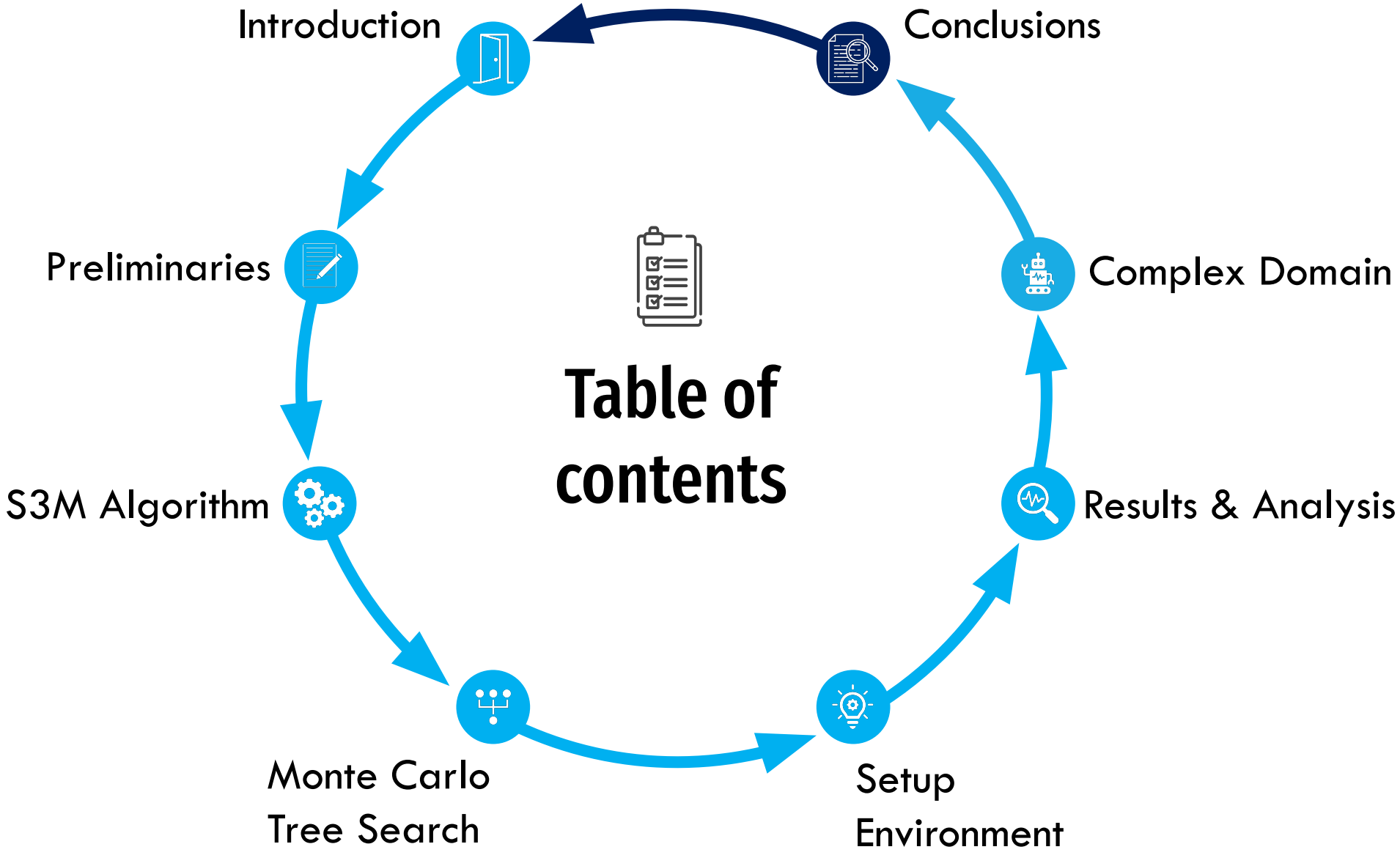
- The agent takes a sequence of actions for each episode
- The antagonist is an enemy but not an RL agent
- When using Smart Sampling, reaching the goal will result in the updating of the Q-values



EXAMPLE

- The agent takes a sequence of actions for each episode
- The antagonist is an enemy but not an RL agent
- When using Smart Sampling, reaching the goal will result in the updating of the Q-values
  - Some sequences of actions will be much more favored than others, so limiting exploration




Introduction
Conclusions
Preliminaries
Complex Domain
Table of contents
S3M Algorithm
Results & Analysis
Monte Carlo Tree Search
Setup Environment

CONCLUSIONS

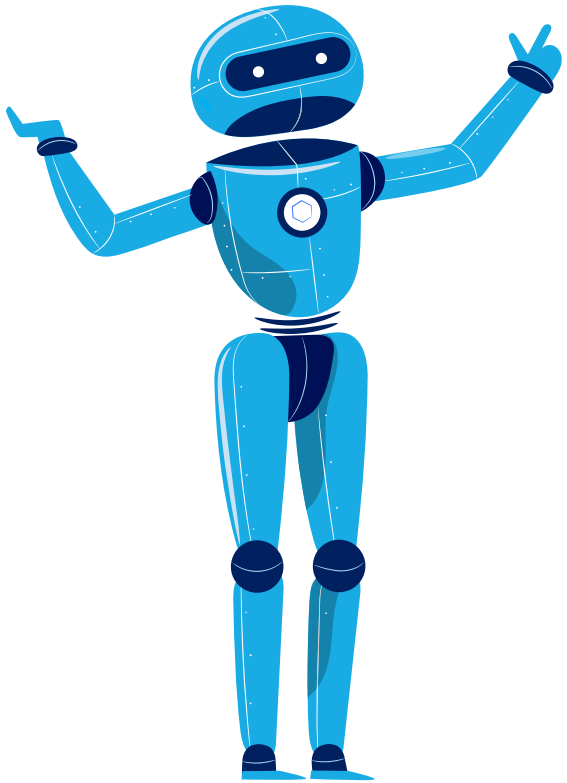
Theoretical presentation on **RDP**

S3M implementation

Analysis of the results

Posteitaliane

Cons analysis
and definition of a **complex domain**



THANKS FOR YOUR ATTENTION