Pure Past Action Masking

Natasha Alechina

Open University
Utrecht University

natasha.alechina@ou.nl

OUrsi, 21 November 2023

Reinforcement learning to logical specification

- specify in logic what an agent should achieve: this specification is used to generate rewards for the agent (Promis 2023 talk)
- specify in logic what the agent should *not* do (a 'shield', or a 'restraining bolt' which is an automaton attached to the agent intercepting unsafe actions) (this talk)
- the learned policy is guaranteed to satisfy the logical specification
- the talk is based on joint work with Giovanni Varricchione, Mehdi Dastani,Giuseppe De Giacomo, Brian Logan, Giuseppe Perelli

Introduction to Reinforcement Learning

- an agent learns to complete a task in a given environment
- environment dynamics are unknown to the agent
- however, the agent can perform actions in the environment and make observations
- each time the agent performs an action, the environment transitions to a new state and the agent receives a reward

See, Sutton & Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2020

Reinforcement learning

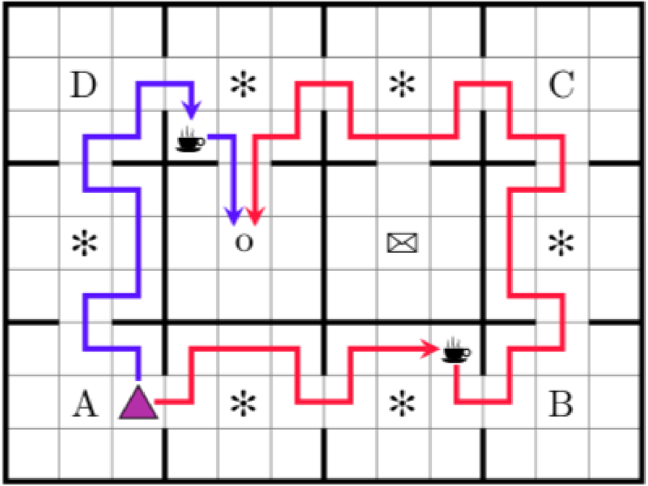
Reinforcement learning

- agent and environment interact at discrete time steps: $t = 0, 1, 2, \ldots$
- agent observes state at step t: $s_t \in S$
- agent chooses an action at step t: $a_t \in A(s_t)$
- environment returns the state resulting from executing the action $s_t + 1$, and the corresponding reward $r_t + 1 \in \mathbb{R}$
- probability distribution over the outcomes of action

Example: OfficeWorld

- OfficeWorld is a simple grid-world environment that includes coffee stations and a mail room
- not all rooms are connected to adjacent rooms, and some rooms contain fragile “decorations”
- goal of the agent is to deliver coffee and/or mail while not stepping on the decorations

Example:OfficeWorld



RL Problem

- the single agent reinforcement learning problem is to learn an optimal policy π^* that maximises the expected discounted future rewards
- a policy is a function $\pi : S \to \Delta(A)$ mapping each state to a probability distribution over the set of actions.
- executing a policy produces a sequence of states $s_0, s_1, s_2, \ldots$

Safe RL definition from Alshiekh et al 2018

Definition (Safe RL)

Safe RL is the process of learning an optimal policy while satisfying a temporal logic safety specification φ_s during the learning and execution phases.

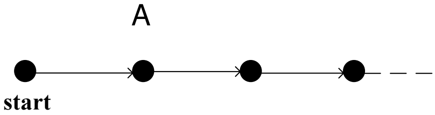
Formal Language of Temporal Logic

- the language of Linear Time Temporal Logic (LTL) describes properties of sequences of states, or paths (e.g. generated by an RL agent executing its policy)
- includes properties of states, boolean connectives (not, and, or, implies), and
- temporal operators: NeXt (X), Globally (G), Until (U)

Next

In the next state, some A holds (that can be about temporal property again)

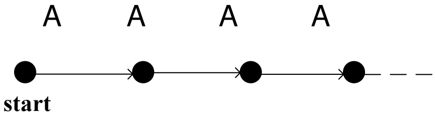
X A



Globally

In all states on the path, A holds

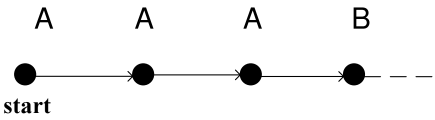
G A



Until

In some future state, B holds, and in all states before that, A holds

A U B



Shields for Safe RL

- Shields (Alshiekh et al 2018, ElSayed-Aly et al 2021) are a recently proposed approach to **provably** safe RL.
- the main idea behind shields is to ensure the agent does not take actions that can lead to safety violations (it looks ahead, arbitrary number of steps)
- the shield is placed in the classic reinforcement learning loop; depending on the variant of the shield, it operates either **before** (*preemptive shield*) the agent decides which action to take, or **after** (*post-posed shield*)

Preemptive Shields

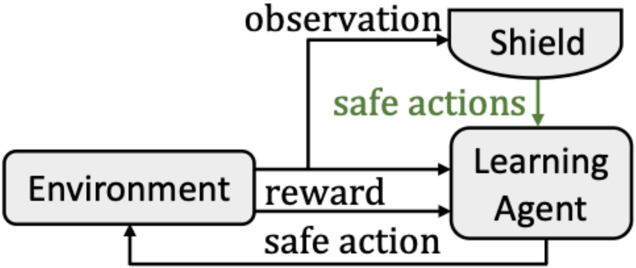


Figure 1: Preemptive shield pipeline

Preemptive shield outputs a set of safe actions at each timestep

How shields work (high-level description)

- a shield is a Mealy machine (automaton with output)
- on input of its current state and current observations, it outputs sets of safe actions for the agent to choose from
- the shield should be minimally restrictive, that is, an action is only excluded if it may lead to an unsafe state (eventually)
- example: a car heading towards a cliff; actions that should be excluded by the shield involve continuing in the same direction; stopping or going in a different direction should be allowed

Watertank example

- an RL agent is learning to optimise energy consumption by a watertank
- safety property: the agent has to keep the tank from depleting and overflowing; every time the valve is opened/closed, this setting must be kept for at least 3 seconds/timesteps:

$$
\begin{aligned}
& G(\mathit{level} > 0) \wedge G(\mathit{level} < 100) \\
\wedge & G((\mathit{open} \wedge X\mathit{close}) \rightarrow XX\mathit{close} \wedge XXX\mathit{close}) \\
\wedge & G((\mathit{close} \wedge X\mathit{open}) \rightarrow XX\mathit{open} \wedge XXX\mathit{open})
\end{aligned}
$$

Complexity of generating shields

- building a safety DFA $\mathcal{D}^\varphi$ that recognizes the safety language of a safety LTL formula φ takes time double exponential in the size of φ
- the size of the automaton $\mathcal{D}^\varphi$ is also double exponential in that of φ
- the size of the safety game $\mathcal{G}$ (on a product of the safety automaton and an automaton that is an abstraction of the MDP) is double exponential in the size of φ, times the size of the MDP abstraction automaton $\mathcal{D}^M$
- Solving safety games takes linear, in the size of the game, time
- this implies the following:

The time complexity to generate a shield and the size of the shield itself is double exponential in the size of the input safety LTL formula φ

Our idea

- use a ‘cheaper’ specification language
- we use a logic for talking about the past: Pure Past LTL (PPLTL)
- for each action, we specify a ‘precondition’ in PPLTL which says when it is safe to execute this action

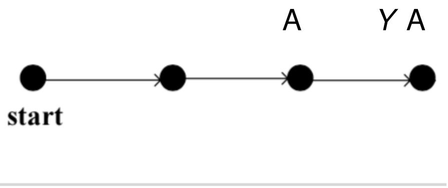
Pure Past Temporal Logic PPLTL

- the language of PPLTL describes properties of finite sequences of states, paths or histories (e.g. generated by an RL agent executing its policy from some initial state)
- includes properties of states, boolean connectives (not, and, or, implies), and
- temporal operators: Yesterday (Y), Historically (H), Since (S)

Yesterday

In the previous state, some A holds

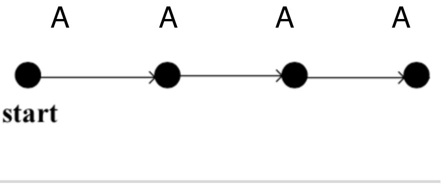
$Y\ A$



Historically

In all states on the path, A holds

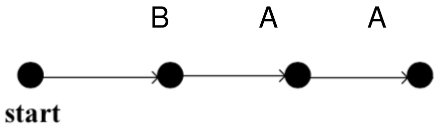
H A



Since

In some past state, B holds, and in all states since then, A holds

A $\mathcal{S}$ B



Pure Past Action Masking

Definition

A pure past action mask (PPAM) $= (\mathcal{L}, \{\varphi_a : a \in A\})$ is a pair where:

- $\mathcal{L}$ is the set of possible PPAM "observations";
- $\{\varphi_a : a \in A\}$ is the set of PLTL formulas, each constraining its corresponding action.

PPAM Pipeline

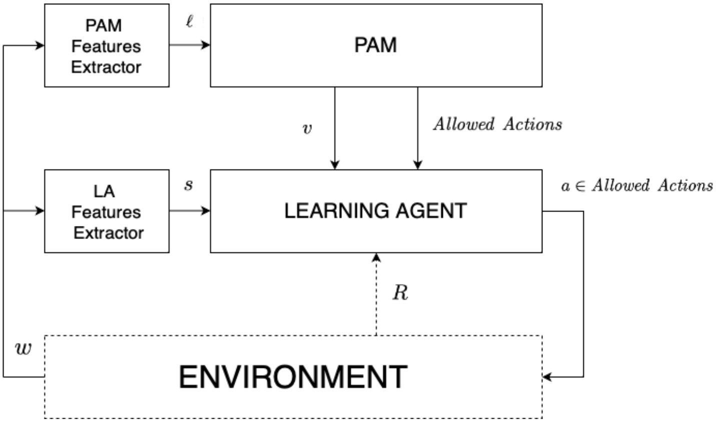


Figure 2: Interaction between the environment, the (learning) agent and the PPAM.

Watertank

Recall the Watertank example

We can easily specify a PPAM that enforces the same constraints of the shield as follows:

$$\mathcal{L} = 2^{\mathcal{F}}, \quad \text{where } \mathcal{F} = \{close, open, level \leq 93, level \geq 4\}$$

$$\varphi_{open} = level \leq 93 \wedge (close \rightarrow Y close \wedge YY close)$$
$$\varphi_{close} = level \geq 4 \wedge (open \rightarrow Y open \wedge YY open)$$

Complexity and Expressiveness

- the new MDP for the agent to learn is single exponential in the size of the PPAM (of the formulas)
- for every Safety LTL specification (for a shield), there is a corresponding PPAM (a set of past formulas), and vice versa
- however the size of the corresponding formulas can be very large; similarly to translation between past and future temporal formulas
- some safety properties are more naturally expressed in future time logic, and some in past time logic

References

- Sutton & Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2020
- Leike et al. AI Safety Gridworlds. arXiv:1711.09883
- Mohammed Alshiekh et al. *Safe Reinforcement Learning via Shielding*. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence AAAI Press, 2018, pp. 2669-2678.
- Ingy ElSayed-Aly et al. *Safe Multi-Agent Reinforcement Learning via Shielding*. In: Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems. IFAAMAS, 2021, pp. 483-491.