Pure-Past Action Masking

Giovanni Varricchione[1], Natasha Alechina[2,1], Mehdi Dastani[1], Giuseppe De Giacomo[3], Brian Logan[4,1], Giuseppe Perelli[5]

[1]Utrecht University
[2]Open University
[3]University of Oxford
[4]University of Aberdeen
[5]Sapienza University of Rome
g.varricchione@uu.nl, natasha.alechina@ou.nl, m.m.dastani@uu.nl,
giuseppe.degiacomo@ox.ac.uk, brian.logan@abdn.ac.uk, giuseppe.perelli@di.uniroma1.it

Abstract

We present Pure-Past Action Masking (PPAM), a lightweight approach to action masking for safe reinforcement learning. In PPAM, actions are disallowed ("masked") according to specifications expressed in Pure-Past Linear Temporal Logic (PPLTL). PPAM can enforce non-Markovian constraints, i.e., constraints based on the history of the system, rather than just the current state of the (possibly hidden) MDP. The features used in the safety constraint need not be the same as those used by the learning agent, allowing a clear separation of concerns between the safety constraints and reward specifications of the (learning) agent. We prove formally that an agent trained with PPAM can learn any optimal policy that satisfies the safety constraints, and that they are as expressive as shields, another approach to enforce non-Markovian constraints in RL. Finally, we provide empirical results showing how PPAM can guarantee constraint satisfaction in practice.

1 Introduction

While AI agents based on reinforcement learning (RL) (Sutton and Barto 2018) have been remarkably successful in a variety of domains, ensuring the safety of agents developed using learning remains a major research challenge (Amodei et al. 2016; Hadfield-Menell et al. 2017; Orseau and Armstrong 2016). Achieving safe and robust behaviour using state-of-the-art learning algorithms and a single reward function is challenging (Leike et al. 2017). In simulated environments, agents are able to explore the whole action space, regardless of the consequences of their actions. However, in safety-critical applications where agents must learn in the real environment, such as autonomous cars (Mirchevska et al. 2018; Kendall et al. 2019), or chemical processes (Zhou, Li, and Zare 2017; Savage et al. 2021), unconstrained exploration can lead to catastrophic failure, both during training and following deployment. In such cases, it is necessary to ensure that that only safe actions are explored during learning. Such approaches are termed "*provably safe*" in (Krasowski et al. 2023).

A number of approaches to provably safe RL have been proposed in the literature, including: *action replacement*, e.g., (Hunt et al. 2021), where unsafe actions are replaced by (default or sampled) safe ones; *action projection*, e.g., (Cheng et al. 2019), where unsafe actions are replaced by projecting them to the safe action space; and *action masking*, e.g., (Krasowski, Wang, and Althoff 2020), where the agent is able to choose actions only from the safe action space. However, most provably safe approaches to RL are Markovian, that is, the set of actions which are considered "safe" is based only on the current state. There has been some work in which safety constraints are expressed in temporal logic. For example, *shields* (Alshiekh et al. 2018; ElSayed-Aly et al. 2021) is an action masking approach in which safety constraints are expressed in Safety LTL. The use of temporal logic means safety constraints can be specified over execution traces rather than states, allowing a much richer set of *non-Markovian safety constraints* to be used, e.g., "do not close the valve if it was opened within the last three seconds" (Alshiekh et al. 2018). However, generating a shield from a Safety LTL specification requires solving a safety game over an abstraction of the underlying MDP and is double exponential in the size of the Safety LTL specification. Moreover, the agent must learn in the resulting doubly exponential product MDP.

In this paper, we present *Pure-Past Action Masking* (PPAM), a novel, lightweight approach to safe reinforcement learning. In PPAM, safety specifications are expressed in Pure Past Linear-time Temporal Logic (PPLTL) (De Giacomo et al. 2020). PPLTL is a variant of Linear-time Temporal Logic in which formulas are evaluated over finite traces and only past modalities are allowed. PPLTL formulas have a very natural interpretation: the formula must be true at the end of the trace and is evaluated "backwards". Many specifications about actions are easier and more natural to express when referring to the past (Lichtenstein, Pnueli, and Zuck 1985), and past temporal logics have been used in closely related problems with non-Markovian aspects, e.g., non-Markovian models in reasoning about actions (Gabaldon 2011), non-Markovian rewards in MDPs (Bacchus, Boutilier, and Grove 1996), and expressing norms in multi-agent systems (Alechina et al. 2015). PPAM has the same expressive power as shields but has significantly lower computational complexity (linear overhead in the state size). Moreover, the features used in the safety constraint need not be the same as those used by the learning agent, allowing

a clearer separation of concerns between safety constraints and reward specifications than with shields. Critically, we show that PPAM does not lead to any loss of generality, i.e., an agent trained with a mask is able to learn any optimal and safe (with respect to the safety constraints) policy in the MDP, if there is one. We illustrate our approach with a number of examples from the literature, and present experimental results which show that, in addition to ensuring safety, PPAM can improve sample efficiency.

2 Background

In this section, we briefly recall Markov Decision Processes (MDPs), and shields (Alshiekh et al. 2018; ElSayed-Aly et al. 2021) approach to safe RL.

2.1 Markov Decision Process

The (learning) agent and its environment are modelled as a Markov Decision Process (MDP).

Definition 1 (Markov Decision Process) *A (factorized) Markov Decision Process* $M_{ag} = (F, S, s_0, Act, A, Tr, R, \gamma)$ *is a tuple where:*

- F *are the (boolean) features of the MDP*[1]*;*
- $S = 2^F$ *is a set of states (corresponding to features evaluations);*
- s_0 *is the initial state;*
- Act *is a finite non-empty set of actions;*
- $A : S \to 2^{Act}$ *is an action availability function which maps each state of the MDP to the set of actions available in that state;*
- $Tr : S \times Act \rightharpoonup Pr(S)$ *is a transition function which for a pair of state* s *and action* a *returns a probability distribution for the resulting state if* $a \in A(s)$*, and is otherwise undefined;*
- $R : S \times Act \times S \to \mathbb{R}$ *a reward function; and*
- $\gamma \in [0, 1)$ *is the discount factor.*

In RL, the transition dynamics and reward (i.e., the Tr and R functions) are typically hidden from the agent, who has to learn the optimal policy by trial-and-error. A policy $\rho : S^+ \to Pr(Act)$ maps a sequence of states $\vec{s} = s_0 \dots s_n$ to a probability distribution over actions, with the property that for any action $a \in Act$ with $a \notin A(s_n)$ the probability of action a according to $\rho(\vec{s})$ is zero. The value of a policy is the expected sum of rewards obtained by an agent choosing actions according to the policy, discounted by the discount factor γ, when starting in the initial state. The reinforcement learning problem is to learn an optimal policy ρ^* that maximises the expected discounted future reward. Since by (Puterman 1994) there is always a Markovian optimal policy, we can define a *solution* to the reinforcement learning problem given an MDP M as an optimal policy of the form $\rho : S \to Pr(Act)$ where for each $s \in S$ and each action a, if $\rho(s)(a) > 0$, then $a \in A(s)$.

[1]For simplicity we consider only boolean features, but all our results extend to arbitrary features or even featureless MDPs.

2.2 Shields

Shields are an approach to guarantee satisfaction of safety constraints both during and after training (Alshiekh et al. 2018; ElSayed-Aly et al. 2021). Shields are synthesised from Safety LTL specifications, allowing them to express non-Markovian constraints over traces. Safety LTL (Manna and Pnueli 1995) is the fragment of LTL (Pnueli 1977) in which only the $\bigcirc$ ("*next*") and $\Box$ ("*always*") modalities are allowed. Shields come in two variants: *post-posed* and *preemptive*. Post-posed shields act *after* the agent has chosen an action to perform. When a post-posed shield detects that the chosen action would or could lead to a constraint violation, it modifies the action to a *safe* default one. Preemptive shields act *before* the agent has chosen an action to perform. A preemptive shield outputs, at each timestep, the set of actions it considers to be *safe*, from which the agent chooses one to perform. As PPAM is an action masking approach, we focus on preemptive shields, and we will refer to them as simply "shields" below.

Definition 2 (Shield (Alshiekh et al. 2018)) *A (preemptive) shield is a tuple* $\mathcal{S} = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$*, where:*

- Q *is the finite set of states;*
- q_0 *is the initial state;*
- Σ_I *is the input alphabet, where* $\Sigma_I = \Sigma_I^1 \times \Sigma_I^2$*;*
- $\Sigma_O = 2^{Act}$ *is the output alphabet;*
- $\delta : Q \times \Sigma_I \to Q$ *is the state transition function;*
- $\lambda : Q \times \Sigma_I^1 \to \Sigma_O$ *is the output function.*

An *MDP observer function* $f : S \to L$ maps states of the underlying MDP to observations from a set L. The input alphabet Σ_I is given by $L \times Act$, and $\Sigma_I^1 = L$.

A shield is built from a Safety LTL specification and an abstraction of the underlying MDP. The abstraction is a deterministic safety word automaton that rejects prefixes of sequences that do not correspond to possible runs of the underlying MDP (restricted to the relevant sets of features); essentially, it describes the physics of the system. Given an infinite run that is physically possible, it cycles through accepting states indefinitely. The Safety LTL specification is translated into a safety automaton (Kupferman and Vardi 2001). A safety automaton is a tuple $\mathcal{A}_s = (Q_s, q_{0_s}, \Sigma_s, \delta_s, F_s)$, where Q_s is the set of states of the automaton, q_{0_s} the initial state, Σ_s the input alphabet, $\delta_s : Q_s \times \Sigma_s \to Q_s$ the transition function, and $F_s \subseteq Q_s$ the set of "*safe*" states. An infinite word over Σ_s is accepted by a safety automaton if and only if the run induced by it visits only safe states, i.e., states in F_s.

The product of the safety automaton and the abstract MDP can be seen as a infinite game between the agent and the environment, where the agent wins if only *safe* states (i.e., states in which the Safety LTL specification is satisfied) are visited. By computing the winning region of the game, we know which states the agent is allowed to reach. The shield is extracted from the winning strategy in the game, and restricts the action space available to the agent (through its output function λ) to that specified by the winning strategy.

Synthesising a shield requires time double exponential in the Safety LTL specification and linear in the size of the abstracted MDP. Given a Safety LTL formula φ, it is possible

to build a safety automaton $\mathcal{A}_\varphi$ of size double exponential in the size of φ, such that $\tau, i \models \varphi$ iff $\tau_i \in L(\mathcal{A}_\varphi)$. Generating and solving the product game between the automaton and the abstracted MDP can be done in linear time through standard safety games-solving techniques (Mazala 2002).

As an illustration, we briefly recall the "hot water tank" scenario from (Alshiekh et al. 2018).

Example 1 *In the "hot water tank" scenario, the aim is to learn an energy-efficient controller for a hot water storage tank with a maximum capacity of 100 litres. A reward is associated with each water tank level, depending on how much energy is needed to keep the water in the tank hot. The outflow from the tank is always between 0 and 1 litres per second. The agent can open or close the tank intake valve. The inflow is between 1 and 2 litres per second when the valve is open. Whenever the value is opened or closed, the setting must be maintained for at least three seconds to prevent the valve from wearing out. The safety specification is that "*the tank water level must always be greater than 0 and less than 100, and if the valve is opened (closed) it should remain open (closed) for at least three seconds*". The safety constraints can be expressed using the following Safety LTL formula:*

$$\Box(level > 0) \wedge \Box(level < 100) \wedge$$
$$\Box((open \wedge \bigcirc close) \rightarrow \bigcirc \bigcirc close \wedge \bigcirc \bigcirc \bigcirc close) \wedge$$
$$\Box((close \wedge \bigcirc open) \rightarrow \bigcirc \bigcirc open \wedge \bigcirc \bigcirc \bigcirc open)$$

The first two conjuncts specify that the storage level should always be between 1 and 99 liters. The last two establish that whenever the inflow is switched off (respectively, on), then it remains so for the next three seconds.

3 Pure-Past Linear-time Temporal Logic

PPAM safety properties are expressed in Pure-Past Linear-time Temporal Logic (PPLTL) (De Giacomo et al. 2020). As the name suggests, PPLTL is the past-time version of LTL on finite traces, in which formulas are evaluated on the last state in the history looking back towards the beginning of the trace. As a result, the paths on which formulas are evaluated are naturally finite.

Given a set of propositional symbols (or "*fluents*") $Prop$, the formulas of PPLTL are defined by:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \ominus\varphi \mid \varphi \mathcal{S} \varphi$$

where $p \in Prop$. $\ominus\varphi$ means "in the previous state, φ" ('yesterday'), and $\varphi_1 \mathcal{S} \varphi_2$ means "φ_2 happened in the past, and since then φ_1 is true" ('since'). We also define the abbreviations $\diamondsuit\!\!-\varphi$ as $true\, \mathcal{S} \varphi$ (meaning "in some state in the past φ") and $\boxminus\varphi$ as $\neg\diamondsuit\!\!-\neg\varphi$ ("in all states in the past φ").

Formulas of PPLTL are interpreted over finite traces $\tau = \tau_0 \tau_1 \ldots \tau_{n-1}$ where τ_i at instant i is a propositional interpretation over $Prop$. $n = length(\tau)$ is the length of τ. Given τ, a PPLTL formula φ, and instant i, the relation $\tau, i \models \varphi$ (φ holds at i in τ) is defined inductively as follows:

- $\tau, i \models p$ iff $p \in \tau_i$;
- $\tau, i \models \neg\varphi$ iff $\tau, i \not\models \varphi$;
- $\tau, i \models \varphi_1 \wedge \varphi_2$ iff $\tau, i \models \varphi_1$ and $\tau, i \models \varphi_2$;
- $\tau, i \models \ominus\varphi$ iff $\tau, i-1 \models \varphi$, with $i > 0$, or $i = 0$ and $\varphi \equiv \bot$, i.e., φ is a contradiction;
- $\tau, i \models \varphi_1 \mathcal{S} \varphi_2$ iff there exists k with $0 \leq k \leq i$ such that $\tau, k \models \varphi_2$ and for all j with $k < j \leq i$, $\tau, j \models \varphi_1$.

Given a PPLTL formula φ, we will write $\tau \models \varphi$ as an abbreviation of $\tau, length(\tau) \models \varphi$.

In order to evaluate a PPLTL formula φ on a trace τ, it is sufficient to know the truth values of the subformulas of φ in the current and previous timestep; formally:

Proposition 1 *Let φ be a PPLTL formula and $Subf(\varphi)$ the set of its subformulas. Moreover, let τ and τ' be two finite traces of length n and n' respectively, such that $\tau_n = \tau'_{n'}$ and $\tau, n-1 \models \psi$ iff $\tau', n'-1 \models \psi$ for each $\psi \in Subf(\varphi)$. Then, it holds that $\tau, n \models \varphi$ iff $\tau', n' \models \varphi$.*

This claim follows from results in (De Giacomo, Favorito, and Fuggitti 2022) which are stated in a somewhat different form.

Due to the fact that $Subf(\varphi)$ is linear in φ itself, the proposition above implies the following.

Observation 1 *Evaluating a PPLTL formula φ on a trace τ can be done in time linear in $|\varphi|$ and constant in $length(\tau)$, given that the truth values of subformulas of φ in the preceding state are known.*

Crucially, this allows us to evaluate the truth value of a PPLTL specification using only a set of truth values that is linear in size of the formula, regardless of how many timesteps have elapsed.

4 Pure-Past Action Masking

In this section, we introduce Pure-Past Action Masking (PPAM). In PPAM, a safety specification comprises a set of PPLTL formulas evaluated on the history which specify the set of actions allowed given the history. For each action $a \in Act$, a PPAM mask contains a PPLTL formula φ_a over the mask's features (fluents) $Prop$, and the action a is available if and only if φ_a is true. Notice how there is no coupling between the set of features available to the agent and those available to the mask.

Definition 3 (PPAM mask) *Given a (learning) agent modelled by an MDP $M_{ag} = (F, S, s_0, Act, A, Tr, R, \gamma)$, a PPAM mask $(\mathcal{L}, \{\varphi_a : a \in Act\})$ is a pair where:*

- *$\mathcal{L} = 2^{Prop}$ is the set of states of the mask;*
- *$\{\varphi_a : a \in Act\}$ is a set containing a PPLTL formula φ_a over $Prop$ for each action $a \in Act$, specifying when a is permitted.*

As an example, we show how the safety specification for the "hot water tank" scenario can be expressed using PPAM action specifications.

Example 2 *The "hot water tank" safety specification (see Example 1) can be expressed using the following PPLTL action specifications:*

$$\varphi_{open} = level \leq 93 \wedge (close \rightarrow \ominus close \wedge \ominus \ominus close)$$

$$\varphi_{close} = level \geq 4 \wedge (open \rightarrow \ominus open \wedge \ominus \ominus open)$$

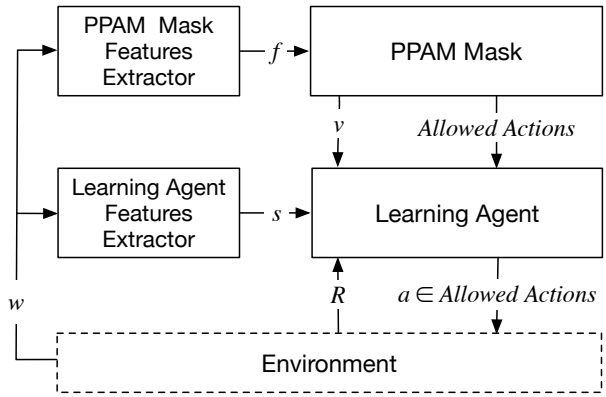

Figure 1: Interaction between the environment, agent and the PPAM mask. $Allowed\ Actions \subseteq A(s)$ is the set of permitted actions output by the PPAM mask, $v \in \mathcal{L}$ is the PPAM mask's state, $a \in Allowed\ Actions$ is the action chosen by the agent, R is the reward given by the environment after a is performed, and w is the resulting state of the environment. From w, we obtain the MDP state s through the learning agent's feature extractor, and f through the feature extractor of the PPAM mask (notice these need not be aligned).

We now state the learning problem for PPAM.

Definition 4 (PPAM Learning Problem) *An instance of an RL problem with a PPAM mask is a pair $M_{ag}^{ppam} = \langle M_{ag}, PPAM \rangle$ where $M_{ag} = (F, S, s_0, Act, A, Tr, R, \gamma)$ is a factorized MDP with Tr and R hidden, and $PPAM = (\mathcal{L}, \{\varphi_a : a \in Act\})$ is a PPAM mask with a set of PPLTL formulas φ_a over Prop, where $\mathcal{L} = 2^{Prop}$.*

A solution to the problem is a policy $\rho^* : (S \times \mathcal{L})^+ \to Pr(Act)$ that maximises the expected discounted cumulative reward and conforms to the PPAM mask's specification, that is, for every $\tau \in (S \times \mathcal{L})^+$, $\rho^*(\tau)(a) > 0$ only if $\tau' \models \varphi_a$ where τ' is τ projected on the second component (the sequence of sets of mask fluents).

Note that the solution is a policy in which all actions are safe (an action only has non-zero probability if the corresponding 'precondition' formula holds). In the following, we will assume that the PPAM learning problem is such that, for any possible trace that can be generated, there is always at least one action that can be taken, i.e., that the problem has at least one solution.

We now show how to solve the PPAM learning problem. We begin by lifting the evaluation of formulas on sequences of mask states from atomic fluents to all subformulas of mask formulas. By $Subf(\varphi)$ we denote the set of subformulas of φ (nodes in its parse tree). Let $V = 2^{\{Subf(\varphi_a):a \in Act\}}$ be the set of all possible truth assignments to $\{Subf(\varphi_a) : a \in Act\}$. Given a sequence $\tau \in \mathcal{L}^+$, and the valuation of fluents in each state, it is possible to evaluate formulas from $v \in V$ on τ according to the truth definition for PPLTL formulas. In fact, as we have seen earlier, it is sufficient to know the valuation of $\{Subf(\varphi_a) : a \in Act\}$ in the preceding state and the valuation of fluents in the current state in order to evaluate $\{Subf(\varphi_a) : a \in Act\}$.

Note that each $v \in V$ determines the set of actions that are permitted given the mask's specification (those $a \in Act$ where $\varphi_a \in v$, that is, φ_a is true). We are going to use this to reduce the non-Markovian PPAM learning problem to a Markovian learning problem.

Below we prove (Theorem 1) that the PPAM learning problem always has a Markovian solution $\rho : S \times V \to Pr(Act)$ that maximises the expected discounted cumulative reward and is safe (only actions permitted by the mask are used). The transition to a Markovian policy is possible because each $v \in V$ encodes sufficient information about the history of the system.

First, we define an important construction used in the proof of Theorem 1. It is an MDP that is essentially a product of the original MDP and the mask's states and transitions.

Definition 5 (Product MDP) *Given an instance of a PPAM learning problem $M_{ag}^{ppam} = \langle M_{ag}, PPAM \rangle$ with $M_{ag} = (F, S, s_0, Act, A, Tr, R, \gamma)$ and $PPAM = (\mathcal{L}, \{\varphi_a : a \in Act\})$, we define a new product MDP $M_{ag \times ppam} = (F', S', s_0', Act', A', Tr', R', \gamma)$, where*

- *$F' = F$;*
- *$S' = S \times V$ where $V = 2^{\{Subf(\varphi_a):a \in Act\}}$ (states are pairs of a state in S and a set of mask's subformulas);*
- *$s_0' = (s_0, v_0)$ where v_0 contains the set ℓ_0 of mask's fluents true in the corresponding state of the world and $\{\psi \in Subf(\varphi_a) : a \in Act \text{ and } \ell_0 \models \psi\}$;*
- *$Act' = Act$;*
- *$A'((s, v)) = A(s) \cap \{a : \varphi_a \in v\}$ (only safe actions are available);*
- *$Tr' : S \times V \times Act \rightharpoonup Pr(S \times V)$ is defined as follows. As in (De Giacomo et al. 2019), we assume that there is a probability distribution $Tr^{ag \times ppam} : S \times \mathcal{L} \times Act \rightharpoonup Pr(S \times \mathcal{L})$ induced by the world (since the mask's fluents are caused by the agent acting in the world). This probability distribution is unknown to the agent, just as Tr is unknown. We lift the probability distribution $Tr^{ag \times ppam}$ to $Tr' : S \times V \times Act \rightharpoonup Pr(S \times V)$ as follows. The probability of transitioning from (s, v) to (s', v') on executing action a is the same as the probability given by $Tr^{ag \times ppam}$ of transitioning from (s, f) to (s', f') by a, where f is the set of fluents in v, and v' is the maximal subset of $\{Subf(\varphi_a) : a \in Act\}$ that is true given the set of formulas true in ('the previous set') v and the 'current' set of fluents f'. This set v' is unique by Proposition 1[2];*
- *$R'((s, v), a, (s', v')) = R(s, a, s')$.*

Note that the size of each state (s, v) is linear in the size of s and the size of the mask. The key feature of this product MDP is that only safe (permitted by the mask) actions are available to the agent. We prove in Theorem 1 below that learning on the product MDP with access to only the safe actions is sufficient for solving the PPAM learning problem. Critically, the optimality of learning with only safe actions is *entailed by the proof*: it is not part of the definition of the PPAM learning problem.

[2]Observe that the construction of the transition function in the product MDP is the reason to include subformulas of φ_a in v, and the transition function can be computed in linear time by Observation 1.

Theorem 1 *An RL problem with a PPAM mask $M_{ag}^{ppam} = \langle M_{ag}, PPAM \rangle$, with $M_{ag} = (S, s_0, Act, A, Tr, R, \gamma)$ and $PPAM = (\mathcal{L}, \{\varphi_a : a \in Act\})$, can be reduced to RL over the product MDP $M_{ag \times ppam}$ such that optimal policies for M_{ag}^{ppam} can be learned by learning corresponding optimal Markovian policies for $M_{ag \times ppam}$.*

Proof. First, observe that the actions available to the agent in M_{ag}^{ppam} after each $(s_0, \ell_0) \dots (s_n, \ell_n)$ (where ℓ_i are sets of fluents at timestep i) are the same as the actions available to the agent in the MDP $M_{ag \times ppam}$ in (s_n, v_n) where v_n is the set of $\psi \in \{Subf(\varphi_a) : a \in Act\}$ such that $\ell_0 \dots \ell_n \models \psi$. This is because the evaluation of formulas from $\{Subf(\varphi_a) : a \in Act\}$ including φ_a for each $a \in Act$ is the same in $(s_0, \ell_0) \dots (s_n, \ell_n)$ and in (s_n, v_n). Hence, instead of masking actions based on $(s_0, \ell_0) \dots (s_n, \ell_n)$, we can mask them depending on (s_n, v_n) without loss of information. Note that the rewards are the same in both MDPs. This means that the reward for performing a after $(s_0, \ell_0) \dots (s_n, \ell_n)$ is the same as the reward for performing a in (s, v_n). Since $Tr' : S \times V \times Act \rightharpoonup Pr(S \times V)$ corresponds to $Tr^{ag \times ppam} : S \times \mathcal{L} \times Act \rightharpoonup Pr(S \times \mathcal{L})$, the optimal non-Markovian policy $\rho^* : (S \times \mathcal{L})^+ \to Pr(Act)$ produces exactly the same reward as a Markovian policy $\rho : S \times V \to Pr(Act)$ obtained by replacing $\ell_0, \dots, \ell_n$ with the corresponding v_n. □

Note that, by construction, the learned policy is guaranteed to conform to the PPAM mask specification (since it is learned in the product MDP $M_{ag \times ppam}$ where only safe actions are available) and to be optimal (among policies that conform to the PPAM mask specifications). In other words, this result tells us that PPAM masks do not restrict agents from finding optimal (safe) policies for the original learning problem, while at the same time automatically guaranteeing safety.

Moreover, the input to the learning algorithm is the product MDP $M_{ag \times ppam}$. In the product MDP, the number of new features added to the states is linear with respect to the size of the mask's specification, as the new feature space is exactly the union of F (the feature space of M_{ag}) and $\{Subf(\varphi_a) : a \in Act\}$. This leads to the following corollary regarding the complexity of using PPAM.

Corollary 1 *RL with a PPAM mask incurs a single exponential, in the size of the mask specification, overhead in the size of the state space. Moreover, if the MDP is factorized, a PPAM mask leads to a linear, again in the size of the mask specification, overhead in the size of the feature space.*

In practice, the increase in the size of the state space may be lower than the single exponential blowup in Corollary 1, as the truth value of some subformulas may imply that of other subformulas.

5 Comparison with Shields

In this section, we briefly compare the expressive power of PPAM and shields. Such a comparison is not entirely straightforward. Shields are built using an abstraction of the underlying MDP that describes the dynamics of the learning environment. In contrast, the set of fluents used to specify PPAM safety constraints are independent of the features of the MDP seen by the learning agent. For the comparison to be meaningful, we must assume that the states in the abstract MDP used to build the shield are a subset of the fluents available to the PPAM.

Let $\mathcal{S} = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$ be a shield synthesised from some safety automaton (Safety LTL specification) $\mathcal{A}_s$, where $Q = Q_{\mathcal{M}} \times Q_s$ is the set of states, $Q_{\mathcal{M}}$ is the set of states of the MDP abstraction, Q_s is the sets of states of the safety automaton used to synthesise $\mathcal{S}$, and $\Sigma_I = L \times Act$.

First, we prove a lemma connecting DFAs, built from (Safety) LTL formulas, and PPLTL.

Lemma 1 *For any state q_s of the safety automaton $\mathcal{A}_s$ used to synthesise the shield $\mathcal{S}$, it is possible to provide a PPLTL formula ϕ_{q_s} describing exactly the paths leading to q_s from the initial state of $\mathcal{A}_s$.*

Proof. As the safety automaton $\mathcal{A}_s$ is synthesised from a (Safety) LTL formula, it must be counter-free (Manna and Pnueli 1990). The claim then follows via a result from (Zuck 1986), showing that for any given counter-free automaton it is possible to build a PPLTL formula ϕ_q recognizing all paths leading to any of its states q. □

Notice that, if the MDP abstraction is a counter-free automaton, then we could apply Lemma 1 to obtain a PPLTL formula for each of its states $q_{\mathcal{M}}$. For generality, we assume that it is an arbitrary DFA and include the set of states $Q_{\mathcal{M}}$ of the MDP abstraction amongst the fluents that we can use to define our PPLTL formulas.

The next lemma is crucial for stating our expressiveness result. We assume that we have access to the state of the MDP abstraction, the observation from L and the action executed by the agent, i.e., that the set of fluents is $Prop = Q_{\mathcal{M}} \cup L \cup Act$.[3] We show how, for each action $a \in Act$, we can build a PPLTL formula φ_a that will constrain a in exactly the same way as $\mathcal{S}$.

Lemma 2 *Given a shield $\mathcal{S}$, for any action $a \in Act$ we can build a PPLTL formula φ_a over fluents $Prop = Q_{\mathcal{M}} \cup L \cup Act$ such that, for any shield state $(q_{\mathcal{M}}, q_s)$ and observation ℓ reached by $\mathcal{S}$ after some trace τ, $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$ if and only if $\tau \models \varphi_a$.*

Proof. First, let $a \in Act$ be an arbitrary action. If an MDP abstract state-observation pair $(q_{\mathcal{M}}, \ell)$ is such that $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$ for any safety automaton state q_s (reachable from the initial state q_0) and for which ℓ is observable from $(q_{\mathcal{M}}, q_s)$, then $q_{\mathcal{M}}$ and ℓ being the current MDP abstraction state and observation is a sufficient precondition to allow the agent to take action a. Thus, we define Λ^a as the set of all such MDP abstraction state-observation pairs.

Now, let $q_{\mathcal{M}}$ be an MDP abstraction state and ℓ an observation such that there are two distinct safety automaton states q_s, q_s' for which: (i) both q_s and q_s' are reachable from

[3] We implicitly assume that finite traces we consider when evaluating the formula φ_a will be of the form $(q_{\mathcal{M}}^0, \ell^0) a^0 (q_{\mathcal{M}}^1, \ell^1) a^1 \dots (q_{\mathcal{M}}^n, \ell^n)$, where a^i is the action taken after $(q_{\mathcal{M}}^i, \ell^i)$.

q_0, (ii), ℓ can be observed in both $(q_{\mathcal{M}}, q_s)$ and $(q_{\mathcal{M}}, q'_s)$, and (iii), $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$ and $a \notin \lambda((q_{\mathcal{M}}, q'_s), \ell)$. For $q_{\mathcal{M}}$ and ℓ, we need to keep track of the current state q_s in the safety automaton, so that we allow a if and only if the current state q_s is such that $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$. Let $Q_s^{a,q_{\mathcal{M}},\ell} = \{q_s \in Q_s : a \in \lambda((q_{\mathcal{M}}, q_s), \ell)\}$. Thanks to Lemma 1, we can build for each of these $q_s \in Q_s^{a,q_{\mathcal{M}},\ell}$ a PPLTL formula ϕ_{q_s} that is satisfied if and only if, given the current history, the safety automaton $\mathcal{A}_s$ is in state q_s.[4]

We are now ready to define the PPLTL formula φ_a that will constrain action a. In the formula we implicitly omit all the $(q_{\mathcal{M}}, \ell) \in (Q_{\mathcal{M}} \times L) \setminus \Lambda^a$ such that $Q_s^{a,q_{\mathcal{M}},\ell} = \emptyset$.

$$\varphi_a = \bigvee_{(q_{\mathcal{M}},\ell)\in\Lambda^a} (q_{\mathcal{M}} \wedge \ell) \vee \bigvee_{(q_{\mathcal{M}},\ell)\in(Q_{\mathcal{M}}\times L)\setminus\Lambda^a} \left(q_{\mathcal{M}} \wedge \ell \wedge \left(\bigvee_{q_s \in Q_s^{a,q_{\mathcal{M}},\ell}} \phi_{q_s} \right) \right)$$

We now show that action a is allowed by the shield $\mathcal{S}$ if and only if the current trace τ satisfies φ_a. Let $(q_{\mathcal{M}}, q_s)$ be the current state of the shield and ℓ the current observation, reached after following the trace τ. Observe that for $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$, we either have that $a \in \lambda((q_{\mathcal{M}}, q'_s), \ell)$ for any state of the safety automaton q'_s such that $(q_{\mathcal{M}}, q'_s)$ is reachable from $q_{0,\mathcal{S}}$ and ℓ is observable at $(q_{\mathcal{M}}, q'_s)$, or not.

In the first case, it means that whenever the state of the MDP abstraction is $q_{\mathcal{M}}$ and the observation is ℓ, the shield will always allow action a. Notice that the fluent for state $q_{\mathcal{M}}$ and for observation ℓ are true in the last timestep of trace τ if and only if $q_{\mathcal{M}}$ is the current state of the MDP abstraction and ℓ the current observation. Hence, $\mathcal{S}$ allows action a if and only if $\tau \models \varphi_a$, since $\tau \models q_{\mathcal{M}} \wedge \ell$ for $(q_{\mathcal{M}}, \ell) \in \Lambda^a$.

In the second case, it means that the current safety automaton state q_s is such that, when coupled with the current MDP abstraction state $q_{\mathcal{M}}$ and observation ℓ, $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$. This is true if and only if the trace τ is such that: (i) $q_{\mathcal{M}}$ and ℓ are true at its final timestep, and (ii), $\tau \models \phi_{q_s}$, by Lemma 1. Hence, we have again that $\mathcal{S}$ allows action a if and only if $\tau \models \varphi_a$, since $\tau \models q_{\mathcal{M}} \wedge \ell \wedge \left(\bigvee_{q_s \in Q_s^{a,q_{\mathcal{M}}}} \phi_{q_s}\right)$ for $(q_{\mathcal{M}}, \ell) \in (Q_{\mathcal{M}} \times L) \setminus \Lambda^a$ and $q_s \in Q_s^{a,q_{\mathcal{M}},\ell}$. □

Theorem 2 *For every shield $\mathcal{S}$ for a safety property φ, there exists a PPAM mask over fluents $Prop = Q_{\mathcal{M}} \cup L \cup Act$ such that the policy learned with it will satisfy φ.*

Proof. For each action $a \in Act$, we build the formula φ_a from Lemma 2. Then, it is easy to see that the resulting PPAM mask constrains actions just like the input shield $\mathcal{S}$ does. Therefore, the agent learns exactly the same policy. □

Theorem 2 shows that, when provided with the same information, PPAM masks have the same expressive power as shields in constraining actions.

[4]Notice that the safety automaton transitions on inputs from $L \times Act$, hence we also need Act to be included in the set of fluents available to define the PPLTL formula φ_a.

We conclude this section by briefly discussing some practical considerations relating to the use of PPAMs and shields. PPAMs require that the conditions for the safe execution of each action are specified directly by the designer of the mask. In contrast, with shields, the safety property is specified at system behavior level, and the constraints on actions are synthesised during the construction of the shield. While this avoids the designer of the shield having to specify action conditions, construction of the shield requires that an abstraction of the underlying MDP is available (which may be challenging for cyber-physical systems that learn in the real environment), and entails significant computational overhead compared to PPAM masks. PPAMs are therefore more appropriate in situations where there is a straightforward correspondence between safety specifications and actions, and/or an abstraction of the underlying MDP is unavailable. Another clear difference between the two approaches follows from the logics that they employ, i.e., Safety LTL and PPLTL. Safety LTL has only *future* modalities, and as result, shields are more appropriate when the specification constrains how the system can evolve in the future. On the other hand, PPLTL (as the name suggests) has only *past* modalities, and so PPAMs are more appropriate when the action constraints depend on what has happened in the past.

6 Experimental Evaluation

In this section, we illustrate our approach with two examples from the literature, COCKTAILPARTY (De Giacomo et al. 2019) and BOATRACE (Leike et al. 2017), and present experimental results which show that, in addition to ensuring safety, PPAMs can also improve sample efficiency. Code and more details about the environments is available in a GitHub repository at: github.com/giovannivarr/PPAM-AAAI24/. The repository also includes a PPAM implementation of the WATERTANK environment from (Alshiekh et al. 2018). We have not not included the WATERTANK results here, as they follow directly from Theorem 2 in the previous section: both the PPAM- and shield-based agents are constrained in the same way, and learn exactly the same policies. All experiments were performed on an Apple M1 chip with a 16 GB RAM, macOS Ventura 13.5 laptop.

6.1 COCKTAILPARTY

In the COCKTAILPARTY scenario the agent learns with a *restraining bolt* (RB), which provides additional rewards to the agent depending on whether the safety specification is satisfied. However, restraining bolts do not provide any safety guarantees, either during training or deployment. In the scenario, the agent is a robot which serves drinks (coke and beer) and snacks (biscuits and chips) to four guests at a party. The environment is a 5×5 grid. The robot knows the locations of drinks and snacks and the locations of people, and can execute actions to move in the environment, grasp items and serve them to people. The robot gets a reward when a delivery task is completed. Some of the guests are children who should not be served alcohol, and the robot should not serve more than one drink or snack to each guest. However, as the robot is unable to distinguish different kinds

of people and has no memory of who has been served previously, it will simply learn to bring a drink or snack to any person (choosing the shortest path).

To enforce the constraint that children should not be served alcohol and each guest should be served one drink or snack, we use a PPAM mask to constrain the "`serve`" action, using the following PPLTL formula:

$$\varphi_{\texttt{serve}} = \bigvee_{g \in Guests} [at\_g \wedge ((\neg \diamondsuit served\_food\_g \wedge holding\_food) \vee (\neg \diamondsuit served\_drink\_g \wedge holding\_drink \wedge (\neg minor\_g \vee \neg holding\_alcohol)))]$$

where at_g is true if the robot is in the cell occupied by guest g, $holding\_food$ is true if the robot is holding a snack in the current timestep, $holding\_drink$ is true if the robot is holding a drink, $holding\_alcohol$ is true if the robot is holding an alcoholic drink, $minor\_g$ is true if guest g is a minor, $served\_food\_g$ is true if guest g has been served food previously, and $served\_drink\_g$ is true if the guest has been served a drink previously.

We use the same experimental setup as in (De Giacomo et al. 2019): both agents are trained using n-step Sarsa, configured with learning rate $\gamma = 0.999$, exploration rate $\epsilon = 0.2$, and $n = 100$. We performed 10 runs, each consisting of 4000 iterations lasting at most 1000 steps. Each run is split into non-overlapping intervals containing 100 consecutive iterations. For each such interval, we evaluate the average reward obtained by the agent. Figure 2 shows the median average rewards per interval and the shaded areas enclose the 25^{th} and 75^{th} percentiles. As can be seen, the PRB agent achieves a higher median average reward and learns to complete all four serving tasks more quickly than the RB agent. This is consistent with the observation in (Huang and Ontañón 2022), that the improvement in sample efficiency is due to the fact that the PPAM agent has fewer actions to choose from. Moreover, the median number of violations of the safety specification by the RB agent was 164.5.

6.2 BoatRace

BoatRace is an "AI Safety Gridworld" (Leike et al. 2017) designed to illustrate 'reward gaming' (Skalse et al. 2022), a general phenomenon where an agent exploits an unintended loophole in the reward specification to obtain rewards in unintended (and possibly unsafe) ways. The environment is a 5×5 grid, and the the agent sails around a track, going past 4 different checkpoints, one for each cardinal direction. The agent should learn to complete as many clockwise laps as possible (performance wrt the goal is measured as the number of clockwise movements minus the number of counter-clockwise movements the agent has taken across the entirety of the episode). The agent receives a reward of -1 for each step taken, and, if it arrives on a checkpoint from the "correct" clockwise direction, it also receives a reward of 3. As a result, the agent can maximize its reward by stepping back and forth on a checkpoint. This is not the desired behavior and may be unsafe, e.g., if there are other agents in the environment.

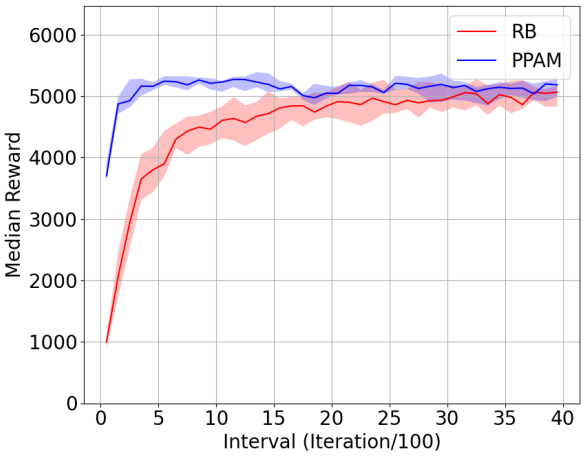

Figure 2: Median reward of PPAM and RB agents.

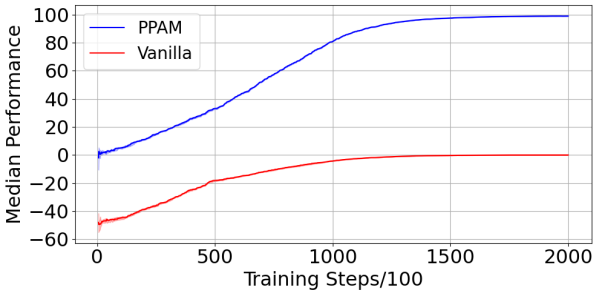

Figure 3: Median performance of PPAM and Q agents.

To ensure correct behavior, we use a PPAM that constrains the agent's movement actions. The PPLTL formula constraining the "`move_west`"' action is:

$$\varphi_{\texttt{move\_west}} = (\neg(north \vee west))\mathcal{S}(south \vee east)$$

where the fluent corresponding to each cardinal direction is true whenever the agent moves onto the corresponding checkpoint in the track. The constraint checks that the last checkpoint that was visited was either east or south, as after these the agent can move west. The PPLTL formulas for the other movement actions are analogous: the agent is allowed to perform the `move_north` action only if the last checkpoint visited was either west or south; the agent is allowed to perform the `move_south` action only if the last checkpoint was either north or east, and the agent is allowed to perform the `move_east` action, only if the last checkpoint was either north or west.

We compare a Q-learning agent trained with a PPAM to a vanilla Q-learning agent. Both agents are configured with learning rate $\gamma = 0.5$, and initial exploration rate $\epsilon = 1$, linearly annealed to 0.01 over 100000 training steps. We performed 10 runs, each consisting of 200000 training steps. The maximum performance obtainable is 100. Lines in Figure 3 show the median performance during training.

As can be seen, the PPAM-based agent learns the optimal policy with respect to the performance function, i.e., it learns how to correctly complete laps. In contrast, the vanilla agent achieves a performance score of 0, meaning that it performs as many clockwise and counter-clockwise movements: in other words, it has learned to exploit the reward function and moves back and forth over a single checkpoint. Note that the poor performance of the non-PPAM agent is not due to the use of Q-learning. In (Leike et al. 2017) agents were trained using deep RL algorithms (Rainbow (Hessel et al. 2018) and A2C (Mnih et al. 2016)), but did not manage to learn a policy that optimizes performance in this environment.

7 Related Work

Markovian action masking is widely used in reinforcement learning, both for efficiency (removing invalid actions) and for removing unsafe actions.

Huang and Ontañón (2022) focus mainly on masking in deep reinforcement learning. They give theoretical foundations for the use of action masking in policy gradient methods, and show that it generates a valid policy gradient. Using a simplified real-time strategy game they show experimentally that invalid action masking scales better with respect to the size of the action space than *invalid action penalty*, which penalizes agents for taking invalid actions.

Kanervisto, Scheller, and Hautamäki (2020) classify action masking (which they call "*remove actions*") as a category of "*action space shaping*" approaches. They present ablation studies in game domains, showing the positive effect of action masking during learning on sample efficiency.

Krasowski et al. (2023) compare three approaches to provably safe RL in experiments where a set of safety constraints must be satisfied. Their results show that provably safe approaches are able to converge to optimal policies in roughly the same time as other approaches, while ensuring that safety constraints are satisfied. However, unlike in our approach, they provide no formal guarantee that removing actions does not eliminate optimal safe policies.

Preemptive shields (Alshiekh et al. 2018; ElSayed-Aly et al. 2021) can be seen as a form of non-Markovian action masking. We have already compared our approach to shields in Section 5. Other logic-based approaches to enforcing non-Markovian constraints include (Wen, Ehlers, and Topcu 2015) and (Mason et al. 2017).

Similarly to shields, Wen, Ehlers, and Topcu (2015) propose an approach that requires the transformation of a Safety LTL formula into an automaton. They then synthesise a maximally permissive strategy using the underlying MDP (rather than an abstract MDP, as in shields). Their approach is therefore double exponential in the input formula in general. However, the complexity reduces to linear for a subset of Safety LTL formulas (formulas of the form $\varphi_0 \wedge \Box\varphi_1$ where φ_0 and φ_1 are Boolean combinations of atomic propositions, and formulas of the form $\bigcirc q$ where q is an atomic proposition).

Mason et al. (2017) present an approach in which they build an abstract MDP from the MDP in which the agent learns. In the abstract MDP, they synthesise abstract policies that satisfy constraints expressed in Probabilistic Computation Tree Logic (PCTL) (Hansson and Jonsson 1994); the abstract policies are then used to constrain the set of actions the agent can perform when learning in the original MDP. If the abstract MDP correctly captures the dynamics of the underlying MDP, their approach guarantees that any abstract policy returned satisfies the constraints and is Pareto optimal with respect to the candidate policies explored; however, it is not guaranteed to find a policy if one exists.

8 Conclusions and Future Work

We have presented *pure-past action masking*: a lightweight, modular formalism for specifying safe or permitted actions of reinforcement learning agents. The mask specification comprises a set of PPLTL formulas evaluated on the history, and specifies the set of actions permitted in the current state. As formulas are evaluated on the history rather than just the current state, PPAM enables non-Markovian action masking. The PPAM specification is enforced by construction: a PPAM agent will always learn an optimal policy which never violates its safety specification during either training or execution. PPAM can be applied in any setting where the fluents relevant to the safety specification can be perceived by the mask, and where the conditions on action availability can be expressed in PPLTL, which is as expressive as Safety LTL. Moreover, using PPAM incurs only a single exponential blowup wrt the size of the PPLTL formula.

In future work, we plan to extend the framework of PPAM to multi-agent reinforcement learning, as has been done with shields (ElSayed-Aly et al. 2021). We would also like to investigate possible ways of implementing PPAM in environments with continuous action spaces, as in, e.g., (Krasowski et al. 2023). Finally, another interesting extension would be to employ PPAMs in partially observable MDPs (POMDPs). As PPAMs make no assumptions about the underlying structure of the (PO)MDP, if the PPAM's fluents are fully observable, then this trivially reduces to the present work. If the PPAM's fluents are not full observable, we can build on work on evaluating temporal logic formulas on incomplete traces, e.g., (Joshi, Tchamgoue, and Fischmeister 2017).

Acknowledgments

This work was supported by PNRR MUR project PE0000013-FAIR, partially supported by ERC Advanced Grant WhiteMech (No. 834228), EU ICT-48 2020 project TAILOR (No. 952215), the ONRG project N62909-22-1-2005, the InDAM-GNCS project "Strategic Reasoning in Mechanism Design", and the project OCENW.M.21.377 funded by the Dutch Research Council (NWO). For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

References

Alechina, N.; Bulling, N.; Dastani, M.; and Logan, B. 2015. Practical Run-Time Norm Enforcement with Bounded Lookahead. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, 443–451. International Foundation for Autonomous Agents and Multiagent Systems.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI 2018)*, 2669–2678. AAAI Press.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P. F.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *CoRR*, abs/1606.06565.

Bacchus, F.; Boutilier, C.; and Grove, A. J. 1996. Rewarding Behaviors. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996)*. AAAI Press.

Cheng, R.; Orosz, G.; Murray, R. M.; and Burdick, J. W. 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 19)*, 3387–3395. AAAI Press.

De Giacomo, G.; Favorito, M.; and Fuggitti, F. 2022. Planning for Temporally Extended Goals in Pure-Past Linear Temporal Logic: A Polynomial Reduction to Standard Planning. *CoRR*, abs/2204.09960.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 128–136. AAAI Press.

De Giacomo, G.; Stasio, A. D.; Fuggitti, F.; and Rubin, S. 2020. Pure-Past Linear Temporal and Dynamic Logic on Finite Traces. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, 4959–4965. International Joint Conferences on Artificial Intelligence Organization.

ElSayed-Aly, I.; Bharadwaj, S.; Amato, C.; Ehlers, R.; Topcu, U.; and Feng, L. 2021. Safe Multi-Agent Reinforcement Learning via Shielding. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (IJCAI 2021)*, 483–491. International Foundation for Autonomous Agents and Multiagent Systems.

Gabaldon, A. 2011. Non-Markovian control in the Situation Calculus. *Artificial Intelligence*, 175(1): 25–48.

Hadfield-Menell, D.; Dragan, A. D.; Abbeel, P.; and Russell, S. 2017. The Off-Switch Game. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, (IJCAI 2017)*, 220–227. International Foundation for Autonomous Agents and Multiagent Systems.

Hansson, H.; and Jonsson, B. 1994. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6(5): 512–535.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, 3215–3222. AAAI Press.

Huang, S.; and Ontañón, S. 2022. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. In *The International FLAIRS Conference Proceedings, 35 (FLAIRS-35)*. Florida Online Journals.

Hunt, N.; Fulton, N.; Magliacane, S.; Hoang, T. N.; Das, S.; and Solar-Lezama, A. 2021. Verifiably Safe Exploration for End-to-End Reinforcement Learning. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (HSCC 2021)*, 1–11. Association for Computing Machinery.

Joshi, Y.; Tchamgoue, G. M.; and Fischmeister, S. 2017. Runtime verification of LTL on lossy traces. In *Proceedings of the Symposium on Applied Computing (SAC 2017)*, 1379–1386. Association for Computing Machinery.

Kanervisto, A.; Scheller, C.; and Hautamäki, V. 2020. Action space shaping in deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, 479–486. IEEE.

Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.-M.; Lam, V.-D.; Bewley, A.; and Shah, A. 2019. Learning to Drive in a Day. In *2019 International Conference on Robotics and Automation (ICRA)*, 8248–8254. IEEE.

Krasowski, H.; Thumm, J.; Müller, M.; Schäfer, L.; Wang, X.; and Althoff, M. 2023. Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking. arXiv:2205.06750.

Krasowski, H.; Wang, X.; and Althoff, M. 2020. Safe reinforcement learning for autonomous lane changing using set-based prediction. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–7. IEEE.

Kupferman, O.; and Vardi, M. Y. 2001. Model Checking of Safety Properties. *Formal Methods in System Design*, 19(3): 291–314.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI Safety Gridworlds. *CoRR*, abs/1711.09883.

Lichtenstein, O.; Pnueli, A.; and Zuck, L. 1985. The glory of the past. In *Logics of Programs*, 196–218. Springer Berlin Heidelberg.

Manna, Z.; and Pnueli, A. 1990. A Hierarchy of Temporal Properties (Invited Paper, 1989). In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing (PODC 1990)*, 377–410. Association for Computing Machinery.

Manna, Z.; and Pnueli, A. 1995. *Temporal verification of reactive systems: safety*. Springer New York.

Mason, G. R.; Calinescu, R. C.; Kudenko, D.; and Banks, A. 2017. Assured reinforcement learning with formally verified abstract policies. In *9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*. York.

Mazala, R. 2002. Infinite Games. In Grädel, E.; Thomas, W.; and Wilke, T., eds., *Automata Logics, and Infinite Games: A Guide to Current Research*. Springer Berlin Heidelberg.

Mirchevska, B.; Pek, C.; Werling, M.; Althoff, M.; and Boedecker, J. 2018. High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2156–2162. IEEE.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, 1928–1937. JMLR.

Orseau, L.; and Armstrong, S. 2016. Safely Interruptible Agents. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI 2016)*, 557–566. AUAI Press.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, 46–57. IEEE.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Savage, T.; Zhang, D.; Mowbray, M.; and Chanona, E. A. D. R. 2021. Model-free safe reinforcement learning for chemical processes using Gaussian processes. *IFAC-PapersOnLine*, 54(3): 504–509.

Skalse, J.; Howe, N.; Krasheninnikov, D.; and Krueger, D. 2022. Defining and Characterizing Reward Gaming. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 9460–9471. Curran Associates, Inc.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Wen, M.; Ehlers, R.; and Topcu, U. 2015. Correct-by-synthesis reinforcement learning with temporal logic constraints. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4983–4990. IEEE.

Zhou, Z.; Li, X.; and Zare, R. N. 2017. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12): 1337–1344.

Zuck, L. 1986. *Past temporal logic*. The Weizmann Institute of Science.

Appendix A: Proof of Proposition 1

Proof. The proof proceeds by induction on the structure of φ. We have the following.

- $\varphi = p$ for some atomic proposition p. Then, $\tau, n \models p$ iff $p \in \tau_n = \tau'_{n'}$ iff $\tau', n' \models p$;
- $\varphi = \neg\psi$. Then, $\tau, n \models \neg\psi$ iff $\tau, n \not\models \psi$ iff, by induction hypothesis $\tau', n' \not\models \psi$ iff $\tau', n' \models \neg\psi$;
- Similarly for the other boolean cases;
- $\varphi = \ominus\psi$. Then, $\tau, n \models \ominus\psi$ iff $\tau, n-1 \models \psi$ iff, since $\psi \in \mathit{Subf}(\ominus\psi)$, $\tau', n'-1 \models \psi$ iff $\tau', n' \models \ominus\psi$;
- $\varphi = \psi_1 \mathcal{S} \psi_2$ and consider the one-step unfolding equivalence $\psi_1 \mathcal{S} \psi_2 \equiv \psi_2 \vee (\psi_1 \wedge \ominus\psi_1 \mathcal{S} \psi_2)$. Then, we have that $\tau, n \models \psi_1 \mathcal{S} \psi_2$ iff $\tau, n \models \psi_2 \vee (\psi_1 \wedge \ominus\psi_1 \mathcal{S} \psi_2)$ iff $\tau, n \models \psi_2$ or $\tau, n \models \psi_1 \wedge \ominus\psi_1 \mathcal{S} \psi_2$ and we distinguish two cases:
  - $\tau, n \models \psi_2$, then observe that $\mathit{Subf}(\psi_2) \subseteq \mathit{Subf}(\varphi)$ and so, by induction hypothesis, we obtain that $\tau', n' \models \psi_2$, which in turns implies $\tau', n' \models \psi_1 \mathcal{S} \psi_2$';
  - $\tau, n \models \psi_1 \wedge \ominus\psi_1 \mathcal{S} \psi_2$ iff $\tau, n \models \psi_1$ and $\tau, n \models \ominus\psi_1 \mathcal{S} \psi_2$. On the one hand, from $\tau, n \models \psi_1$ we easily obtain that $\tau', n' \models \psi_1$. On the other hand, from $\tau, n \models \ominus\psi_1 \mathcal{S} \psi_2$ it follows that $\tau, n-1 \models \psi_1 \mathcal{S} \psi_2$. Now, since $\psi_1 \mathcal{S} \psi_2 \in \mathit{Subf}(\varphi)$, it holds that $\tau', n'-1 \models \psi_1 \mathcal{S} \psi_2$ and so that $\tau', n' \models \ominus\psi_1 \mathcal{S} \psi_2$. We therefore obtain $\tau', n' \models \psi_1 \wedge \ominus\psi_1 \mathcal{S} \psi_2$, which implies $\tau', n' \models \psi_1 \mathcal{S} \psi_2$.

□

Appendix B: Proof of Theorem 1

Proof. First, observe that the actions available to the agent in M_{ag}^{ppam} after each $(s_0, \ell_0) \dots (s_n, \ell_n)$ (where ℓ_i are sets of fluents at timestep i) are the same as the actions available to the agent in the MDP $M_{ag \times ppam}$ in (s_n, v_n) where v_n is the set of $\psi \in \{\mathit{Subf}(\varphi_a) : a \in \mathit{Act}\}$ such that $\ell_0 \dots \ell_n \models \psi$. This is because the evaluation of formulas from $\{\mathit{Subf}(\varphi_a) : a \in \mathit{Act}\}$ including φ_a for each $a \in \mathit{Act}$ is the same in $(s_0, \ell_0) \dots (s_n, \ell_n)$ and in (s_n, v_n). Hence, instead of masking actions based on $(s_0, \ell_0) \dots (s_n, \ell_n)$, we can mask them depending on (s_n, v_n) without loss of information. Note that the rewards are the same in both MDPs. This means that the reward for performing a after $(s_0, \ell_0) \dots (s_n, \ell_n)$ is the same as the reward for performing a in (s, v_n). Since $Tr' : S \times V \times \mathit{Act} \rightharpoonup Pr(S \times V)$ corresponds to $Tr^{ag \times ppam} : S \times \mathcal{L} \times \mathit{Act} \rightharpoonup Pr(S \times \mathcal{L})$, the optimal non-Markovian policy $\rho^* : (S \times \mathcal{L})^+ \to Pr(\mathit{Act})$ produces exactly the same reward as a Markovian policy $\rho : S \times V \to Pr(\mathit{Act})$ obtained by replacing $\ell_0, \dots, \ell_n$ with the corresponding v_n. □

Appendix C: Proof of Lemma 2

In the following, we denote by $Q_{\mathcal{M}}$ and Q_s the set of states of, respectively, the MDP abstraction and safety automaton that were used to synthesise the input shield $\mathcal{S}$; L is the set of input observations for the shield, we denote with ℓ elements of L.

Proof. First, let $a \in \mathit{Act}$ be an arbitrary action. If an MDP abstract state-observation pair $(q_{\mathcal{M}}, \ell)$ is such that $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$ for any safety automaton state q_s (reachable from the initial state q_0) and for which ℓ is observable from $(q_{\mathcal{M}}, q_s)$, then $q_{\mathcal{M}}$ and ℓ being the current MDP abstraction state and observation is a sufficient precondition to allow the agent to take action a. Thus, we define Λ^a as the set of all such MDP abstraction state-observation pairs.

Now, let $q_{\mathcal{M}}$ be an MDP abstraction state and ℓ an observation such that there are two distinct safety automaton states q_s, q'_s for which: (i) both q_s and q'_s are reachable from q_0, (ii), ℓ can be observed in both $(q_{\mathcal{M}}, q_s)$ and $(q_{\mathcal{M}}, q'_s)$, and (iii), $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$ and $a \notin \lambda((q_{\mathcal{M}}, q'_s), \ell)$. For $q_{\mathcal{M}}$ and ℓ, we need to keep track of the current state q_s in the safety automaton, so that we allow a if and only if the current state q_s is such that $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$. Let $Q_s^{a, q_{\mathcal{M}}, \ell} = \{q_s \in Q_s : a \in \lambda((q_{\mathcal{M}}, q_s), \ell)\}$. Thanks to Lemma 1, we can build for each of these $q_s \in Q_s^{a, q_{\mathcal{M}}, \ell}$ a PPLTL formula ϕ_{q_s} that is satisfied if and only if, given the current history, the safety automaton $\mathcal{A}_s$ is in state q_s[5].

We are now ready to define the PPLTL formula φ_a that will constrain action a. In the formula we implicitly omit all the $(q_{\mathcal{M}}, \ell) \in (Q_{\mathcal{M}} \times L) \setminus \Lambda^a$ such that $Q_s^{a, q_{\mathcal{M}}, \ell} = \emptyset$.

$$\varphi_a = \bigvee_{(q_{\mathcal{M}}, \ell) \in \Lambda^a} (q_{\mathcal{M}} \wedge \ell) \vee \bigvee_{(q_{\mathcal{M}}, \ell) \in (Q_{\mathcal{M}} \times L) \setminus \Lambda^a} \left(q_{\mathcal{M}} \wedge \ell \wedge \left(\bigvee_{q_s \in Q_s^{a, q_{\mathcal{M}}, \ell}} \phi_{q_s} \right) \right)$$

We now show that action a is allowed by the shield $\mathcal{S}$ if and only if the current trace τ satisfies φ_a. Let $(q_{\mathcal{M}}, q_s)$ be the current state of the shield and ℓ the current observation, reached after following the trace τ. Observe that $a \in \lambda((q_{\mathcal{M}}, q_s), \ell)$ if and only if, for any state of the safety automaton q'_s such that $(q_{\mathcal{M}}, q'_s)$ is reachable from $q_{0,\mathcal{S}}$ and ℓ is observable at $(q_{\mathcal{M}}, q'_s)$, either $a \in \lambda((q_{\mathcal{M}}, q'_s), \ell)$ or not.

In the first case, it means that whenever the state of the MDP abstraction is $q_{\mathcal{M}}$ and the observation is ℓ, the shield will always allow action a. Notice that the fluent for state $q_{\mathcal{M}}$ and for observation ℓ are true in the last timestep of trace τ if and only if $q_{\mathcal{M}}$ is the current state of the MDP abstraction and ℓ the current observation. Hence, $\mathcal{S}$ allows action a if and only if $\tau \models \varphi_a$, since $\tau \models q_{\mathcal{M}} \wedge \ell$ for $(q_{\mathcal{M}}, \ell) \in \Lambda^a$.

In the second case, it means that the current safety automaton state q_s is such that, when coupled with the current MDP abstraction state $q_{\mathcal{M}}$ and observation ℓ, $a \in \lambda((q_{\mathcal{M}}, q_s))$. This is true if and only if the trace τ is such that: (i) $q_{\mathcal{M}}$ and ℓ are true at its final timestep, and (ii), $\tau \models \phi_{q_s}$, by Lemma 1. Hence, we have again that $\mathcal{S}$ allows action a if and only if $\tau \models \varphi_a$, since $\tau \models q_{\mathcal{M}} \wedge \ell \wedge \left(\bigvee_{q_s \in Q_s^{a, q_{\mathcal{M}}}} \phi_{q_s} \right)$ for

[5] Notice that the safety automaton transitions on inputs from $L \times \mathit{Act}$, hence why we also need Act to be included in the set of fluents available to define the PPLTL formula φ_a.

$(q_{\mathcal{M}}, \ell) \in (Q_{\mathcal{M}} \times L) \setminus \Lambda^a$ and $q_s \in Q_s^{a,q_{\mathcal{M}},\ell}$. □

Appendix D: Figures of the Environments

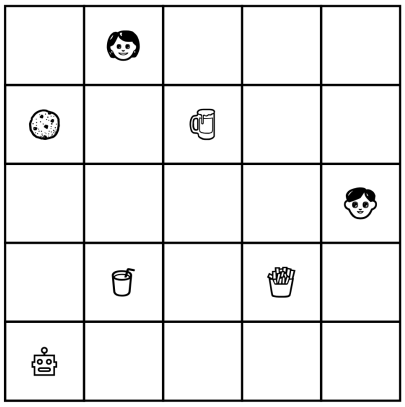

Figure 4: COCKTAILPARTY environment (De Giacomo et al. 2019).

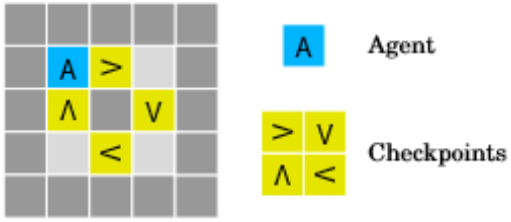

Figure 5: BOATRACE environments (Leike et al. 2017). Dark grey tiles represent walls.

Appendix E: Further Explanation of BOATRACE

The PPLTL formulas for the `move_west`, `move_north`, `move_south` and `move_east` actions are as follows:

$$\varphi_{\texttt{move\_west}} = (\neg(north \vee west))\mathcal{S}(south \vee east)$$

$$\varphi_{\texttt{move\_north}} = (\neg(north \vee east))\mathcal{S}(west \vee south)$$

$$\varphi_{\texttt{move\_south}} = (\neg(south \vee west))\mathcal{S}(north \vee east)$$

$$\varphi_{\texttt{move\_east}} = (\neg(south \vee east))\mathcal{S}(north \vee west)$$