Safe Reinforcement Learning Via Shielding

M. Alshiekh, R. Bloem, et al.

Akhoury Shauryam

Chennai Mathematical Institute

Table of contents

Prelimaries

Motivation

Reinforcement Learning (RL) algorithms are powerful tools for discovering policies that maximize rewards in various domains.

Reinforcement Learning (RL) algorithms are powerful tools for discovering policies that maximize rewards in various domains.

However, traditional RL approaches do not inherently prioritize safety during the learning or execution phases. This lack of safety assurance can be particularly concerning in critical applications where incorrect actions could lead to harmful or catastrophic outcomes.

Motivation

Reinforcement Learning (RL) algorithms are powerful tools for discovering policies that maximize rewards in various domains.

However, traditional RL approaches do not inherently prioritize safety during the learning or execution phases. This lack of safety assurance can be particularly concerning in critical applications where incorrect actions could lead to harmful or catastrophic outcomes.

Therefore, there is a growing need to develop methods that not only optimize for reward but also ensure adherence to safety specifications throughout the learning process.

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- S is a finite set of states

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- S is a finite set of states
- s_1 is a unique state, representing the initial state $s_1 \in S$

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- S is a finite set of states
- s_1 is a unique state, representing the initial state $s_1 \in S$
- $\mathcal{A}$ is a set of States $\{a_1, a_2, .., a_n\}$

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- S is a finite set of states
- s_1 is a unique state, representing the initial state $s_1 \in S$
- $\mathcal{A}$ is a set of States $\{a_1, a_2, .., a_n\}$
- $\mathcal{P} : S \times \mathcal{A} \rightarrow \textsc{Dist}(S)$ is the transition function.

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- S is a finite set of states
- s_1 is a unique state, representing the initial state $s_1 \in S$
- $\mathcal{A}$ is a set of States $\{a_1, a_2, .., a_n\}$
- $\mathcal{P} : S \times \mathcal{A} \to \textsc{Dist}(S)$ is the transition function.
- $\mathcal{R} : S \times \mathcal{A} \times S \to \mathbb{R}$ is the reward function.

A Markov Decision Process (MDP) is defined as a tuple,
$\mathcal{M} = (S, s_1, \mathcal{A}, \mathcal{P}, \mathcal{R})$

- S is a finite set of states
- s_1 is a unique state, representing the initial state $s_1 \in S$
- $\mathcal{A}$ is a set of States $\{a_1, a_2, .., a_n\}$
- $\mathcal{P} : S \times \mathcal{A} \rightarrow \textsc{Dist}(S)$ is the transition function.
- $\mathcal{R} : S \times \mathcal{A} \times S \rightarrow \mathbb{R}$ is the reward function.

The system works in a interaction between an agent and an environment.

RL: Continued

Beginning at s_t, the agent picks an action a_t at time t, it goes to a state s_{t+1} based on the distribution described by $\mathcal{P}$, and it gets a reward r_{t+1} based on $\mathcal{R}$

Beginning at s_t, the agent picks an action a_t at time t, it goes to a state s_{t+1} based on the distribution described by $\mathcal{P}$, and it gets a reward r_{t+1} based on $\mathcal{R}$

The goal of the agent is to maximize the Expected Average Rewards, and to make the influence for later actions less, we use a discount factor $\gamma \in [0, 1]$

RL: Continued

Beginning at s_t, the agent picks an action a_t at time t, it goes to a state s_{t+1} based on the distribution described by $\mathcal{P}$, and it gets a reward r_{t+1} based on $\mathcal{R}$

The goal of the agent is to maximize the Expected Average Rewards, and to make the influence for later actions less, we use a discount factor $\gamma \in [0, 1]$

The objective of the agent is to find a policy π, a function from S to $\mathcal{A}$ which maximizes the Expected Value of $\sum_{t=0}^{\text{inf}} \gamma^t \cdot r_t$

LTL

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

LTL

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

Some relevant Temporal Operators are:

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

Some relevant Temporal Operators are:

- X: Next

LTL

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

Some relevant Temporal Operators are:

- X: Next
- U: Until

LTL

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

Some relevant Temporal Operators are:

- X: Next
- U: Until
- G: Global

LTL

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

Some relevant Temporal Operators are:

- X: Next
- U: Until
- G: Global
- F: Finally

LTL

LTL stands for Linear Temporal Logic, it is a standard to encode formulae about a path's future in a transition system.

Some relevant Temporal Operators are:

- X: Next
- U: Until
- G: Global
- F: Finally
- R: Release

We can encode needed Safety properties using LTL.

Objective

We can encode needed Safety properties using LTL.

Our objective here is to introduce a novel approach to reinforcement learning that incorporates the enforcement of safety properties expressed in temporal logic.

Objective

We can encode needed Safety properties using LTL.

Our objective here is to introduce a novel approach to reinforcement learning that incorporates the enforcement of safety properties expressed in temporal logic.

Furthermore, we seek to establish the requirements that a procedure must meet to ensure that it does not compromise the convergence guarantees of the learning agent.

Objective

We can encode needed Safety properties using LTL.

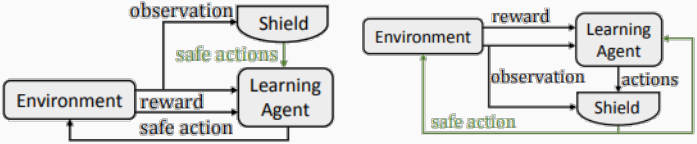
Our objective here is to introduce a novel approach to reinforcement learning that incorporates the enforcement of safety properties expressed in temporal logic.

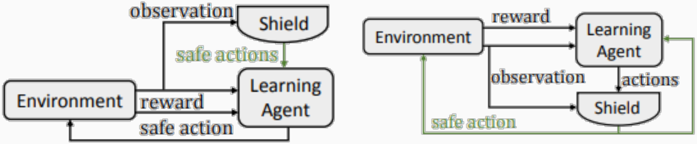
Furthermore, we seek to establish the requirements that a procedure must meet to ensure that it does not compromise the convergence guarantees of the learning agent.

Working

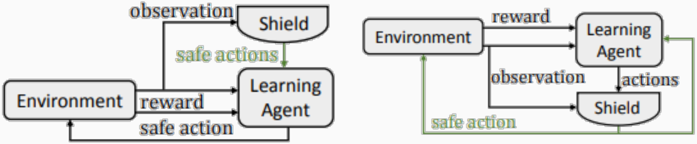
In Teacher-Guided Reinforcement Learning (RL), a human or external entity, known as the teacher, intervenes to provide guidance and advice to the learning agent, particularly in scenarios where safety is paramount. The teacher observes the agent's actions and can suggest safe actions or steer the agent away from potentially harmful decisions. This intervention aims to prevent undesirable outcomes and ensure the efficacy of the learning process.

One common application of Teacher-Guided RL is in Q-learning, where the teacher advises the agent on safe actions when necessary. Unlike traditional RL, where the agent learns solely from trial and error, Teacher-Guided RL incorporates external knowledge to enhance safety and guide the learning process toward more desirable outcomes.


observation
Shield
safe actions
Environment
reward
Learning Agent
safe action
Environment
reward
Learning Agent
observation
actions
Shield
safe action



The image above describes the framework for Shielding. The two types of Shielding are Preemptive and Post-Posed.



The image above describes the framework for Shielding. The two types of Shielding are Preemptive and Post-Posed.

Their separate working and properties are described later. The general framework tries to make the agent avoid actions that shield considers as unsafe, keeping an upper bound of changes in mind.

Working

Shielding in reinforcement learning involves the integration of a correct-by-construction reactive system, known as a shield, into the learning process. The shield continuously monitors the actions chosen by the learning agent and intervenes only when necessary to ensure safety.

Working

Shielding in reinforcement learning involves the integration of a correct-by-construction reactive system, known as a shield, into the learning process. The shield continuously monitors the actions chosen by the learning agent and intervenes only when necessary to ensure safety.

If the chosen action is deemed unsafe based on the system's specifications and environment dynamics, the shield overwrites it with a safe alternative. This proactive approach minimizes interference with the agent's learning while prioritizing the preservation of safe system behavior.

The interaction between the agent, the environment, and the shield can be described as follows: At each time step t, the shield computes a set of all safe actions $\{a_1^t, \ldots, a_k^t\}$.

Preemptive

The interaction between the agent, the environment, and the shield can be described as follows: At each time step t, the shield computes a set of all safe actions $\{a_1^t, \ldots, a_k^t\}$.

This set is derived by considering all available actions and filtering out those that would violate the safety specification ϕ_s. The agent then receives this filtered list from the shield and selects an action $a_t \in \{a_1^t, \ldots, a_k^t\}$.

Preemptive

The interaction between the agent, the environment, and the shield can be described as follows: At each time step t, the shield computes a set of all safe actions $\{a_1^t, \ldots, a_k^t\}$.

This set is derived by considering all available actions and filtering out those that would violate the safety specification ϕ_s. The agent then receives this filtered list from the shield and selects an action $a_t \in \{a_1^t, \ldots, a_k^t\}$.

Subsequently, the environment executes the chosen action a_t, transitions to the next state s_{t+1}, and provides the reward r_{t+1}. Essentially, the role of the shield is to dynamically adjust the available actions for the agent at each time step to ensure that only safe actions are considered.

The Preemtpive shielding approach effectively transforms the original MDP M into a modified version $M_0 = (S_0, s_1, A_0, P_0, R_0)$, where unsafe actions at each state are removed.

Preemptive: Properties

The Preemtpive shielding approach effectively transforms the original MDP M into a modified version $M_0 = (S_0, s_1, A_0, P_0, R_0)$, where unsafe actions at each state are removed.

Here, S_0 is the product of the original MDP and the state space of the shield.

Preemptive: Properties

The Preemtpive shielding approach effectively transforms the original MDP M into a modified version $M_0 = (S_0, s_1, A_0, P_0, R_0)$, where unsafe actions at each state are removed.

Here, S_0 is the product of the original MDP and the state space of the shield.

For every state $s \in S_0$, a new subset of available actions $\mathcal{A}_0 \subseteq \mathcal{A}$ is created by applying the shield to $\mathcal{A}$, eliminating all unsafe actions. Transition function P_0 from each state $s \in S_0$ contains only transition distributions from P for actions within $\mathcal{A}_0$.

The Post-Posed shield monitors the agent's actions and replaces them with safe alternatives when necessary to avoid violating ϕ_s. At each step t, the agent selects an action a_1^t. If a_1^t is unsafe with respect to ϕ_s, the shield substitutes it with a safe action a_1^t instead.

The Post-Posed shield monitors the agent's actions and replaces them with safe alternatives when necessary to avoid violating ϕ_s. At each step t, the agent selects an action a_1^t. If a_1^t is unsafe with respect to ϕ_s, the shield substitutes it with a safe action a_1^t instead.

The environment then executes a_1^t, transitions to s_{t+1}, and provides r_{t+1}. The agent receives a_1^t and r_{t+1}, updating its policy accordingly. There are two approaches to handling the reward for a_1^t when it's replaced

The two approaches to handling the reward are:

The two approaches to handling the reward are:

- Assign a punishment r_0^{t+1} to a_1^t. The agent learns that selecting a_1^t at state s_t is unsafe without violating ϕ_s. However, this doesn't guarantee exclusion of unsafe actions from the final policy, requiring the shield to remain active post-learning.

Post-Posed Reward Updates

The two approaches to handling the reward are:

- Assign a punishment r_0^{t+1} to a_1^t. The agent learns that selecting a_1^t at state s_t is unsafe without violating ϕ_s. However, this doesn't guarantee exclusion of unsafe actions from the final policy, requiring the shield to remain active post-learning.
- Assign the reward r_{t+1} to a_1^t. The agent updates the unsafe action with the reward, potentially including unsafe actions in an optimal policy. Since unsafe actions are always replaced with safe ones, this doesn't pose an issue, but the shield remains necessary during learning and execution.

Post-Posed: Properties

One significant advantage of post-posed shielding is its ability to operate seamlessly even when the learning algorithm has transitioned into the execution phase, thereby adhering to a fixed policy.

One significant advantage of post-posed shielding is its ability to operate seamlessly even when the learning algorithm has transitioned into the execution phase, thereby adhering to a fixed policy.

At each step, the learning algorithm solely perceives the MDP state (without knowledge of the shield's state), while the shield intervenes to rectify the learner's actions as needed, ensuring the system's safe operation.

Post-Posed: Properties

One significant advantage of post-posed shielding is its ability to operate seamlessly even when the learning algorithm has transitioned into the execution phase, thereby adhering to a fixed policy.

At each step, the learning algorithm solely perceives the MDP state (without knowledge of the shield's state), while the shield intervenes to rectify the learner's actions as needed, ensuring the system's safe operation.

Remarkably, the learning agent doesn't even require awareness of being shielded.

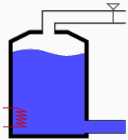
Specifying for RL as LTL

Water Storage

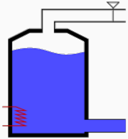
We're designing an energy-efficient controller for a hot water tank. A heater keeps the water warm, its energy use depends on the tank's level, which we're unsure of.

Water Storage

We're designing an energy-efficient controller for a hot water tank. A heater keeps the water warm, its energy use depends on the tank's level, which we're unsure of.



We're designing an energy-efficient controller for a hot water tank. A heater keeps the water warm, its energy use depends on the tank's level, which we're unsure of.



Water flows out at 0-1 liters/sec, and in at 1-2 liters/sec when the valve's open (closed otherwise). The tank holds 100 liters max, and valve changes need a 3-second minimum to preserve it. Overflow and running dry must be prevented.

Suppose we use *level* as a variable for the amount of waters currently in the tanker and *open* and *close* as its states. Then for our given scenario, we can encode this in LTL using:

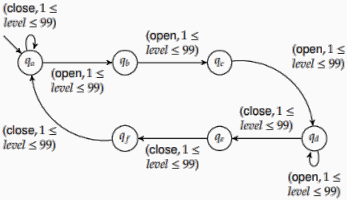
Specifications for LTL

Suppose we use *level* as a variable for the amount of waters currently in the tanker and *open* and *close* as its states. Then for our given scenario, we can encode this in LTL using:

$G(\text{level} > 0) \wedge G(\text{level} < 100)$
$\wedge G((\text{open} \wedge X\text{close}) \rightarrow XX\text{close} \wedge XXX\text{close})$
$\wedge G((\text{close} \wedge X\text{open}) \rightarrow XX\text{open} \wedge XXX\text{open})$

Suppose we use *level* as a variable for the amount of waters currently in the tanker and *open* and *close* as its states. Then for our given scenario, we can encode this in LTL using:

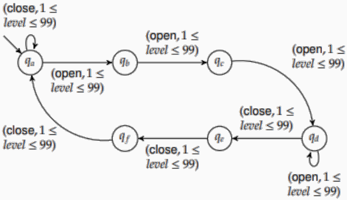
$$G(\text{level} > 0) \wedge G(\text{level} < 100)$$
$$\wedge G((\text{open} \wedge X\text{close}) \rightarrow XX\text{close} \wedge XXX\text{close})$$
$$\wedge G((\text{close} \wedge X\text{open}) \rightarrow XX\text{open} \wedge XXX\text{open})$$



Specifications for LTL

Suppose we use *level* as a variable for the amount of waters currently in the tanker and *open* and *close* as its states. Then for our given scenario, we can encode this in LTL using:

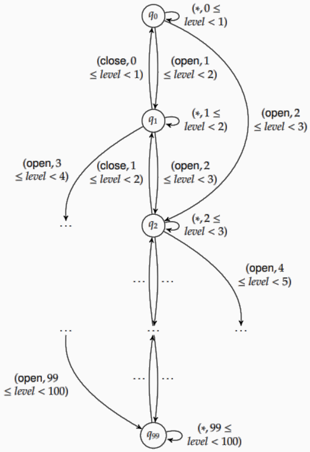
$$G(\text{level} > 0) \wedge G(\text{level} < 100)$$
$$\wedge G((\text{open} \wedge X\text{close}) \rightarrow XX\text{close} \wedge XXX\text{close})$$
$$\wedge G((\text{close} \wedge X\text{open}) \rightarrow XX\text{open} \wedge XXX\text{open})$$

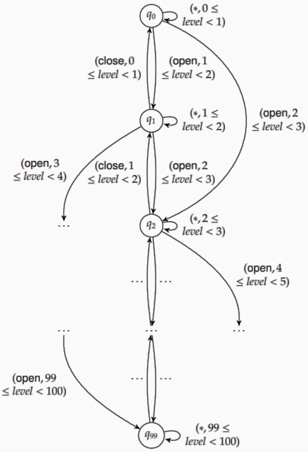


The above Temporal Path shows the available transitions in our model.

Synthesis

Further Specifications

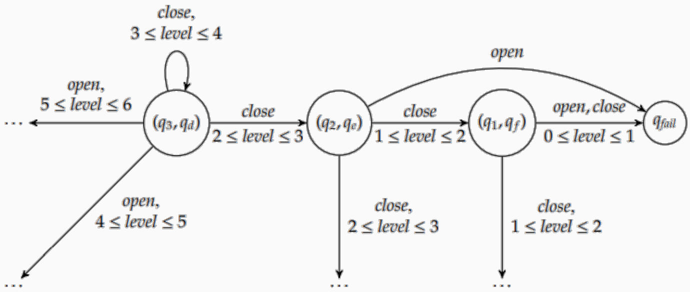




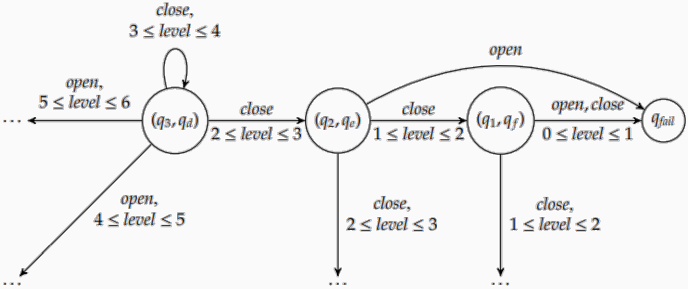
Like how the previous abstraction, only cared about the specification about the Water Heater being in the same state for at-least 3 seconds, this takes care of the level of water not overflowing or drying out.

Since the previous two specifications do not use a fail state, we take the product of the pairwise elements from both the graphs, describe the transition and add a fail state.

Since the previous two specifications do not use a fail state, we take the product of the pairwise elements from both the graphs, describe the transition and add a fail state.



Since the previous two specifications do not use a fail state, we take the product of the pairwise elements from both the graphs, describe the transition and add a fail state.



Using this, we can treat this as a Safety Game and check whether our current action leads to an unsafe state.

Minimum Interference

The agent at each step gives out a ranking of actions it wants to take at time t. $a_1 > a_2 > a_3 ...$

Minimum Interference

The agent at each step gives out a ranking of actions it wants to take at time t. $a_1 > a_2 > a_3 \ldots$

Picking some $|rank|$ as a Minimum Interference Parameter, the Shield chooses an action from the first $|rank|$ ranked actions unless none of them are safe.

The agent at each step gives out a ranking of actions it wants to take at time t. $a_1 > a_2 > a_3 \ldots$

Picking some $|rank|$ as a Minimum Interference Parameter, the Shield chooses an action from the first $|rank|$ ranked actions unless none of them are safe.

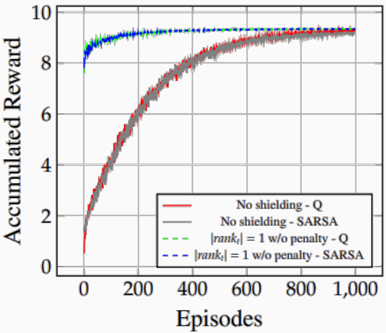
If all the actions aren't unsafe, this provides a correct-by-construction method of keeping RL safe during learning and deployment.

Results

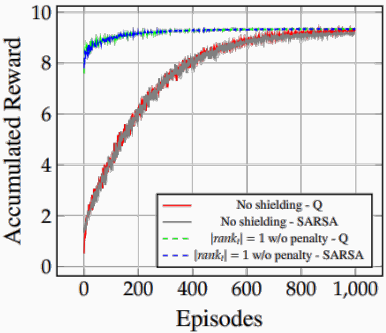
Water Tank

Post-Posed Shield synthesis from the two propositions for the water tank experiment took under a second.

Post-Posed Shield synthesis from the two propositions for the water tank experiment took under a second.

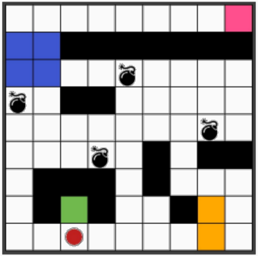


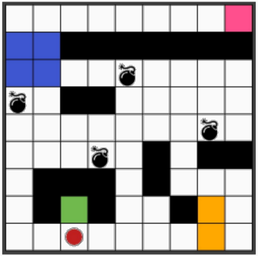
Post-Posed Shield synthesis from the two propositions for the water tank experiment took under a second.



The above graph shows cumulative reward based on number of episodes. It shows how shielding fastens up the process while remaining safe.

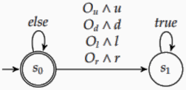
Grid Game

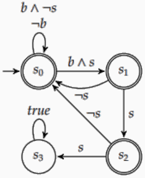


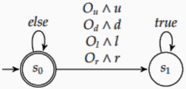


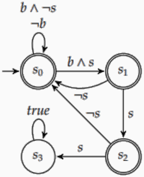
In the above game, the red-dot player has to avoid the walls and cant stay on the bomb for more than 2 consecutive steps while covering the colored squares in a particular order.

Grid Game: Continued

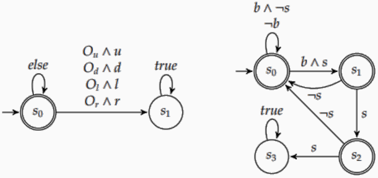




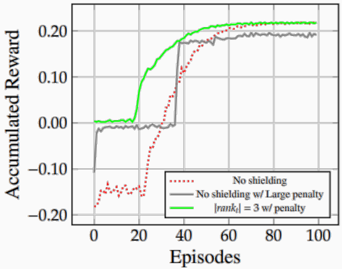




The above describes the 2 specifications we have in a DFA format, using this to synthesise shields:



The above describes the 2 specifications we have in a DFA format, using this to synthesise shields:



Conclusions

Summary

In conclusion, this method introduces a novel approach to reinforcement learning by integrating safety constraints expressed as temporal logic specifications. By shielding the learning algorithm from violating these specifications, we ensure safer decision-making while minimizing the need for intricate understanding of the underlying learning process.

Summary

In conclusion, this method introduces a novel approach to reinforcement learning by integrating safety constraints expressed as temporal logic specifications. By shielding the learning algorithm from violating these specifications, we ensure safer decision-making while minimizing the need for intricate understanding of the underlying learning process.

Our experiments across various reinforcement learning scenarios demonstrate the effectiveness of shielded agents, consistently performing as well as, if not better than, unshielded counterparts. While constructing an abstraction for safety enforcement requires effort, our results underscore its significance in ensuring not only safe learning but also potentially enhancing learning performance.

Future Work

In a Future Work, author R. Bloem presents Probabilistic Shielding, it:

Future Work

In a Future Work, author R. Bloem presents Probabilistic Shielding, it:

- Explores the integration of a probabilistic shield. This would enable decision-making to adhere to safety constraints with high probability, enhancing the robustness of our learning system.

Future Work

In a Future Work, author R. Bloem presents Probabilistic Shielding, it:

- Explores the integration of a probabilistic shield. This would enable decision-making to adhere to safety constraints with high probability, enhancing the robustness of our learning system.
- Investigating this further could result in a boost up in the number of episodes required to reach convergence.

Thank You!