Explainable Multi-Agent Reinforcement Learning for Temporal Queries

Kayla Boggess[1], **Sarit Kraus**[2] and **Lu Feng**[1]
[1]University of Virginia
[2]Bar-Ilan University
{kjb5we, lu.feng}@virginia.edu, sarit@cs.biu.ac.il

Abstract

As multi-agent reinforcement learning (MARL) systems are increasingly deployed throughout society, it is imperative yet challenging for users to understand the emergent behaviors of MARL agents in complex environments. This work presents an approach for generating policy-level contrastive explanations for MARL to answer a temporal user query, which specifies a sequence of tasks completed by agents with possible cooperation. The proposed approach encodes the temporal query as a PCTL* logic formula and checks if the query is feasible under a given MARL policy via probabilistic model checking. Such explanations can help reconcile discrepancies between the actual and anticipated multi-agent behaviors. The proposed approach also generates correct and complete explanations to pinpoint reasons that make a user query infeasible. We have successfully applied the proposed approach to four benchmark MARL domains (up to 9 agents in one domain). Moreover, the results of a user study show that the generated explanations significantly improve user performance and satisfaction.

1 Introduction

As multi-agent reinforcement learning (MARL) systems are increasingly deployed throughout society, it is imperative yet challenging for users to understand the emergent behaviors of MARL agents in complex environments. Recent work [Boggess *et al.*, 2022] proposes methods for generating policy summarization to explain agents' behaviors under a given MARL policy, as well as language explanations to answer user queries about agents' decisions such as "Why don't [*agents*] do [*actions*] in [*states*]?" However, existing methods cannot handle temporal queries involving a sequence of MARL agents' decisions, for example, "Why don't [*agents*] complete [*task 1*], followed by [*task 2*], and eventually [*task 3*]?" Explanations to answer such a temporal user query can help reconcile discrepancies between the actual and anticipated agent behaviors.

Recently, there has been increasing interest in generating policy-level contrastive explanations for RL in the single-agent setting. [Sreedharan *et al.*, 2022] considers a problem setting where the agent comes up with a plan to achieve a certain goal, and the user responds by raising a foil (represented as a sequence of agent states and actions). To show why the agent's plan is preferred over the foil (e.g., the foil leads to an invalid state), explanations are generated by finding missing preconditions of the failing foil action on a symbolic model through sample-based trials. [Finkelstein *et al.*, 2022] considers a similar problem setting, where the user queries about an alternative policy specifying actions that the agent should take in certain states. Explanations are defined as a sequence of Markov decision process (MDP) transforms, such that the RL agent's optimal policy (i.e., seeking to maximize its accumulated reward) in the transformed environment aligns with the user queried policy.

There are many challenges and limitations when applying these approaches in multi-agent environments. First, we need a better representation of user queries. Asking the user to provide concrete information about agents' joint states and joint actions, which grow exponentially with the increasing number of agents, is tedious, if not impractical. Further, these approaches have limited scalability in multi-agent environments due to computational complexity. [Sreedharan *et al.*, 2022] requires a large number of samples generated via a random walk to find missing preconditions. [Finkelstein *et al.*, 2022] computes a sequence of MDP transforms (e.g., mapping the entire state/action space) and retrains the agent policy in each transformed MDP. Moreover, the generated explanations may not capture agent cooperation requirements that are essential for understanding multi-agent behaviors.

We address these challenges by developing an approach to generate policy-level contrastive explanations for MARL. Our proposed approach takes the input of a temporal user query specifying which tasks should be completed by which agents in what order. Any unspecified tasks are allowed to be completed by the agents at any point in time. The user query is then encoded as a PCTL* logic formula, which is checked against a multi-agent Markov decision process (MMDP) representing an abstraction of a given MARL policy via *probabilistic model checking* [Kwiatkowska *et al.*, 2017]. If the MMDP satisfies the PCTL* formula, then the user query is feasible under the given policy (i.e., there exists at least one policy execution that conforms with the user query). Otherwise, our approach deploys a guided rollout procedure to

sample more of the MARL agents' behaviors and update the MMDP with new samples. If the updated MMDP still does not satisfy the PCTL* formula, the proposed approach generates correct and complete explanations that pinpoint the causes of all failures in the user query.

Computational experiments on four benchmark MARL domains demonstrate the scalability of our approach (up to 9 agents in one domain). It only took seconds to check the feasibility of a user query and generate explanations when needed. Additionally, we conducted a user study to evaluate the quality of generated explanations, where we adapted [Sreedharan *et al.*, 2022] to generate baseline explanations. The study results show that, compared with the baseline, explanations generated using our approach significantly improve user performance (measured by the number of correctly answered questions) and yield higher average user ratings on *explanation goodness metrics* (e.g., understanding, satisfaction) [Hoffman *et al.*, 2018].

2 Related Work

2.1 Explainable Reinforcement Learning

A growing body of research in explainable RL has emerged in recent years, as surveyed in [Wells and Bednarz, 2021; Heuillet *et al.*, 2021; Puiutta and Veith, 2020]. Existing works can be categorized according to different axes (e.g., timing, scope, form, setting). We position our proposed approach based on these categorizations as follows.

First, there are *intrinsic* and *post-hoc* methods depending on the timing when the explanation is generated. The former (e.g., [Topin *et al.*, 2021; Landajuela *et al.*, 2021]) builds intrinsically interpretable policies (e.g., represented as decision trees) at the time of training, while the latter (e.g., [Sreedharan *et al.*, 2022; Hayes and Shah, 2017]) generates post-hoc explanations after a policy has been trained. Our proposed approach belongs to the latter.

Second, existing works can be distinguished by the scope of explanations. Some methods provide explanations about policy-level behaviors (e.g., [Topin and Veloso, 2019; Amir and Amir, 2018]), while others explain specific, local decisions (e.g., [Olson *et al.*, 2021; Madumal *et al.*, 2020]). Our work focuses on explaining discrepancies between actual and anticipated policy-level behaviors.

Additionally, current approaches generate explanations in diverse forms, including natural language [Hayes and Shah, 2017], saliency maps [Atrey *et al.*, 2019], reward decomposition [Juozapaitis *et al.*, 2019], finite-state machines [Danesh *et al.*, 2021], and others. Our proposed approach generates language explanations following [Hayes and Shah, 2017] and [Boggess *et al.*, 2022], both of which use the Quine-McCluskey algorithm to compute a minimized Boolean formula and then translate the formula into an explanation using language templates.

Finally, the majority of existing works on explainable RL focus on the single-agent setting. There is very little prior work considering multi-agent environments. [Heuillet *et al.*, 2022] estimates the contribution of each agent for a group plan, but only as a general explanation of a model and not for a specific instance given by a user. [Boggess *et al.*, 2022] develops methods to generate policy summarization and query-based language explanations for MARL. However, as discussed in Section 1, existing methods cannot handle temporal queries considered in this work.

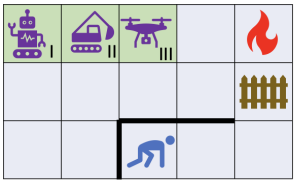
2.2 Contrastive Explanations

[Miller, 2019] identifies being *contrastive* ("Why A but not B?") as one of the key desired properties of an explanation. The research thread on contrastive explanations for RL has been drawing increasing attention since then. For example, [Madumal *et al.*, 2020] generates contrastive explanations for *"why action"* and *"why not action"* queries via counterfactual analysis of a structural causal model; [Lin *et al.*, 2021] develops a deep RL architecture with an embedded self-prediction model to explain why a learned agent prefers one action over another; and [Olson *et al.*, 2021] computes counterfactual state explanations (i.e., minimal changes needed for an alternative action). These works all focus on generating contrastive explanations for the RL agent's local decisions in a state. By contrast, several recent works [Sreedharan *et al.*, 2022; Finkelstein *et al.*, 2022] generate policy-level contrastive explanations in the single-agent setting. However, as discussed in Section 1, these methods are not suitable for MARL. Our proposed approach advances the state of the art by developing an approach for generating contrastive explanations about MARL agents' policy-level behaviors.

3 Program Formulation

We consider a problem setting where a MARL policy has been trained over N agents, denoted by $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, which is a function mapping a set of joint states $\mathcal{X} = \{\mathbf{x} = (x^1, \ldots, x^N)\}$ to a distribution over a set of joint actions $\mathcal{A} = \{\mathbf{a} = (a^1, \ldots, a^N)\}$. Execution of policy π yields a sequence of joint states and joint actions $\mathbf{x}_0 \xrightarrow{\mathbf{a}_0} \mathbf{x}_1 \xrightarrow{\mathbf{a}_1} \cdots$ where $\mathbf{a}_t \sim \pi(\cdot|\mathbf{x}_t)$ at each step t. Suppose that the goal of the agents is to jointly complete a set G of tasks (sub-goals). Let $R^i : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ denote the reward function that determines the immediate reward received by agent i. A positive reward $R^i(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1}) > 0$ is only received when a task $g \in G$ is completed by agent i at step t. We assume that each agent can complete at most one task at a step and, if multiple agents cooperate to complete a task, each of them would receive a positive reward at the same step.

To start with, the user is presented with a high-level plan that summarizes one possible execution of the given MARL policy π. For example, consider a MARL domain where three robotic agents are trained to complete search and rescue tasks shown in Figure 1(a). We can compute a high-level plan by applying the policy summarization method proposed in [Boggess *et al.*, 2022]. Figure 1(b) illustrates an example plan, where columns indicate the order of tasks completed by agents and each row corresponds to an agent's task sequence. Agent cooperation is represented by multiple agents sharing the same task in the same column. In this example, robots II and III first cooperate to fight the fire, followed by robots I and II jointly removing the obstacle, and finally robots I and III rescue the victim together.



(a)

	Task 1	Task 2	Task 3
Robot I	-	Obstacle	Victim
Robot II	Fire	Obstacle	-
Robot III	Fire	-	Victim

(b)

Figure 1: Example MARL domain and a high-level plan.

The user may not desire the presented plan and raise an alternative query. The user query does not have to be a complete plan involving all agents and tasks. Instead, the user can query about a partial plan such as *"Why don't robots I and II remove the obstacle before robot II fights the fire alone?"* We define a *temporal user query* as a list of atomic propositions specifying an order of tasks completed by some agents, denoted by $\rho = \langle \tau_1, \tau_2, \cdots \rangle$, where each τ specifies a task $g \in G$ and designated agents. Tasks not specified in the query can be completed in any order (e.g., before τ_1, between τ_1 and τ_2, or after τ_2). The aforementioned example query is denoted by $\langle \mathsf{obstacle\_robotI\_robotII}, \mathsf{fire\_robotII} \rangle$.

A temporal user query ρ is *feasible* under a MARL policy π if there exists at least one execution of π that conforms with the queried plan ρ. When ρ is infeasible under π, explanations are generated to reconcile discrepancies between the actual and anticipated multi-agent behaviors. We say that an explanation is *correct* if it pinpoints the causes of one or more failures in ρ (e.g., unsatisfied task preconditions or agent cooperation requirements). A correct explanation is *complete* if it identifies the reasons behind all failures of a user query ρ.

This work aims to solve the following problem.

Problem: Given a temporal user query ρ and a trained MARL policy π, check if ρ is feasible under policy π. If ρ is infeasible, generate correct and complete explanations to reconcile discrepancies between the actual and anticipated multi-agent behaviors.

4 Approach

To tackle this problem, we present an approach as illustrated in Algorithm 1. We describe the construction of a policy abstraction (line 1) in Section 4.1, the encoding and checking of the user query (lines 2-5) in Section 4.2, guided rollout (lines 6-9) in Section 4.3, and explanation generation (lines 10-11) in Section 4.4. Additionally, we analyze the correctness and complexity of the approach in Section 4.5.

Algorithm 1 Checking the feasibility of a user query

Input: a temporal user query ρ, a trained MARL policy π
Output: YES, or explanations $\mathcal{E}$

1: construct a policy abstraction MMDP $\mathcal{M}$ given π
2: encode the temporal query ρ as a PCTL* formula φ
3: **if** $\mathcal{M}$ satisfies φ **then**
4: **return** YES
5: **else**
6: $\mathcal{M}' \leftarrow$ update $\mathcal{M}$ via guided rollout (Algorithm 2)
7: **if** $\mathcal{M}'$ satisfies φ **then**
8: **return** YES
9: **else**
10: generate explanations $\mathcal{E}$ (Algorithm 3)
11: **return** $\mathcal{E}$

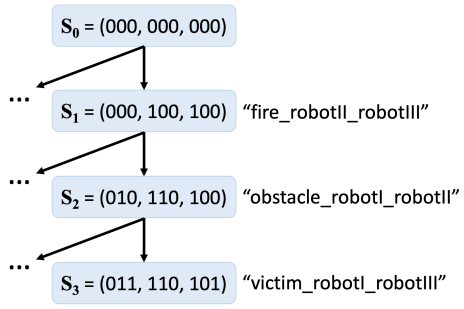


Figure 2: Fragment of an example MMDP.

4.1 Policy Abstraction MMDP

Given a trained MARL policy π, we construct a multi-agent Markov decision process (MMDP) abstraction following the policy abstraction method described in [Boggess *et al.*, 2022]. We denote an MMDP as a tuple $\mathcal{M} = (\mathcal{S}, \mathbf{s}_0, \mathcal{A}, \mathcal{T}, \mathcal{L})$ with a set of joint abstract states $\mathcal{S}$, an initial state $\mathbf{s}_0 \in \mathcal{S}$, a set of joint actions $\mathcal{A}$, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and a labeling function $\mathcal{L} : \mathcal{S} \to 2^{AP}$ that assigns a set of atomic propositions AP to states. A path through $\mathcal{M}$ is a sequence $\mathbf{s}_0 \xrightarrow{\mathbf{a}_0} \mathbf{s}_1 \xrightarrow{\mathbf{a}_1} \cdots$ starting from the initial state $\mathbf{s}_0$.

The state space $\mathcal{S} = \{\mathbf{s} = (s^1, \ldots, s^N)\}$ is defined over a set of Boolean predicates indicating whether a task $g \in G$ has been completed by agent i. The initial state $\mathbf{s}_0$ represents that none of the tasks has been completed. In the example MMDP shown in Figure 2, the initial state is $\mathbf{s}_0 = (000, 000, 000)$. State $\mathbf{s}_1 = (000, 100, 100)$ represents that the fire task has been completed by robotic agents II and III, which is labeled with $\mathcal{L}(\mathbf{s}_1) = \{\mathsf{fire\_robotII\_robotIII}\}$. The next state $\mathbf{s}_2 = (010, 110, 100)$ is labeled with $\mathcal{L}(\mathbf{s}_2) = \{\mathsf{obstacle\_robotI\_robotII}\}$, which only contains the newly completed obstacle task.

The MMDP transition function $\mathcal{T}$ is built by finding corresponding abstract transitions $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ of each sample $(\mathbf{x}, \mathbf{a}, \mathbf{x}')$ observed during the MARL policy evaluation, and transition probabilities are computed via frequency counting. Given a joint state $\mathbf{x} = (x^1, \ldots, x^N)$, we determine a corresponding joint abstract state $\mathbf{s} = (s^1, \ldots, s^N)$ by checking if agent i receives a reward $R^i(\mathbf{x}, \mathbf{a}, \mathbf{x}') > 0$ for completing a task $g \in G$. For each MMDP state $\mathbf{s} \in \mathcal{S}$, we keep track of a set of corresponding sampled joint states, denoted by $X(\mathbf{s}) = \{\mathbf{x}\}$, and count the total number of observed MARL samples, denoted by $C(\mathbf{s})$.

4.2 Query Checking with Temporal Logic

We encode a temporal user query $\rho = \langle \tau_1, \tau_2, \cdots \rangle$ as a PCTL* logic [Aziz *et al.*, 1995] formula φ with a "sequencing" specification template as follows.

$$\varphi = \mathsf{P}_{>0}[\Diamond(\tau_1 \wedge \Diamond(\tau_2 \wedge \Diamond \cdots))]$$

Algorithm 2 Guided rollout

Input: a trained MARL policy π, a policy abstraction MMDP $\mathcal{M}$
Output: an updated MMDP $\mathcal{M}'$

1: unfold $\mathcal{M}$ as a search tree and assign a U value to each node
2: $\mathcal{N} \leftarrow$ tree nodes ordered by U values and sample counts
3: **for** $(k = 0; k <$ `RolloutNum`$; k{+}{+})$ **do**
4: $\quad \mathbf{s} \leftarrow \mathcal{N}$.pop(0)
5: $\quad \mathbf{x} \leftarrow$ pick a corresponding joint state from $X(\mathbf{s})$
6: $\quad \delta \leftarrow$ a rollout execution of π from $\mathbf{x}$ with `DepthLimit`
7: $\quad$ update the MMDP with samples in δ
8: **return** the updated MMDP $\mathcal{M}'$

where $\mathsf{P}_{>0}$ means that the specification should be satisfied with non-zero probability, and $\Diamond$ denotes the logical operator "eventually". The PCTL* formula φ is satisfied in an MMDP $\mathcal{M}$ if there exists a path through $\mathcal{M}$ such that τ_1 eventually becomes true at some point along the path, and τ_2 eventually holds at some point afterward. For example, the MMDP shown in Figure 2 satisfies a PCTL* formula $\mathsf{P}_{>0}[\Diamond(\mathsf{fire\_robotII\_robotIII} \wedge \Diamond\mathsf{victim\_robotI\_robotIII})]$.

To check if $\mathcal{M}$ satisfies φ, we apply *probabilistic model checking* [Kwiatkowska *et al.*, 2017] which offers efficient techniques for the exhaustive exploration of $\mathcal{M}$ to determine if φ holds in any path. If $\mathcal{M}$ satisfies φ, then Algorithm 1 returns YES, indicating that the user query is feasible under the given MARL policy. Otherwise, there does not exist any path through $\mathcal{M}$ that conforms with the user query. Since the MMDP $\mathcal{M}$ is constructed based on samples observed during the MARL policy evaluation, it does not necessarily capture all possible agent behaviors under the given policy π. Thus, $\mathcal{M}$ not satisfying φ is not a sufficient condition for claiming that the user query is infeasible under the given MARL policy. To address this issue, we develop a guided rollout procedure to update the MMDP $\mathcal{M}$ via drawing more samples from the MARL policy π.

4.3 Guided Rollout

Algorithm 2 illustrates the guided rollout procedure, which starts by unfolding paths of the MMDP $\mathcal{M}$ as a search tree. The root node of the tree is the initial state $\mathbf{s}_0$ of $\mathcal{M}$. As the search tree unfolds, we assign a U value to each node representing the degree to which the path from the root node to the current node conforms with the user query. Consider an example user query $\langle \tau_1 = \mathsf{fire\_robotII\_robotIII}, \tau_2 = \mathsf{obstacle\_robotII} \rangle$, unfolding the MMDP in Figure 2 yields $U(\mathbf{s}_0) = 0, U(\mathbf{s}_1) = 1$ for conforming with τ_1, and $U(\mathbf{s}_2) = -\infty$ for violating τ_2. The search tree stops expanding a node with $U = -\infty$ since the user query is already violated along the path.

Let $\mathcal{N}$ be a queue of tree nodes ordered by decreasing U values and, for nodes with the same U value, increasing counts of MARL samples $C(\mathbf{s})$. This ordering prioritizes the exploration of states with a higher degree of user query conformance (i.e., U values) and less sampling. Given a joint abstract state $\mathbf{s} \in \mathcal{N}$, we (randomly) pick a corresponding joint state $\mathbf{x} \in X(\mathbf{s})$ and generate a rollout execution $\delta = \mathbf{x} \xrightarrow{\mathbf{a}} \mathbf{x}' \xrightarrow{\mathbf{a}'} \cdots$ of the policy π starting from $\mathbf{x}$. The rollout depth $|\delta|$ is bounded by a predefined parameter `DepthLimit`. We update the MMDP with samples observed in δ. Then, we consider the next node in $\mathcal{N}$ and repeat the above process (lines 4-7 of Algorithm 2). When the number of rollout executions hits a predefined parameter `RolloutNum`, Algorithm 2 terminates with an updated MMDP, denoted by $\mathcal{M}'$.

Algorithm 3 Generating reconciliation explanations

Input: a user query $\rho = \langle \tau_1, \tau_2, \cdots \rangle$, the updated MMDP $\mathcal{M}'$
Output: explanations $\mathcal{E}$

1: $\mathcal{E} \leftarrow \{\}$
2: **while** ρ is infeasible in $\mathcal{M}'$ **do**
3: $\quad U^{\mathsf{max}} \leftarrow$ the maximum U value in the search tree of $\mathcal{M}'$
4: $\quad$ find a failure τ_j where $j = U^{\mathsf{max}} + 1$
5: $\quad \mathcal{V} \leftarrow$ target MMDP states that complete the task in τ_j
6: $\quad \bar{\mathcal{V}} \leftarrow$ non-target MMDP states
7: $\quad$ **if** $\mathcal{V} \neq \emptyset$ **then**
8: $\quad\quad \phi \leftarrow$ Quine-McCluskey(1=`binary`$(\mathcal{V})$, 0=`binary`$(\bar{\mathcal{V}})$)
9: $\quad\quad \epsilon \leftarrow$ select a minterm in ϕ that is closest to ρ
10: $\quad \mathcal{E} \leftarrow$ insert language explanations
11: $\quad$ update ρ to fix the failure τ_j
12: **return** $\mathcal{E}$

We check if $\mathcal{M}'$ satisfies a PCTL* formula φ encoding the user query ρ (line 7 of Algorithm 1) as described in Section 4.2. If $\mathcal{M}'$ satisfies φ, then the user query ρ is feasible under the given MARL policy π. When $\mathcal{M}'$ does not satisfy φ, the user query is infeasible in the MMDP $\mathcal{M}'$. Given sufficiently large `RolloutNum` and `DepthLimit`, the MMDP $\mathcal{M}'$ provides a good approximation of MARL agents' behaviors under the given policy π. Thus, we can claim that the user query ρ is infeasible under π with high probability. In this case, we generate explanations to reconcile discrepancies between the actual and anticipated multi-agent behaviors.

4.4 Explanation Generation

Algorithm 3 shows the explanation generation procedure. Given the updated MMDP $\mathcal{M}'$ resulting from Algorithm 2, we unfold $\mathcal{M}'$ as a search tree and assign a U value to each tree node following Section 4.3. Let U^{max} denote the maximum U value in the tree. Then, τ_j with $j = U^{\mathsf{max}} + 1$ is a failed task making the query ρ infeasible. For example, consider a user query $\langle \tau_1 = \mathsf{obstacle\_robotI\_robotII}, \tau_2 = \mathsf{victim\_robotI}, \tau_3 = \mathsf{fire\_robotII\_robotIII} \rangle$, which yields $U^{\mathsf{max}} = 0$ indicating that τ_1 fails. To pinpoint the cause of this failure, we find a set of target MMDP states $\mathcal{V}$ where the failed task is completed by some agents (not necessarily by the queried agents). All other possible states (including those not sampled) are placed in a non-target set $\overline{\mathcal{V}}$.

When $\mathcal{V}$ is non-empty, we obtain a minimized Boolean formula ϕ by applying the Quine-McCluskey algorithm [Quine, 1952], which represents the minimal description of the states in the target set $\mathcal{V}$ compared to those in the non-target set $\overline{\mathcal{V}}$. We select a minterm ϵ in ϕ that is closest to ρ (e.g., involving queried agents) and convert ϵ into an explanation using language templates. For example, the MMDP state $\mathbf{s}_2$ in Figure 2 is a target state for τ_1 based on its state label, which indicates that the obstacle task is completed by robots I and II in this state. Applying Quine-McCluskey yields a single-minterm formula $\phi = \mathsf{fire\_robotII} \wedge \mathsf{fire\_robotIII} \wedge \mathsf{obstacle\_robotI} \wedge$

obstacle_robotII. Recall our assumption in Section 3 that each agent can complete at most one task at a step. Thus, the fire task must have been completed by robots II and III in some previous state. We obtain an explanation: *"The robots cannot remove the obstacle because fighting the fire must be completed before removing the obstacle."*

To generate correct and complete explanations for all possible failures in a user query, we update ρ based on the minterm ϵ to fix the failure τ_j. Since ϵ is the closest minterm to ρ, the applied changes are minimal. We check whether the updated ρ is feasible in $\mathcal{M}'$ via probabilistic model checking as described in Section 4.2. If the model checker yields YES, then Algorithm 3 terminates because all failures of the (original) user query have been explained and fixed. Otherwise, the algorithm repeats lines 3-11 for the updated ρ. Following the previous example, we update the query as $\langle \tau_1 = \mathsf{fire\_robotII\_robotIII}, \tau_2 = \mathsf{obstacle\_robotI\_robotII}, \tau_3 = \mathsf{victim\_robotI} \rangle$, which results in $U^{\mathsf{max}} = 2$, indicating that the updated query still has a failure $\tau_3 = \mathsf{victim\_robotI}$. The MMDP state $\mathbf{s}_3$ in Figure 2 is a target state where the victim task is completed. Applying Quine-McCluskey yields $\phi = \mathsf{victim\_robotI} \wedge \mathsf{victim\_robotIII}$, which only contains one minterm and is translated into an explanation: *"The robots cannot rescue the victim because Robot I needs Robot III to help rescue the victim."* We further update the query as $\langle \tau_1 = \mathsf{fire\_robotII\_robotIII}, \tau_2 = \mathsf{obstacle\_robotI\_robotII}, \tau_3 = \mathsf{victim\_robotI\_robotIII} \rangle$, which is feasible because the MMDP path $\mathbf{s}_0 \rightarrow \mathbf{s}_1 \rightarrow \mathbf{s}_2 \rightarrow \mathbf{s}_3$ in Figure 2 conforms with this query. The algorithm terminates and returns the generated explanations of all failures.

Note that in the special case where the target states set $\mathcal{V}$ is empty, we skip the Quine-McCluskey and generate an explanation to indicate that the queried task has not been completed in any observed sample. Then, we update the user query by removing the failed task and continue with Algorithm 3.

4.5 Correctness and Complexity

Correctness. The correctness of our proposed approach, with respect to the problem formulated in Section 3, is stated below and the proof is given in the appendix.

Proposition 1. *Given a temporal user query ρ and a trained MARL policy π, if Algorithm 1 returns YES, then the query ρ must be feasible under the policy π; otherwise, Algorithm 1 generates correct and complete explanations $\mathcal{E}$.*

Complexity. We analyze the complexity of the following key steps in the proposed approach.

- The time complexity of checking an MMDP against a PCTL* formula φ defined in Section 4.2 via probabilistic model checking is double exponential in $|\varphi|$ (i.e., equal to the length of the user query $|\rho|$) and polynomial in the size of the MMDP [Baier and Katoen, 2008]. The MMDP state space size $|\mathcal{S}|$ is bounded by $\mathcal{O}(2^{|G|^N})$, depending on the number of agents N and tasks $|G|$. However, only a small set of reachable states is usually induced in practice (as shown in Table 1), given a well-trained MARL policy.
- The time complexity of guided rollout (Algorithm 2) is given by $\mathcal{O}\big(\texttt{RolloutNum} \cdot \texttt{DepthLimit}\big)$. As discussed above, the larger the parameter values of `RolloutNum` and `DepthLimit`, the better approximation of MARL policy behaviors captured by the updated MMDP $\mathcal{M}'$.
- The time complexity of explanation generation (Algorithm 3) is given by $\mathcal{O}\big(\lambda \cdot (3^{N \cdot |G|}/\sqrt{N \cdot |G|})\big)$, where λ is the number of failures in the user query, and $\mathcal{O}\big(3^{N \cdot |G|}/\sqrt{N \cdot |G|}\big)$ is the time complexity of Quine-McClusky [Chandra and Markowsky, 1978].

Even though the complexity is high, in practice it is possible to check query feasibility and generate explanations in reasonable times as shown in the next section.

5 Computational Experiments

To demonstrate the scalability of our approach, we developed a prototype implementation and applied it to four benchmark MARL domains [1].

(1) *Search and Rescue (SR)*, where multiple robotic agents cooperate to complete tasks such as fighting fires and rescuing victims [Boggess *et al.*, 2022].

(2) *Level-Based Foraging (LBF)*, where agents play a mixed cooperative-competitive game to collect food scattered in a gridworld [Papoudakis *et al.*, 2021].

(3) *Multi-Robot Warehouse (RWARE)*, where robots collaboratively move and deliver requested goods [Papoudakis *et al.*, 2021].

(4) *PressurePlate (PLATE)*, where agents are required to cooperate during the traversal of a gridworld, with some agents staying on pressure plates to open the doorway for others to proceed [McInroe and Christianos, 2022].

Our prototype implementation used the Shared Experience Actor-Critic [Christianos *et al.*, 2020] for MARL policy training and evaluation. All models were trained and evaluated until converging to the expected reward, or up to 10,000 steps, whichever occurred first. The PRISM probabilistic model checker [Kwiatkowska *et al.*, 2011] was applied for checking the feasibility of user queries. We set the guided rollout parameters as $\texttt{RolloutNum} = 10$ and $\texttt{DepthLimit} = 50$. The experiments were run on a machine with 2.1 GHz Intel CPU, 132 GB of memory, and CentOS 7 operating system.

Table 1 shows experimental results. For each case study, we report the number of agents N, the number of tasks $|G|$, and the length of user queries $|\rho|$. Additionally, we report the size of policy abstraction MMDPs $\mathcal{M}$ in terms of the number of (reachable) states $|\mathcal{S}|$ and the number of transitions $|\mathcal{T}|$. In general, the MMDP size increases with a growing number of agents and tasks. However, an unequal distribution of agent actions under the MARL policy π can lead to a smaller MMDP $\mathcal{M}$ (e.g., LBF-5) as agents take the same trajectories more often leading to less exploration.

We consider two temporal queries (i.e., a feasible query and an infeasible query with the same length $|\rho|$) in each case study and report the runtime of Algorithm 1. For infeasible queries, we also report the number of failures λ, which were controlled to grow with the environment size as the longer the

[1] Code available at github.com/kjboggess/ijcai23

Case Study				MMDP $\mathcal{M}$		Feasible	Infeasible	
Domain	N	$\|G\|$	$\|\rho\|$	$\|\mathcal{S}\|$	$\|\mathcal{T}\|$	Time (s)	λ	Time (s)
SR	3	3	3	28	127	0.8	1	2.2
	4	4	4	163	674	1.5	2	5.3
	5	5	5	445	1,504	24.4	3	89.8
LBF	3	3	3	67	344	0.9	1	2.9
	4	4	4	211	781	2.1	2	7.6
	5	5	5	152	454	4.5	3	20.5
RWARE	2	4	3	98	268	0.8	1	15.5
	3	6	5	442	1,260	3.7	2	42.2
	4	8	8	1,089	2,751	21.7	3	85.2
PLATE	5	3	3	87	181	0.8	1	3.0
	7	4	4	85	175	0.9	2	25.7
	9	5	5	132	266	1.4	3	126.8

Table 1: Experimental results on four benchmark MARL domains.

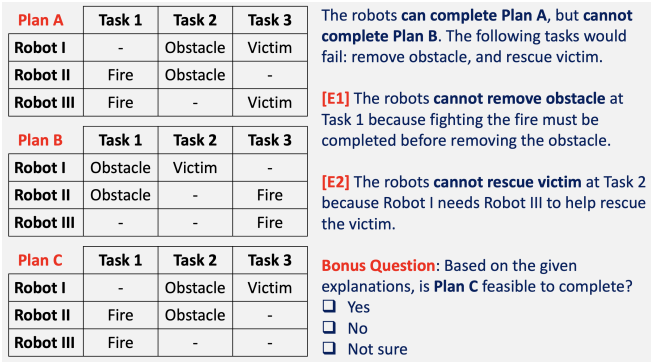


Figure 3: Example of the user study interface displaying explanations generated by the proposed approach.

query length $|\rho|$, the larger number of task failures it may contain. The size of generated explanations is equal to the number of failures (i.e., one for each task failure in the user query). Experimental results show that all queries were solved efficiently within seconds. Checking an infeasible query is generally slower than checking a feasible query in the same case study, due to the extra time needed for guided rollout and generating explanations.

In summary, computational experiments demonstrate that our approach can be successfully applied to various benchmark MARL domains with a large number of agents (e.g., up to 9 agents in the PLATE domain), for checking the feasibility of temporal user queries and generating explanations when needed.

6 User Study

We evaluate the quality of generated reconciliation explanations via a user study. [2] Section 6.1 describes the study design and Section 6.2 analyzes the results.

6.1 Study Design

User interface. The study was conducted via the Qualtrics survey platform. Instead of allowing participants to raise queries in real-time, we generated explanations for a selected set of temporal queries *a priori*, which enables us to present the same set of explanations to different participants. Figure 3 shows an example of the user interface. Participants were shown the agents' original plan (Plan A) and an alternate plan representing a temporal query (Plan B). An explanation was presented to explain why Plan B was infeasible. Participants were then asked to use the provided explanation to decide if a new query (Plan C) was feasible. Participants were incentivized with bonus payments to answer the question correctly.

Participants. We recruited 88 participants (i.e., fluent English speakers over the age of 18) through university mailing lists (52% male, 45.5% female, 2.3% non-binary). They had an average age of 23.9 (SD = 6.1). To ensure data quality, a demonstration was given, attention checks were injected, and the time to complete the survey was tracked.

[2]This study was approved by University of Virginia Institutional Review Boards IRB-SBS #5226.

Baseline. We adapted the explanation generation method in [Sreedharan *et al.*, 2022], which was initially proposed for the single-agent setting, as a baseline for comparison. We extended the method for joint states and actions and limited its sampling to the given policy instead of the larger environment. Furthermore, we use the same user interface as shown in Figure 3 to avoid any confounding variables regarding presentation in the study. The baseline method takes the input of a user query expressed as a sequence of agent states and actions, for which we converted a high-level plan (e.g., Plan B in Figure 3) into a low-level execution of joint states and joint actions. Explanations generated using the baseline method could fail to capture agent cooperation requirements in multi-agent environments. Moreover, the baseline method only provides explanations for the first point of failure rather than all failures in a user query. For example, the baseline explanations for Plan B in Figure 3 changes the second sentence in the explanation to "The first failed task would be: remove obstacle." and only contains E1. Participants would not be able to answer the bonus question correctly without knowing E2.

Independent variables. We employed a within-subject study design where participants were asked to complete two trials for evaluating explanations generated using the baseline method and our proposed approach, respectively. There were 4 sets of temporal queries (i.e., two single-failure queries and two with multiple failures) and bonus questions in each trial. The queried plans and questions used in the two trials were different but had a similar difficulty level. Participants were presented with the same set of plans and questions and were randomly assigned to two groups (i.e., evaluating the baseline explanations before or after the proposed explanations) to counterbalance the ordering confound effect.

Dependent measures. We counted the number of questions correctly answered by participants as a performance measure. Additionally, at the end of each trial, participants were instructed to rate on a 5-point Likert scale (1 - strongly disagree, 5 - strongly agree) the following statements regarding *explanations good metrics* adapted from [Hoffman *et al.*, 2018].

- The explanations help me *understand* how the robots complete the mission.
- The explanations are *satisfying*.

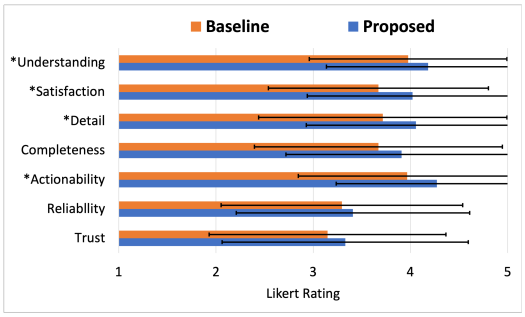


Figure 4: Mean and SD of participant ratings on explanation goodness metrics ("*" indicates statistically significant difference with the significant level set as $\alpha = 0.05$).

- The explanations are sufficiently *detailed*.
- The explanations are sufficiently *complete*, that is, they provide me with all the needed information to answer the questions.
- The explanations are *actionable*, that is, they help me know how to answer the questions.
- The explanations let me know how *reliable* the robots are for completing the mission.
- The explanations let me know how *trustworthy* the robots are for completing the mission.

Hypotheses. We tested two hypotheses stated below.

- **H1:** Explanations generated by our proposed approach *enable participants to answer more questions correctly* than the baseline explanations.
- **H2:** Explanations generated by our proposed approach *lead to higher ratings on explanation goodness metrics* than the baseline explanations.

6.2 Results Analysis

Question-answering performance. Participants were able to answer more questions correctly based on explanations generated by our proposed approach (M=3.1 out of 4, SD=1.0) than those generated with the baseline method (M=0.6 out of 4, SD=0.8). A paired t-test ($\alpha = 0.05$) shows a statistically significant difference ($t(87)$=-17.0, $p \leq 0.01$, d=1.8). *Thus, the data supports H1.*

Recall that the baseline method only provides explanations for the first point of failure in a user query and could not always correctly identify agent cooperation requirements. By contrast, our approach generates correct and complete explanations for all failures in a user query, which help participants to better understand agent behaviors under a given policy, and thus, leads to better question-answering performance.

Explanation goodness ratings. Figure 4 shows that participants gave higher subjective ratings to the proposed explanations than the baseline explanations on average, with respect to *all* explanation goodness metrics.

We used the Wilcoxon signed-rank test ($\alpha = 0.05$) to evaluate hypothesis H2. Statistically significant differences were found for the following four metrics: *understanding* (W=315.0, Z=-1.6, $p \leq 0.05$, r=-0.1), *satisfaction* (W=236.0, Z=-2.2, $p \leq 0.01$, r=-0.2), *detail* (W=255.0, Z=-1.6, $p \leq 0.01$, r=-0.1), and *actionability* (W=105.5, Z=-2.0, $p \leq 0.02$, r=-0.1). But no significant difference was found on other metrics: *completeness* (W=389.5, Z=-1.2, $p \leq 0.1$, r=-0.1), *reliability* (W=255.5, Z=-0.5, $p \leq 0.4$, r=-0.04), and *trust* (W=181.5, Z=-1.0, $p \leq 0.07$, r=-0.1). *Thus, the data partially supports H2.*

Participants' superior question-answering performance is consistent with their statistically significant higher subjective ratings on understanding, detail, and actionability (i.e., the proposed explanations provide detailed and actionable information for answering questions). Furthermore, the baseline explanations were rated significantly less satisfying, because they may miss essential information (e.g., agent cooperation) for answering questions. Participants may misjudge the explanations' completeness as they were unaware of the total number of failures in a queried plan. Finally, the generated explanations are mostly about missing task preconditions, which are less useful for participants to judge how reliable and trustworthy the robots are for completing the mission.

Summary. Results of the user study show that, compared with the baseline, explanations generated by our proposed approach significantly improve participants' performance in correctly answering questions, and lead to higher average ratings on explanation goodness metrics such as understanding and satisfaction.

7 Conclusion

This work presents an approach for generating policy-level contrastive explanations for MARL to answer a temporal user query, which specifies a sequence of tasks to be completed by agents with possible cooperation. The proposed approach checks if the user query is feasible under the given MARL policy and, if not, generates correct and complete explanations to pinpoint reasons that make a user query infeasible. A prototype implementation of the proposed approach was successfully applied to four benchmark MARL domains with a large number of agents (e.g., up to 9 agents in one domain). In all the experiments, it only took seconds to check the feasibility of a user query and generate explanations when needed. Additionally, a user study was conducted to evaluate the quality of generated explanations. The study results show that explanations generated using the proposed approach can help improve user performance, understanding, and satisfaction.

There are several directions to explore for possible future work. First, we will evaluate the proposed approach with different MARL methods. While the prototype implementation only uses one MARL algorithm, the proposed approach should be compatible with any MARL method because it only relies on sampling possible MARL executions. Second, we will leverage the expressiveness of PCTL* logic and investigate a richer set of user queries. For example, a "coverage" query which specifies a set of tasks to be covered in any order, and a "sequencing with avoidance" query which asks for the completion of a sequence of tasks while avoiding some other tasks to be completed by specific agents. Lastly, we would like to apply the proposed approach to a wide range of MARL environments in real-world scenarios.

Acknowledgments

This work was supported in part by U.S. National Science Foundation under grant CCF-1942836, U.S. Office of Naval Research under grant N00014-18-1-2829, U.S. Air Force Office of Scientific Research under grant FA9550-21-1-0164, Israel Science Foundation under grant 1958/20, and the EU Project TAILOR under grant 952215. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the grant sponsors.

A Appendix

Proposition 1. *Given a temporal user query ρ and a trained MARL policy π, if Algorithm 1 returns YES, then the query ρ must be feasible under the policy π; otherwise, Algorithm 1 generates correct and complete explanations $\mathcal{E}$.*

Proof. We prove the following two cases.
Case 1: When Algorithm 1 returns YES, the policy abstraction MMDP $\mathcal{M}$ or the updated MMDP $\mathcal{M}'$ satisfies the PCTL* formula φ encoding the user query ρ, indicating that there must exist a path through $\mathcal{M}$ or $\mathcal{M}'$ that conforms with ρ. By construction, every abstract MMDP transition $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ in $\mathcal{M}$ or $\mathcal{M}'$ with non-zero probability maps to at least one sampled decision $(\mathbf{x}, \mathbf{a}, \mathbf{x}')$ of the given MARL policy π. Thus, there must exist an execution of policy π that conforms with the user query ρ. By definition, the user query ρ is feasible under the given MARL policy π.
Case 2: Algorithm 1 returns explanations $\mathcal{E}$ generated via Algorithm 3. As described in Section 4.4, Algorithm 3 terminates when all failures in the user query ρ have been explained and fixed. Given a finite-length temporal query ρ, there is a finite number of failures. For any failure in the query, if the target states set $\mathcal{V}$ is non-empty, then the failure must be fixable using a Quine-McCluskey minterm that represents a target state where the failed task is completed. If $\mathcal{V}$ is empty, then the failure is removed from the query. Thus, the termination of Algorithm 3 is guaranteed. By definition, the generated explanations are *correct* (i.e., identifying the causes of one or more failures in ρ) and *complete* (i.e., finding the reasons behind all failures in ρ). □

References

[Amir and Amir, 2018] Dan Amir and Ofra Amir. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1168–1176, 2018.

[Atrey *et al.*, 2019] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[Aziz *et al.*, 1995] Adnan Aziz, Vigyan Singhal, Felice Balarin, Robert K Brayton, and Alberto L Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *International Conference on Computer Aided Verification*, pages 155–165. Springer, 1995.

[Baier and Katoen, 2008] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[Boggess *et al.*, 2022] K Boggess, S Kraus, and L Feng. Toward policy explanations for multi-agent reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[Chandra and Markowsky, 1978] Ashok K Chandra and George Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24(1):7–11, 1978.

[Christianos *et al.*, 2020] Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Thirty-fourth Conference on Neural Information Processing Systems*, pages 10707–10717. Curran Associates Inc, 2020.

[Danesh *et al.*, 2021] Mohamad H Danesh, Anurag Koul, Alan Fern, and Saeed Khorram. Re-understanding finite-state representations of recurrent policy networks. In *International Conference on Machine Learning*, pages 2388–2397. PMLR, 2021.

[Finkelstein *et al.*, 2022] Mira Finkelstein, Lucy Liu, Yoav Kolumbus, David C Parkes, Jeffrey Rosenschein, Sarah Keren, et al. Explainable reinforcement learning via model transforms. In *Advances in Neural Information Processing Systems*, 2022.

[Hayes and Shah, 2017] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–312. IEEE, 2017.

[Heuillet *et al.*, 2021] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685, 2021.

[Heuillet *et al.*, 2022] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Collective explainable ai: Explaining cooperative strategies and agent contribution in multiagent reinforcement learning with shapley values. *IEEE Computational Intelligence Magazine*, 17(1):59–71, 2022.

[Hoffman *et al.*, 2018] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

[Juozapaitis *et al.*, 2019] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on explainable artificial intelligence*, 2019.

[Kwiatkowska *et al.*, 2011] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[Kwiatkowska *et al.*, 2017] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic model checking: Advances and applications. In *Formal System Verification*, pages 73–121. Springer, 2017.

[Landajuela *et al.*, 2021] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR, 2021.

[Lin *et al.*, 2021] Zhengxian Lin, Kin-Ho Lam, and Alan Fern. Contrastive explanations for reinforcement learning via embedded self predictions. In *International Conference on Learning Representations*, 2021.

[Madumal *et al.*, 2020] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2493–2500, 2020.

[McInroe and Christianos, 2022] Trevor McInroe and Filippos Christianos. Repo for the multi-agent pressureplate environment. https://github.com/uoe-agents/pressureplate, 2022.

[Miller, 2019] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[Olson *et al.*, 2021] Matthew L Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence*, 295:103455, 2021.

[Papoudakis *et al.*, 2021] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[Puiutta and Veith, 2020] Erika Puiutta and Eric Veith. Explainable reinforcement learning: A survey. In *International cross-domain conference for machine learning and knowledge extraction*, pages 77–95. Springer, 2020.

[Quine, 1952] Willard V Quine. The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531, 1952.

[Sreedharan *et al.*, 2022] Sarath Sreedharan, Utkarsh Soni, Mudit Verma, Siddharth Srivastava, and Subbarao Kambhampati. Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations. In *International Conference on Learning Representations*, 2022.

[Topin and Veloso, 2019] Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2514–2521, 2019.

[Topin *et al.*, 2021] Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9923–9931, 2021.

[Wells and Bednarz, 2021] Lindsay Wells and Tomasz Bednarz. Explainable ai and reinforcement learning—a systematic review of current approaches and trends. *Frontiers in artificial intelligence*, 4:48, 2021.