A

Presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

by

APPROVAL SHEET

This

is submitted in partial fulfillment of the requirements
for the degree of

Author:

Advisor:

Advisor:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

Abstract

There has been a significant increase in multi-agent reinforcement learning (MARL) systems over the last several years for exciting applications. Yet, when functioning as black boxes, these systems can cause user misunderstanding and misuse since users do not always know when or why agents perform certain actions. Generating explanations regarding agent decisions is crucial as it improves system transparency, increases user satisfaction, and facilitates human-agent collaboration. Yet, the current work on explainable reinforcement learning (xRL) focuses mostly on the single-agent setting. So, there is a general need to generate methods for policy summarization to explain agents' global behaviors under a given MARL policy, as well as language explanations to answer user queries about agents' local decisions.

This dissertation will focus on generating summarizations and explanations for MARL. It explores two main questions. First, *how do we generate summarizations and explanations for centralized MARL?* We will generate explanations for centralized multi-agent reinforcement learning, treating all agents' actions as a single joint action. We will create global summaries and query-based explanations to address questions like when, why, and what actions are taken in specific states or conditions. Additionally, temporal explanations will clarify the feasibility of plans over time. Finally, we will assess the effectiveness of all methods through computational evaluations and user studies.

Secondly, *how do we generate summarizations and explanations for decentralized MARL?* We will produce explainable methods for decentralized reinforcement learning, where agents are given their own individual policies. We will provide both global policy summaries and query-based insights. Additionally, we will assess the effectiveness of all methods through computational evaluations and user studies.

First, chapter 2 discusses the existing work surrounding explainable MARL. Then, chapters 3 and 4 present the main contributions of the work regarding centralized MARL. Chapter 3 provides information on the developed summarization and explanation methods. While chapter 4 presents work to generate contrastive temporal explanations. The contributions of the work regarding decentralized MARL are presented in chapters 5 and 6. These chapters show the methods developed for summarizations and explanations, respectively. Finally, chapter 7 summarizes those contributions and broader impacts.

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Lu Feng, for her support and guidance throughout my graduate career. I am grateful for all the knowledge and encouragement that she has provided over the years. Without her, I would not have had the opportunity to pursue this path. Furthermore, I would like to thank Dr. Sarit Kraus for her mentorship in both research and life. I hope that one day I have the privilege to be as great of a researcher and person as she is.

I would also like to thank my committee members, Dr. Tariq Iqbal and Dr. Nicola Bezzo, for their help and support throughout this process. I would particularly like to thank Dr. Seongkook Heo for serving as my committee chair, collaborator, and teacher over the years.

Many thanks to the members, faculty, and staff of the University of Virginia Link Lab for allowing me to pursue my doctorate. They have provided support and kindness every step of the way and I am proud to say that I got to be a part of it. Additionally, thank you to Dr. Jessen Havill for inspiring me to pursue computer science in the first place.

Thank you to my collaborators, Shenghui “Vivian" Chen, Yunhao Yang, Dr. Ufuk Topcu, Dr. Bryan Choi, Robert Kim, and Dr. David Parker. I appreciate all the knowledge that you provided over the years and the various projects that we worked on. Also, thank you to my lab mates, Meiyi Ma, Josephine Lamp, Ingy ElSayed-Aly, Shili Sheng, MJ Kwon, Shuyang Dong, and Khang Huynh.

Finally, thank you to my friends and family for supporting me through this process. Thank you to my extended family for supporting me from near and far. Thank you to Matt Clark, Steve Lamp, and Elizabeth Palmieri for the many nights spent on research, projects, and fun. Thank you to Kaylee White and Mehar Sethi for at least trying to understand computer science. Thank you to my partner Edwin St. Hilaire for being my best friend. I wouldn’t have been able to do this without you. Thank you to my sister, Kayce Boggess, for being beside me in all aspects of life. Thank you to my parents, Kevin Boggess and Kimberly Ewing, for being my role models and showing me what I could achieve. Ultimately, thank you to the best little dogs, Kody, Katybird, Kash, and Keifer, for being the greatest support anyone could ask for.

To my mother.
Thank you for taking every single one of those surveys.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Imagine a search and rescue mission where multiple cooperative robots are executing a multi-agent reinforcement learning policy. A human operator in the field receives decision-making support via an explainer providing policy summaries and query-based explanations using real-time data from these robots. The summarizations help the human operator understand the robots' general behaviors (task completion, agent cooperation, task order), while the explanations provide specific answers in response to the operator's queries (when?, why not?, what?). Using this information, the human operator can make informed decisions, such as assisting with urgent tasks or requesting additional resources, improving overall mission performance.

There has been a significant increase in multi-agent reinforcement learning (MARL) systems over the last several years for exciting applications such as cooperative AI [1] and autonomous driving [2]. Yet, these systems can cause user misunderstanding and misuse when functioning as black boxes since users do not always know when or why agents perform certain actions. Generating explanations regarding agent decisions is crucial as it improves system transparency, increases user satisfaction, and facilitates human-agent collaboration [3], [4]. Yet, the current work on explainable reinforcement learning (xRL) focuses mostly on the single-agent setting, making the situation described above currently impossible [5]–[7].

So, there is a general need to generate methods for policy summarization to explain agents' global behaviors under a given MARL policy, as well as language explanations to answer user queries about agents' local decisions. The generated policy summarizations can help users to have a general view of agent decisions and support human-agent collaboration (e.g., users may adjust their workflow based on agents' task sequence). Furthermore, answered queries can be basic such as "Why don't [*agents*] do [*actions*] in [*states*]?" to provide answers to specific agent decisions, debug faulty agent behavior, and refine user mental models. However, existing methods obviously cannot handle even more advanced 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*]?" These more advanced explanations

help to reconcile discrepancies between anticipated agent behaviors and actual behaviors improving user understanding of possible agent actions [8].

1.2 Challenges

MARL algorithms can be categorized based on their training and execution frameworks. *Centralized training and execution* (CTCE) algorithms, leverage centrally shared information and train a single central policy over joint actions and observations of all agents [9]. *Centralized training with decentralized execution* (CTDE) algorithms, such as Shared Experience Actor-Critic (SEAC) [10], leverage shared information during training to update policies, but uses each agent's local observations for decentralized execution across independent agent policies. *Decentralized training and execution* (DTDE) algorithms, typified by independent learning [11], operate with agents treating each other as a part of the environmental dynamics and training individual policies in a completely local way. So, there are two general types of multi-agent reinforcement learning: centralized (execution) and decentralized (execution). Each algorithm type provides its own issues and benefits regarding agent performance, but also poses unique challenges when generating explanations.

1.2.1 Challenges of Centralized MARL

Centralized multi-agent reinforcement learning, or centralized MARL is characterized by its joint agent states and actions. Thus, when generating summarizations and explanations the combinatorial nature of MARL (i.e., the joint state/action space grows exponentially with the number of agents) leads to scalability issues. Furthermore, explanations should provide adequate information about agent behavior, including the interaction (e.g., cooperation) among multiple agents, for user understanding. Yet, explanations should avoid redundant information that may overwhelm or confuse users, thus decreasing user satisfaction and trust. Finally, we need a better representation of user queries, as 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 [12], [13].

1.2.2 Challenges of Decentralized MARL

Decentralized multi-agent reinforcement learning, or decentralized MARL, enables agents to make sequential decisions collaboratively by using individual components of a larger joint policy. So, each agent has its own individual policy with individual states and actions. Summarizations and explanations for decentralized MARL policies suffer from the same challenges as centralized MARL such as scalability.

However, users may struggle with understanding the given policies due to the large number of disconnected agents and the complexity of the cooperative tasks. Furthermore, decentralized MARL also suffers from partial visibility, resulting in agents being aware of their direct surroundings, but not the actions of other distant agents. Thus, no one agent can provide a full picture of the environment or joint policy. This leads to an inherent uncertainty in the order of agent actions, which user must be made aware of. So, specific interest must be paid to how agent information is processed and aggregated as an ineffective method may not produce a total picture of the policy or may produce behaviors that can not exist. Additionally, this can affect the presentation of explanations since the information from disparate agents regarding uncertain actions can overwhelm users.

1.3 Dissertation Overview and Contributions

This dissertation will focus on generating summarizations and explanations for multi-agent reinforcement learning. It explores two main questions. First, *how do we generate summarizations and explanations for centralized MARL?* We will generate explainable methods for centralized reinforcement learning, treating all agents' actions as a single joint action. We will create global summaries and query-based explanations to address questions like when, why, and what actions are taken in specific states or conditions. Additionally, temporal explanations will clarify the feasibility of plans over time. Finally, we will assess the effectiveness of all methods through computational evaluations and user studies.

Secondly, *how do we generate summarizations and explanations for decentralized MARL?* We will produce explainable methods for decentralized reinforcement learning, where agents are given their own independent policies. We will provide both global policy summaries and query-based insights. Additionally, we will assess the effectiveness of all methods through computational evaluations and user studies.

1.4 Dissertation Structure

First, chapter 2 discusses the existing work surrounding explainable MARL. Then, chapters 3 and 4 present the main contributions of the work regarding centralized MARL. Chapter 3 provides information on the developed summarization and explanation methods. While, chapter 4 presents work to generate contrastive temporal explanations. The contributions of the work regarding decentralized MARL are presented in chapters 5 and 6. These chapters show the methods developed for summarizations and explanations, respectively. Finally, chapter 7 summarizes those contributions and broader impacts. Figure 1.1 shows the overall structure of the contributions of this dissertation.

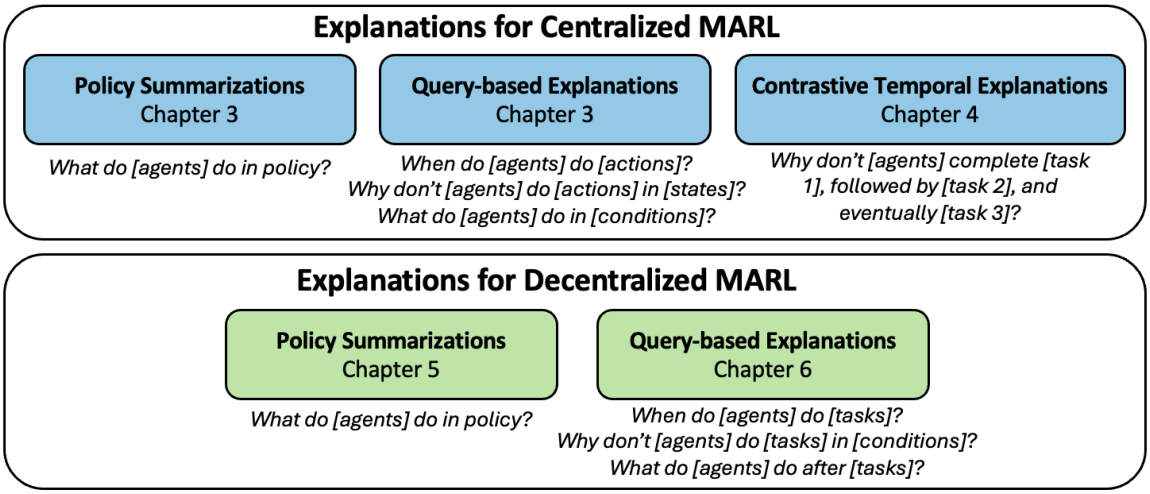


Figure 1.1: Dissertation Overview

1.5 Credits and Other Publications

Some of this work was developed with other researchers and has been published as such. Yet, I am the main contributor for all of the research presented in the dissertation. I credit my colleagues below for their contributions.

The work in chapter 3 was developed under the guidance of Dr. Lu Feng and Dr. Sarit Kraus. However, I am solely responsible for the development of both the summarization and explanation generation algorithms. I also designed and ran the presented computational experiments and user study. This work is published in IJCAI 2022 [12].

Chapter 4 also presents work that was developed under the guidance of Dr. Lu Feng and Dr. Sarit Kraus. Yet, I am still solely responsible for the developed contrastive temporal explanation algorithm, computational experiments, and user study. This work is published in IJCAI 2023 [13].

Chapter 5 was developed with the help of Dr. Lu Feng, Dr. Sarit Kraus, Dr. Seongkook Heo, and Erzhen Hu. I am responsible for the development of the summarization algorithm and implementation for computational experiments. Furthermore, I am solely responsible for designing, running, and analyzing the summarization effectiveness study. I designed the augmented reality user study. However, Erzhen Hu generated the virtual reality interface where the study took place and assisted in running the in-person study. Dr. Lu Feng, Dr. Sarit Kraus, and Dr. Seongkook Heo all provided guidance on this work. This work has not yet been published.

Chapter 6, which contains still unpublished work, was developed under the guidance of Dr. Lu Feng and Dr. Sarit Kraus. I am solely responsible for the developed explanation algorithm, computational

experiments, and user study.

Finally, I have published several other works that are not present in this dissertation. In [14], I helped develop a method, inspired by social sciences, to formalize contrastive (Why action 1 and not action 2?) explanations within Markov decision processes (MDPs) using selectiveness, constrictiveness, and responsibility. Additionally, I helped formalize the notion of uncertain human preferences and present a novel approach that accounts for this uncertainty in the context of multi-objective controller synthesis for MDPs in [15]. [16] advocates for a new computational approach to establish foreseeability of autonomous systems based on the legal "BPL" formula, providing research challenges. Whereas, [17] contains a high-level overview of the work discussed in this dissertation as part of the AAAI doctoral consortium.

Chapter 2

Related Work

2.1 Explainable RL

Explaining agent decision making has recently emerged as a focus area within the explainable AI paradigm. [4] provides a survey about this emerging landscape of explainable decision making. While, [3] proposes Explainable Decisions in Multi-Agent Environments (xMASE) as a new research direction, emphasizing many challenges of generating multi-agent explanations, such as accounting for agent interactions and user satisfaction. Furthermore, explainable RL has been attracting increasing interest, as shown in several recent surveys [5]–[7], [18]. In particular, [5] points out the lack of user studies as a major limitation across existing works.

Explaining Single-Agent RL. Moreover, current approaches mostly focus on the single-agent setting, while generating summarizations and explanations for MARL has received scant attention so far. For example, [19] develops Abstracted Policy Graphs (i.e., Markov chains of abstract states) for summarizing and explaining RL, [20] summarizes agent behavior by extracting trajectories from agent simulations and visualizes them as videos, and [21] generates RL policy descriptions to answer queries. As stated, all three works focus on the single-agent setting which can cause significant issues when applied naively to multi-agent domains.

Explaining Multi-Agent RL. Several works do focus on the multi-agent setting but have significant limitations. [22] derives intrinsically interpretable decision trees and [23] extracts the MARL model as abstract argumentations. However, neither explicitly considers agent cooperation on the same tasks. [24] 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. Furthermore, [25] displays the actions each agent performs in a joint plan and [26] describes an architecture for parsimonious explanations for multiple BDI agents. Yet, both of these works assume that tasks can be completed without agent cooperation and naively aggregate information from multiple agents to produce an explanation. We assume cooperation between agents and apply effective methods to properly combine information.

2.2 Types of Explanations

Existing works can be categorized according to different axes (e.g., timing, scope, form). We position our proposed approach based on these categorizations as follows.

Timing. First, there are *intrinsic* and *post-hoc* methods depending on the timing when the explanation is generated. The former (e.g., [27], [28]) builds intrinsically interpretable policies (e.g., represented as decision trees) at the time of training, while the latter (e.g., [21], [29]) generates post-hoc explanations after a policy has been trained. All our proposed approaches belong to the latter.

Scope. Second, existing works can be distinguished by the scope of explanations. Some methods provide explanations about policy-level behaviors (e.g., [19], [20]), while others explain specific, local decisions (e.g., [30], [31]). Our work focuses on explaining both global behavior via summarizations and local behavior via query-based explanations.

Form. Additionally, current approaches generate explanations in diverse forms, including natural language [21], saliency maps [32], reward decomposition [33], finite-state machines [34], and others. Our proposed approaches generate summarization tables, Hasse diagrams, and natural language explanations using both language templates and large-language models.

2.3 Contrastive Explanations

[35] identifies being *contrastive* ("Why A but not B?") as one of the key desired properties of an explanation as it greatly improves user understanding. The research thread on contrastive explanations for RL has been drawing increasing attention since then. For example, [31] generates contrastive explanations for *"why action"* and *"why not action"* queries via counterfactual analysis of a structural causal model; [36] develops a deep RL architecture with an embedded self-prediction model to explain why a learned agent prefers one action over another; and [30] computes counterfactual state explanations (i.e., minimal changes needed for an alternative action).

By contrast, several recent works [29], [37] generate policy-level contrastive explanations in the single-agent setting. However, these approaches have limited scalability in multi-agent environments due to computational complexity. [29] requires a large number of samples generated via a random walk to find missing preconditions. [37] 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 for all single-agent contrastive methods may not capture agent cooperation requirements that are essential for understanding multi-agent behaviors. So, none of these methods are suitable for MARL.

Our proposed approach for basic ("why not?") local contrastive explanations for centralized MARL

can be found in chapter 3, and basic local contrastive explanations for decentralized MARL can be found in chapter 6. Our contrastive temporal explanations found in chapter 4 advance the state of the art by developing approaches for generating contrastive explanations about MARL agents' global behaviors.

2.4 Explanation Presentation

How the explanation is presented is just as important as the method to generate it, as the presentation of information can increase user understanding, satisfaction, and confidence, while decreasing cognitive load [38]. As stated previously, current approaches generate explanations in diverse forms.

Augmented reality (AR) is commonly used to provide interactions with multi-modal information in a 3D environment [39]. Given the benefits of reducing cognitive load and improving task performance using augmented overlays [40], [41], AR has been increasingly used in human-robot interaction for enhanced decision support [42]. A recent work [43] demonstrates an AR-based XAI approach, providing context-aware explanations like route suggestions for jogging. However, most AR-based XAI studies focus on single-agent settings and do not address the challenges of presenting complex multi-agent explanations. [44] shows an example of an augmented reality system for providing action summaries for a single reinforcement learning agent to promote human-agent collaboration. We explore the use of AR for explainable MARL in chapter 5.

Furthermore, other novel technologies such as large language models (LLMs) can allow for explanations that can be condensed or adapted to meet a user's needs [45]. We plan to utilize these techniques for multi-agent explanations in chapter 6.

Chapter 3

Policy Explanations for Centralized MARL

3.1 Overview

In this chapter, we develop novel methods to generate two types of policy explanations for MARL: (i) policy summarization, and (ii) query-based language explanations. Our methods rely on first building an abstract representation of a MARL policy as a multi-agent Markov decision process (MMDP), which can be obtained by abstracting samples observed during the MARL policy evaluation.

The proposed summarization method generates a summarization about the most probable sequence of agent behavior under a given MARL policy, by finding the most probable path through the MMDP abstraction and extracting information about agent cooperation and task sequence. Additionally, the developed methods for generating language explanations answer three types of queries about agent behavior, including "When do [agents] do [actions]?", "Why don't [agents] do [actions] in [states]?", "What do [agents] do in [conditions]?"

Our work is inspired by the method proposed in [21], which computes a minimal Boolean logic expression covering states satisfying the query criteria, and converts the Boolean expression to explanations via language templates. However, we find that a naive adaptation of this method for MARL generates explanations with redundant information and has limited scalability. Further, the generated explanations do not necessarily capture agent cooperation, which is imperative for explaining MARL. We proposed improved methods that address these limitations by leveraging MARL domain knowledge (e.g., agent cooperation requirements) to filter relevant agent states and actions.

We applied a prototype implementation of the proposed methods to three benchmark MARL domains: (i) multi-robot search and rescue, (ii) multi-robot warehouse, and (iii) level-based foraging [11]. Experimental results demonstrate that the proposed methods can generate policy summarizations and query-based explanations for large MARL environments with up to 19 agents.

Finally, we conducted a user study to evaluate the quality of generated explanations. We measured user performance on correctly answering questions based on explanations, to test user understanding of agent behavior. We also collected user subjective ratings on *explanation goodness metrics* [46]. The results show that the generated explanations significantly improve user performance and increase

subjective ratings on various metrics including user satisfaction.

3.2 Policy Abstraction

In the context of MARL, a group of N agents interact with each other in a common environment and make decisions influenced by the joint states of all agents. Agent decisions can be captured by a joint policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, which is a function mapping the set of joint states $\mathcal{X} = \{(x^1, \ldots, x^N)\}$ to a probabilistic distribution over the set of joint actions $\mathcal{A} = \{(a^1, \ldots, a^N)\}$, where x^i (*resp.* a^i) denotes the state (*resp.* action) of agent i. Once a policy is trained, agents can act upon it in any given state. But there is a lack of a global view of the entire policy. Further, the size of the policy grows exponentially with the number of agents and state variables. To address these issues, we propose to build an abstract representation of the policy as the basis for generating explanations about agent behavior.

We use the multi-agent Markov decision process (MMDP) framework to represent MARL policy abstraction. Formally, an MMDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, where $\mathcal{S} = \{(s^1, \ldots, s^N)\}$ is the joint (abstract) state space, $\mathcal{A}$ is the joint action space, and $\mathcal{T}$ is the transition function. Let $\mathcal{F}$ be a set of Boolean predicates indicating features of the MARL domain. We denote by $f(x^i) = 1$ if an agent state x^i satisfies a feature predicate $f \in \mathcal{F}$. An abstract state s^i is then given by the satisfaction of all feature predicates $f \in \mathcal{F}$, where each bit of the binary encoding of $s^i \in \mathbb{N}$ corresponds to the satisfaction of a predicate $f(x^i)$. Thus, the choice of features affects the abstraction level and should include adequate information for explanations. In this work, we assume that users specify a set of feature predicates for a given MARL domain.

Once an MARL policy is trained, we build an MMDP during the policy evaluation stage. For each sample (x, a, x'), determine an MMDP transition $s \xrightarrow{a} s'$ by finding the abstract state s (*resp.* s') corresponding to x (*resp.* x'). When policy evaluation terminates (e.g., converging to the expected reward), compute the transition probability $\mathcal{T}(s, a, s')$ via frequency counting.

Properties. The resulting MMDP is a *sound* abstraction of the MARL policy because, by construction, every MMDP transition with non-zero probability corresponds to at least one sampled policy decision. The state space size $|\mathcal{S}|$ is bounded by $\mathcal{O}(2^{|\mathcal{F}|^N})$, depending on the number of agents N and feature predicates $|\mathcal{F}|$. In practice, a trained MARL policy may only induce a small set of reachable states.

Example 1. Figure 3.1(a) shows an example MARL domain where three robotic agents cooperate to complete search and rescue tasks. Rescuing the victim requires the cooperation of an unmanned aerial vehicle (UAV) and an unmanned ground vehicle (UGV). Any agent can fight the fire, which is blocked by the wall and obstacle. Removing the obstacle requires the cooperation of two UGVs.

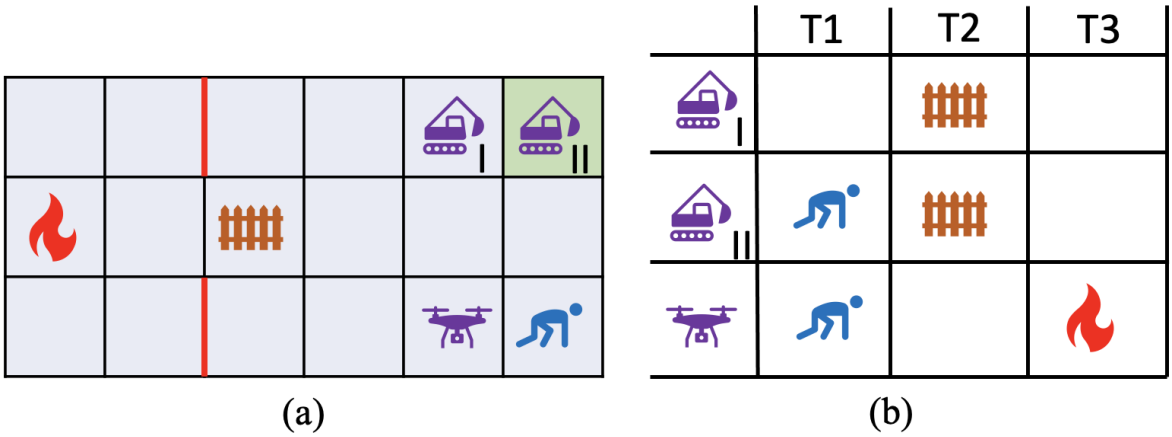


Figure 3.1: Example MARL domain of multi-robot search and rescue.

Algorithm 1 Generating Policy Summarization

Input: policy abstraction $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T})$, task completion predicates $\mathcal{F}_c \subseteq \mathcal{F}$
Output: policy summarization $\mathcal{Z}$

$\mathcal{Z} \leftarrow \{\}$
Compute the most probable path ρ through $\mathcal{M}$
for $0 \leq t \leq |\rho|$ **do**
    $y \leftarrow$ new array
    for $1 \leq i \leq N$ **do**
        $y[i] \leftarrow \{\}$
        for $f \in \mathcal{F}_c$ **do**
            if agent state s_t^i in the path ρ satisfies f **then**
                insert f to $y[i]$
    insert non-empty array y to $\mathcal{Z}$
return $\mathcal{Z}$

Given a trained MARL policy, we build an MMDP abstraction with 6 feature predicates indicating whether each task is detected or completed (e.g., victim_detect, victim_complete). An agent can only detect a task in a neighboring grid (e.g., UAV detects the victim in Figure 3.1(a)). The resulting MMDP has 63 (reachable) states and 577 transitions.

3.3 Policy Summarization

A policy abstraction containing hundreds of states and transitions is too complex for humans to understand. An alternative way of communicating agent behavior is to show execution traces; however, a lengthy trace may be burdensome for users to review. To overcome these limitations, we develop a method to generate policy summarization, illustrating the agent cooperation and task sequence for the most probable sequence of agent behavior under a given MARL policy.

Algorithm 1 shows the proposed method, which takes the input of a policy abstraction $\mathcal{M}$ and a

set of predicates $\mathcal{F}_c$ representing the completion of tasks (subgoals) in a given MARL domain. The first step is to compute the most probable path $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ from the initial state to a goal state in the MMDP $\mathcal{M}$, which represents the most probable sequence of agent decisions under the policy. This problem can be solved by converting the MMDP to a directed weighted graph with edge weight $e(s, a, s') = -\log \mathcal{T}(s, a, s')$ for each transition, and then applying the Dijkstra's algorithm [47] to find the shortest path.

Next, the algorithm loops through every joint state s_t in the path ρ to extract the agent cooperation and task sequence. At each step t, the algorithm checks if an agent state s_t^i satisfies any task completion predicate $f \in \mathcal{F}_c$ and inserts completed task f into the array element $y[i]$ (line 4-9). An agent only satisfies a task completion predicate at step t when it finishes the task and receives a reward. We assume that if a task is completed via the cooperation of multiple agents, they must satisfy the task predicate f at the same step t and each receive a portion of the reward. Thus, the agent cooperation is represented as multiple elements of the array y sharing the same task. Only non-empty arrays containing completed tasks are inserted into the summarization $\mathcal{Z}$. When the algorithm terminates, the generated summarization is visualized as a chart, with each column corresponding to a non-empty y-array and each row representing an agent's task sequence.

Properties. The generated policy summarization $\mathcal{Z}$ is *sound*, because it is derived from the most probable path of a *sound* policy abstraction (see Section 3.2). The complexity of computing the most probable path is bounded by $\mathcal{O}(|\mathcal{S}|^2)$, following the complexity of the Dijkstra's algorithm and depending on the MMDP state space size. The rest of Algorithm 1 is bounded by $\mathcal{O}(|\rho| \cdot N \cdot |\mathcal{F}_c|)$, depending on the path length and the number of agents and tasks.

Example 2. We apply Algorithm 1 using the policy abstraction and task predicates from Example 1. There are 8 states in the most probable path from the initial state (i.e., all agents starting in the green grid) to a goal state (i.e., all tasks have been completed). Figure 3.1(b) visualizes the generated summarization, with column names (i.e., T1, T2, T3) indicating the sequence of task completions: UGV_2 and UAV cooperate to rescue the victim; next, UGV_1 and UGV_2 cooperate to remove the obstacle; and lastly, UAV fights the fire.

3.4 Policy Explanation

While policy summarization provides a global view of the agent behavior under a MARL policy, users may also query about specific agent decisions. In this section, we develop methods for generating language explanations to answer the following three types of queries:

- *"When do [agents] do [actions]?"* for identifying conditions for action(s) of a single or multiple agent(s) .

- *"Why don't [agents] do [actions] in [states]?"* for understanding differences in expected and observed behaviors of a single or multiple agent(s).
- *"What do [agents] do in [predicates]?"* for revealing agent behavior under specific conditions described by the given predicates.

Our work is inspired by a method developed in [21] to generate query-based explanations for single-agent RL. In the following, we propose new methods to tackle limitations posed by adapting this baseline method to multi-agent environments.

3.4.1 Explanations for When Query

Algorithm 2 presents both the baseline and proposed methods for answering "When do agents G_q do actions A_q?", where G_q and A_q are sets of agents and actions, respectively. The text in blue highlights changes about relevancy filters (RF) for the proposed method (called WithRF) compared to the baseline (called NoRF).

WithRF starts the algorithm (line 1-5) by identifying relevant agents G, features F, and action sets A based on domain knowledge (e.g., agent cooperation requirements). For example, consider a query "When does UAV rescue the victim?". The domain knowledge is that rescuing the victim requires the cooperation of a UAV and a UGV (Example 1). Thus, the relevant agent set G is $\{\mathsf{UVA}, \mathsf{UGV\_1}, \mathsf{UGV\_2}\}$. The relevant feature set F is $\{\mathsf{victim\_detect}, \mathsf{victim\_complete}\}$, while predicates about the fire and obstacle are irrelevant. The relevant action sets A is an array with each element representing one possible set of agent actions required for cooperation: $[\{\mathsf{UAV\_rescue}, \mathsf{UGV_1\_rescue}\}, \{\mathsf{UAV\_rescue}, \mathsf{UGV_2\_rescue}\}]$, which can be generated based on the aforementioned domain knowledge about agent cooperation requirements.

Both NoRF and WithRF loop through all the joint states $s \in \mathcal{S}$ of the policy abstraction MMDP and check all the *enabled* (i.e., with non-zero transition probability) joint actions a in state s. In line 9 of Algorithm 2, NoRF checks if a is *compatible* with A_q; that is, every agent action $a \in A_q$ is contained in the joint action $a = (a^1, \ldots, a^N)$. By contrast, WithRF checks if a is compatible with at least one set of relevant actions contained in the array A. Following the previous example, NoRF checks if a contains $\mathsf{UAV\_rescue}$, while WithRF checks if a contains $\{\mathsf{UAV\_rescue}, \mathsf{UGV_1\_rescue}\}$ *or* $\{\mathsf{UAV\_rescue}, \mathsf{UGV_2\_rescue}\}$. Since each element of A is a super-set of A_q, the WithRF check is more restrictive.

If a state s has at least one enabled action a passing the aforementioned checks, s is inserted to the target states set V; and to the non-target states set $\bar{V}$ otherwise. The intuition is that the generated explanations should describe target states satisfying the query criteria and exclude conditions of non-target states. WithRF poses further restrictions that target states need to satisfy criteria captured by

Algorithm 2 Generating Query-Based Explanations

Input: policy abstraction $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, query "when do agents G_q do actions A_q?"
Output: explanations $\mathcal{E}$

1: $G \leftarrow \{\}$; $F \leftarrow \{\}$; $A \leftarrow [\{\}]$
2: **for all** agent action $a^i \in A_q$ **do**
3: insert all relevant agents of a^i to G
4: insert all relevant features of a^i to F
5: insert all relevant action sets of a^i to A
6: $V \leftarrow \{\}$; $\bar{V} \leftarrow \{\}$
7: **for all** joint state $s \in \mathcal{S}$ **do**
8: **for all** joint action a enabled in s **do**
9: **if** a is compatible with A_q [replace A_q with A] **then**
10: insert s to V
11: **else**
12: insert s to $\bar{V}$
13: $B_1 \leftarrow$ States2Boolean(V); $B_0 \leftarrow$ States2Boolean($\bar{V}$)
14: $\phi \leftarrow$ Quine-McCluskey(ones=B_1, zeros=B_0)
15: translate ϕ to explanations $\mathcal{E}$ via language templates
16: **return** $\mathcal{E}$
17: **function** State2Boolean(W)
18: $B \leftarrow \{\}$
19: **for all** $s = (s^1, \ldots, s^N) \in W$ **do**
20: **for** $1 \leq i \leq N$ [replace with $i \in G$] **do**
21: $C \leftarrow$ feature predicate valuations of state s^i
22: **for** $f \in \mathcal{F}$ [replace with $f \in F$] **do**
23: insert $C(f)$ to B
24: **return** B

relevant actions A, such as agent cooperation requirements. Thus, explanations generated by WithRF can provide information about agent cooperation, which may be missed by NoRF explanations.

Next, the algorithm converts the states set V (*resp.* $\bar{V}$) to a list of Boolean formulas B_1 (*resp.* B_0) via the function described in line 17-24. Given a joint state $s = (s^1, \ldots, s^N)$, NoRF finds valuations of every feature predicates $f \in \mathcal{F}$ for all agent state s^i and insert them to the list B. By contrast, WithRF only inserts to B the valuations of relevant features $f \in F$ in relevant agent states s^i for all $i \in G$. Following the previous example, WithRF only considers Boolean formulas related to relevant feature predicates $\{\mathsf{victim\_detect}, \mathsf{victim\_complete}\}$, filtering out features related to the fire and obstacle.

Lastly, the algorithm supplies Boolean formulas B_1 and B_0 to the Quine-McCluskey algorithm [48] and obtains a minimized Boolean formula, which can be translated into language explanations following [21]. The runtime of Quine-McCluskey grows exponentially with the number of variables. Thus, WithRF is more efficient than NoRF, due to the decreased number of Boolean variables. Moreover, filtering out irrelevant agents and features helps WithRF to prevent redundant information in the generated explanations.

Properties. Following the Quine-McCluskey, the complexity of NoRF is bounded by $\mathcal{O}\big(3^{N \cdot |\mathcal{F}|} / \ln(N \cdot$

Query	Explanations generated by NoRF (baseline)	Explanations generated by WithRF (proposed)
When does UAV rescue the victim?	UAV rescues the victim when UAV detects the victim *and* UGV_1 does not detect the fire, *or* UAV detects the victim *and* UGV_1 does not detect the obstacle.	UAV rescues the victim when UAV detects the victim *and* UGV_1 detects the victim, *or* UAV detects the victim *and* UGV_2 detects the victim.
Why don't UGV_1 and UGV_2 remove the obstacle in this state?	UGV_1 and UGV_2 don't remove the obstacle in this state because UGV_1 does not detect the obstacle.	UGV_1 and UGV_2 don't remove the obstacle in this state because UGV_1 does not detect the obstacle *and* UGV_2 does not detect the obstacle.
What does UAV do when it detects the victim?	UAV can rescue the victim, move, or wait when it detects the victim.	UAV is most likely to rescue the victim when it detects the victim.

Table 3.1: Examples of query-based explanations

$|\mathcal{F}|)\big)$. The complexity of WithRF is reduced to $\mathcal{O}\big(3^{|G|\cdot|F|}/\ln(|G|\cdot|F|)\big)$.

Example 3. Table 3.1 (first row) shows the explanations generated by NoRF and WithRF for a when query. The NoRF explanation contains redundant information about the fire and obstacle that are irrelevant to the query. The WithRF explanation completely captures the required agent cooperation for the query, which is missed by the NoRF explanation.

3.4.2 Explanations for Why Not Query

The query "Why don't agents G_q do actions A_q in the joint State s_q?" can be answered by modifying Algorithm 2 as follows. In line 10, adding s to $\bar{V}$ instead of V. Remove line 11-12 and add a new line for inserting the query state s_q to V. The modified algorithm is shown below.

Algorithm 3 generates an explanation describing the differences between the observed behavior in the target query state s_q and the expected behavior of states with actions compatible with the query actions A_q (NoRF) or relevant action sets A (WithRF). The complexity of the modified algorithm follows Algorithm 2.

Example 4. Table 3.1 (second row) shows the explanations generated by NoRF and WithRF for a why not query about the behavior of two agents UGV_1 and UGV_2 in the state shown in Figure 3.1(a). The WithRF explanation captures the required agent cooperation for removing the obstacle, while the NoRF explanation fails to provide such information.

Algorithm 3 Answering "why not" query

Input: policy abstraction $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T})$, query "why don't agents G_q do actions A_q in the joint State s_q"

Output: language explanations $\mathcal{E}$

$G \leftarrow \{\}$; $F \leftarrow \{\}$; $A \leftarrow [\{\}]$
for all agent action $a^i \in A_q$ **do**
    insert all relevant agents of a^i to G
    insert all relevant features of a^i to F
    insert all relevant action sets of a^i to A
$V \leftarrow \{\}$; $\bar{V} \leftarrow \{\}$
insert s_q to V
for all joint state $s \in \mathcal{S}$ **do**
    for all joint action a enabled in s **do**
        if a is compatible with A_q [replace A_q with A] **then**
            insert s to $\bar{V}$
$B_1 \leftarrow$ States2Boolean(V); $B_0 \leftarrow$ States2Boolean($\bar{V}$)
$\phi \leftarrow$ Quine-McCluskey(ones=B_1, zeros=B_0)
translate ϕ to explanations $\mathcal{E}$ via language templates
return $\mathcal{E}$
function State2Boolean(W)
    $B \leftarrow \{\}$
    for all $s = (s^1, \ldots, s^N) \in W$ **do**
        for $1 \leq i \leq N$ [replace with $i \in G$] **do**
            $C \leftarrow$ feature predicate valuations of state s^i
            for $f \in \mathcal{F}$ [replace with $f \in F$] **do**
                insert $C(f)$ to B
    return B

3.4.3 Explanations for What Query

Algorithm 4 answers the query "What do agents G_q do when satisfying predicates F_q?". We first identify all the satisfying joint states $s = (s^1, \ldots, s^N)$; that is, for all $i \in G_q$, agent state s^i satisfies predicates F_q. The baseline NoRF method is to generate a list of all possible enabled actions for agents G_q in these states. The proposed WithRF method improves the baseline by filtering agent actions that are relevant to predicates F_q and finding the most likely relevant agent actions from the list via frequency counting. Since the proposed methods do not need to call the Quine-McCluskey, the complexity of both NoRF and WithRF are only bounded by $\mathcal{O}(|G_q| \cdot |\mathcal{S}| \cdot |\mathcal{A}|)$, depending on the number of query agents, joint state space, and joint action space of the policy abstraction.

Example 5. Table 3.1 (third row) shows the explanations generated by NoRF and WithRF for a what query. The WithRF explanation is more concise than the NoRF explanation and only contains relevant action for the query predicate.

Algorithm 4 Answering "what" query

Input: policy abstraction $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T})$, query "what do agents G_q do when satisfying predicates F_q"
Output: language explanations $\mathcal{E}$

$A_q \leftarrow \{\}$
$\alpha \leftarrow$ find all relevant actions of predicates F_q
for all joint state $s = (s^1, \ldots, s^N) \in \mathcal{S}$ **do**
  if s^i satisfies F_q for all $i \in G_q$ **then**
    for all $a = (a^1, \ldots, a^N)$ enabled in s **do**
      for all $i \in G_q$ **do**
        insert a^i to A_q [only if $a^i \in \alpha$]
$A_q \leftarrow$ the most frequent action for each agent in A_q
generate explanations $\mathcal{E}$ with A_q via language templates
return $\mathcal{E}$

3.5 Computational Experiments

We implemented and applied the proposed methods to three MARL domains. The first domain is multi-robot search and rescue (SR) similar to Example 1. The second and third domains are benchmarks taken from [11]. Multi-robot warehouse (RWARE) considers multiple robotic agents cooperatively delivering requested items. Level-based foraging (LBF) considers a mixed cooperative-competitive game where agents must navigate a grid world to collect randomly scattered food. Our implementation used the Shared Experience Actor-Critic [10] for MARL policy training and evaluation. To simulate a centralized policy, each agent views the entire global state and has access to the actions of other agents. All models were trained and evaluated to 10,000 steps, or until converging to the expected reward, whichever occurred first. The experiments were run on a laptop with a 1.4 GHz Quad-Core Intel i5 processor and 8 GB RAM.

Table 3.2 shows experimental results. For each domain, we report the number of agents N and the number of states $|\mathcal{S}|$ and transitions $|\mathcal{T}|$ of generated policy abstraction MMDP. It is unsurprising that the MMDP size grows exponentially with the number of agents. We report the most probable path length $|\rho|$ and the chart size $|\mathcal{Z}|$ of generated policy summarizations, which are more compact and easier to interpret than complex MMDP abstractions. All summarizations were generated within 1 second (thus not shown in the table). Additionally, we compare NoRF and WithRF methods in terms of the number of clauses in the generated query-based explanations and runtime. The results show that WithRF is more succinct in general and has better scalability than NoRF. In particular, NoRF failed to generate explanations for "when" and "why not" queries within an hour for large cases with more than 4 agents, while WithRF generated explanations for all cases within seconds. Both methods generate explanations for "what" queries efficiently, thanks to the lower complexity than other queries.

In summary, experimental results demonstrate that the proposed methods can generate policy

Case Study		MMDP		Summarization		When Query			
				Path	Chart	NoRF		WithRF	
Domain	N	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\rho\rvert$	$\lvert\mathcal{Z}\rvert$	$\lvert\mathcal{E}\rvert$	Time (ms)	$\lvert\mathcal{E}\rvert$	Time (ms)
SR	3	63	577	8	3x2	4	72.4	4	1.2
	4	732	5,048	23	4x4	-	timeout	6	31.4
	5	839	4,985	24	5x4	-	timeout	10	73.6
RWARE	2	8	62	5	2x2	4	2.6	4	2.7
	4	16	387	5	4x1	12	3,321.0	5	82.7
	19	114	1,500	15	19x2	-	timeout	7	3,890.6
LBF	2	5	13	4	2x2	2	0.7	2	0.7
	4	15	355	5	4x1	10	3,598.5	2	1.4
	9	482	5,841	13	9x2	-	timeout	2	200.2

Case Study		Why Not Query				What Query			
		NoRF		WithRF		NoRF		WithRF	
Domain	N	$\lvert\mathcal{E}\rvert$	Time (ms)	$\lvert\mathcal{E}\rvert$	Time (ms)	$\lvert\mathcal{E}\rvert$	Time (ms)	$\lvert\mathcal{E}\rvert$	Time (ms)
SR	3	1	86.8	2	0.8	3	1.7	1	1.3
	4	-	timeout	3	14.9	3	15.7	1	14.5
	5	-	timeout	4	23.4	6	24.6	1	19.8
RWARE	2	4	2.0	2	0.4	3	0.1	1	0.1
	4	6	23.0	3	1.3	3	0.1	1	0.1
	19	-	timeout	4	56.5	3	21.7	1	20.1
LBF	2	3	0.4	2	0.3	2	0.1	1	0.1
	4	8	3,695.0	2	1.6	3	0.6	1	0.6
	9	-	timeout	2	101.3	2	24.9	1	19.8

Table 3.2: Experimental results on three MARL domains (timeout set as one hour).

summarizations and query-based explanations for large MARL domains (e.g., RWARE with 19 agents, which is the largest number of possible agents in the provided environments).

3.6 User Study

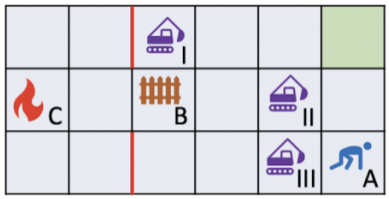
We conducted a user study [1] via the Qualtrics to evaluate the quality of generated explanations. We describe the study design in Section 3.6.1 and analyze results in Section 3.6.2.

3.6.1 Study Design

Participants. We recruited 116 eligible participants (i.e., fluent English speakers over the age of 18) through university mailing lists. 62.1% of participants self-identified as male, 37.1% as female, and 0.8% preferred not to say. The age distribution is 76(18-24), 31(25-34), 7(35-49), 2(50-64). Participants were instructed to answer multiple-choice questions about agent behavior for multi-robot search and rescue tasks. They were incentivized with bonus payments to answer questions correctly

[1] This study was approved by University of Virginia Institutional Review Boards IRB-SBS #4701.

based on the provided explanations. To ensure data quality, attention checks were injected during the study. Figures 3.2 through 3.6 show examples of the user interface. The gif associated with Figure 3.6 can be found at https://github.com/kjboggess/IJCAI2022/blob/main/MissionGifExample.gif.

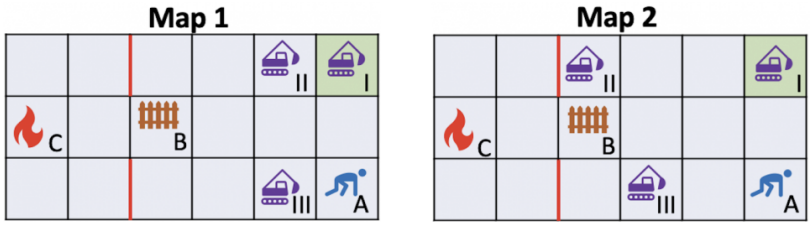


Explanation: Robot III rescues victim A when robot III detects victim A *and* robot I detects victim A, *or* robot III detects victim A *and* robot II detects victim A.

Question: Based on the explanation, does robot III rescue victim A in the above map?

- ○ Yes
- ○ No

Figure 3.2: Question based on explanations for a "when" query.



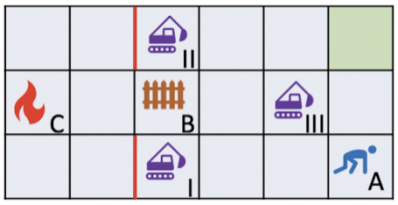
Explanation: Robot I and robot II don't remove obstacle B in **map 1** because robot I does not detect obstacle B *and* robot II does not detect obstacle B.

Question: Based on the explanation, do robot I and robot II remove obstacle B in **map 2**?

- ○ Yes
- ○ No

Figure 3.3: Question based on explanations for a "why not" query.

Independent variables. We employed a within-subject study design with the explanation generation methods as independent variables. Participants were asked to complete two trials for evaluating policy summarizations. They were presented with charts generated by Algorithm 1 in one trial, and GIF animations illustrating the most probable sequence of agent behavior (i.e., visualization of the most probable path in the policy abstraction) in the other trial. For each trial, there were two questions about agent behavior in various environments (i.e., 3×6 and 6×6 grid world). Questions used in the two trials are different but had similar difficulty. All participants were presented with the same



Explanation: Robot III is most likely to move when it does not detect victim A.

Question: Based on the explanation, what does robot III most likely do in the above map?

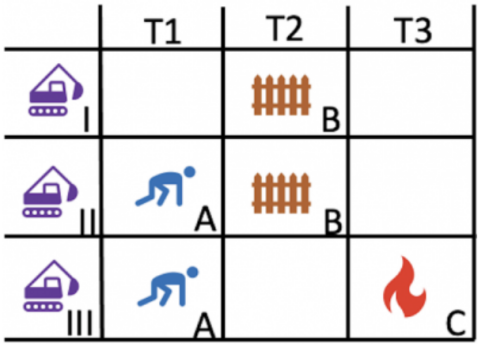
- ○ Wait
- ○ Move
- ○ Rescue Victim
- ○ Fight Fire
- ○ Remove Obstacle
- ○ ⊘ I don't know

Figure 3.4: Question based on explanations for a "what" query.

set of four randomly generated questions for summarization trials. To counterbalance the ordering confound effect, they were randomly assigned to answer the first two questions based on either charts or GIF, and the other two questions based on the remaining method. Additionally, participants were asked to complete two trials for evaluating query-based explanations generated by NoRF and WithRF methods, with 6 questions (2 environments $\times$ 3 query types) in each trial. Participants answered the same set of 12 randomly generated questions for query-based trials, and were randomly assigned to different groups similarly to summarization trials.

Dependent measures. We measured *user performance* by counting the number of correctly answered questions in each trial. In addition, at the end of each trial, participants were asked to rate in a 5-point Likert scale (1 - strongly disagree, 5 - strongly agree) about *explanation goodness* metrics (i.e., understanding, satisfaction, detail, completeness, actionability, reliability, trust) adapted from [46]. The questions asked are as follows:

- The explanations help me *understand* how the team of robots completes the search and rescue mission.
- The explanations are *satisfying*.
- The explanations are sufficiently *detailed*.
- The explanations are sufficiently *complete*, that is, they provide me with all the needed information to answer the questions.



Question: Which robots are required to remove obstacle B?

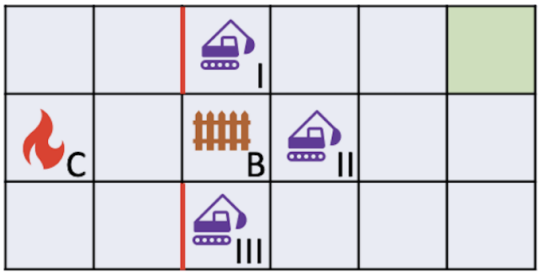
- Robot I
- Robot II
- Robot III
- Robot I and Robot II
- Robot I and Robot III
- Robot II and Robot III
- Robot I and Robot II and Robot III

Figure 3.5: Question based on policy summarization (sequence chart).

- The explanations are *actionable*, that is, they help me know how to answer the questions.
- The explanations let me know how *reliable* the robot team is for completing the mission.
- The explanations let me know how *trustworthy* the robot team is for completing the mission.

Hypotheses. We make the following hypotheses in this study.

- **H1:** Chart-based summarizations lead to better user performance than GIF-based.
- **H2:** Chart-based summarizations yield higher user ratings on explanation goodness metrics than GIF-based.
- **H3:** Query-based explanations generated by WithRF lead to better user performance than those by NoRF.



Question: Which robots are required to remove obstacle B?

- Robot I
- Robot II
- Robot III
- Robot I and Robot II
- Robot I and Robot III
- Robot II and Robot III
- Robot I and Robot II and Robot III

Figure 3.6: Question based on a policy summarization (GIF animation)

- **H4:** Query-based explanations generated by WithRF yield higher user ratings on explanation goodness metrics than those by NoRF.

3.6.2 Results Analysis

We used a paired t-test to evaluate hypotheses H1 and H3, and used the Wilcoxon Signed-rank test to evaluate hypotheses H2 and H4. We set the significant level as $\alpha = 0.05$.

Evaluating policy summarizations. Participants answered more questions correctly with chart-based summarizations (M=1.8 out of 2, SD=0.6) than GIF-based (M=0.9 out of 2, SD=0.4), with statistical significance ($t(462)$=-15.8, $p \leq$0.01, d=1.5). *Thus, the data supports H1.*

Figure 3.7 shows average participant ratings about summarizations. Chart-based summarizations yield higher ratings on the perceived completeness than GIF-based with statistical significance (W=371.5, Z=-2.4, $p \leq$0.02, r=-0.2). But no significant difference was found regarding the other

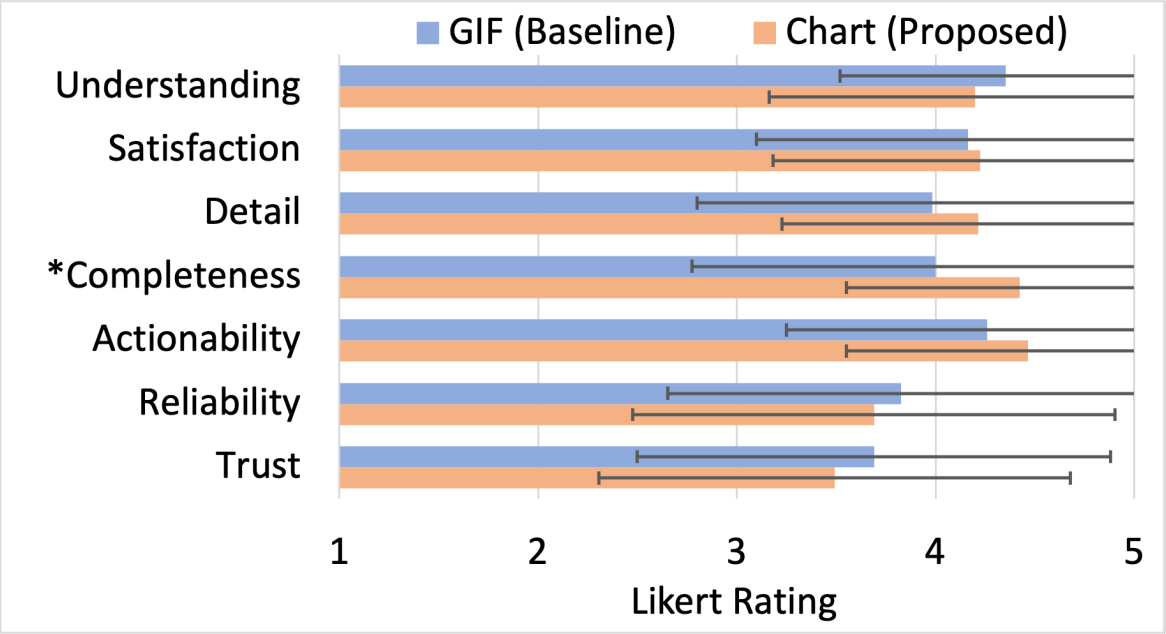


Figure 3.7: Mean and SD of participant ratings about policy summarizations ("*" indicates statistically significant difference).

metrics. *Thus, the data partially supports H2.*

Evaluating query-based explanations. Participants answered more questions correctly with explanations generated by WithRF (M=5.2 out of 6, SD=1.7) than NoRF (M=2.3 out of 6, SD=1.0), with statistical significance ($t(1390)$=-21.1, $p \leq 0.01$,d=2.0). *Thus, the data supports H3.*

Figure 3.8 shows that participants gave higher average ratings to WithRF explanations than NoRF explanations. The Wilcoxon test found significant differences on all metrics: understanding (W=319.5, Z=-4.9, $p \leq 0.01$, r=-0.3), satisfaction (W=266.0, Z=-7.0, $p \leq 0.01$, r=-0.5), detail (W=484.0, Z=-3.7, $p \leq 0.01$, r=-0.2), completeness (W=494.5, Z=-6.4, $p \leq 0.01$, r=-0.4), actionability (W=167.0, Z=-6.9, $p \leq 0.01$, r=-0.5), reliability (W=382.5, Z=-3.6, $p \leq 0.01$, r=-0.2), and trust (W=217.0, Z=-3.4, $p \leq 0.01$, r=-0.2). *Thus, the data supports H4.*

Summary. We accept all hypotheses except H2 based on the statistical analysis. The user study shows that our proposed approaches, both query-based and summarizations, are able to improve task performance when trying to determine agent behavior in multi-agent situations. Additionally, participants find our proposed query-based explanations subjectively better in goodness than those produced by the query baseline. However, based on participant preference, we find little to no subjective difference between our proposed summarizations explanations and the summary baseline.

Discussion. In summary, the data supports all hypotheses, while H2 is only partially supported because the statistical test found no significant differences between chart-based and GIF-based sum-

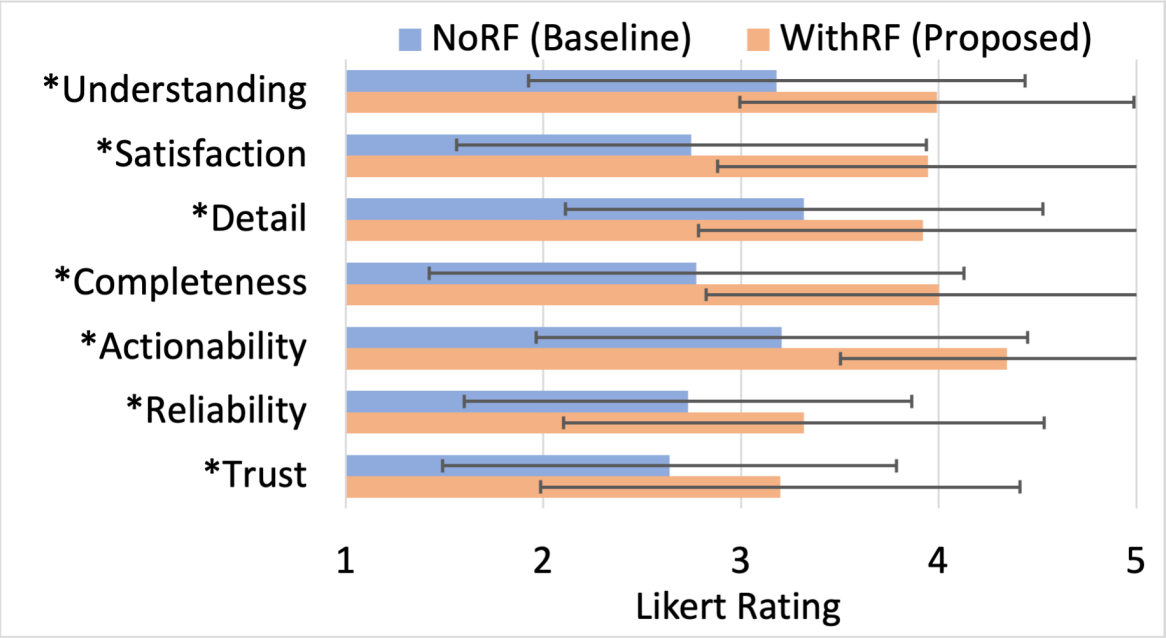


Figure 3.8: Mean and SD of participant ratings about query-based explanations ("*" indicates statistically significant difference).

marizations on most metrics. However, Figure 3.7 shows that participants rated chart-based summarizations close to 4 (agree) on all metrics, and above GIF-based ratings on all metrics except understanding, reliability, and trust. This may be because users showed a strong preference toward the moving nature of GIF animations and the visualized effects of agents completing tasks. But watching a GIF can be more time-consuming and less informative than a quick glance at the chart. This is supported by the results that participants were able to answer more questions correctly with chart-based summarizations, and they rated this method significantly higher on completeness (i.e., providing needed information). Meanwhile, query-based explanations generated by the proposed WithRF method led to significantly better user performance and higher user ratings on all metrics, because users prefer succinct WithRF explanations with adequate information about agent behavior and cooperation. By contrast, NoRF explanations do not necessarily provide essential information about agent cooperation for correctly answering questions, and may contain redundant information that decreases user satisfaction.

3.7 Summary

In this chapter, we developed methods to generate policy summarizations and query-based explanations for MARL. Experimental results on three MARL domains demonstrate the scalability of our methods. Evaluation via a user study shows that our generated MARL policy explanations can im-

prove user understanding about agent behavior and enable them to answer more questions correctly, while maintaining very positive ratings on explanation goodness metrics.

Chapter 4

Temporal Queries for Centralized MARL

4.1 Overview

Although we provide a method to generate explanations for basic queries in chapter 3, existing explainable methods cannot handle more advanced temporal queries either. These queries involve 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 temporal explanations for RL in the single-agent setting. [29] 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. [37] 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.

However, [29] requires a large number of samples generated via a random walk to find missing preconditions. [37] 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.

In this chapter, we develop 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* [49]. 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.

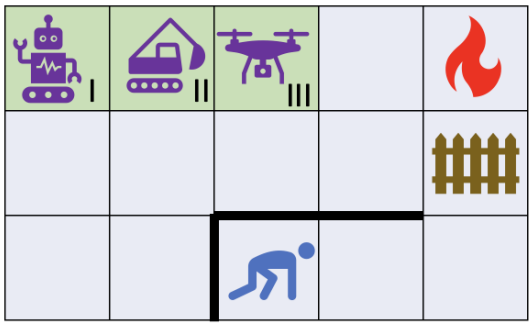
Additionally, we conducted a user study to evaluate the quality of generated explanations, where we adapted [29] to generate baseline explanations. The study results show that, compared with the baseline, explanations generated using our approach significantly improve user performance (measured by the number of correctly answered questions) and yield higher average user ratings on *explanation goodness metrics* (e.g., understanding, satisfaction) [46].

4.2 Problem 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 $\mathcal{X} = \{x = (x^1, \ldots, x^N)\}$ to a distribution over a set of joint actions $\mathcal{A} = \{a = (a^1, \ldots, a^N)\}$. Execution of policy π yields a sequence of joint states and joint actions $x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \cdots$ where $a_t \sim \pi(\cdot|x_t)$ 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 $R^i(x_t, a_t, x_{t+1}) > 0$ 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 4.1(a). We can compute a high-level plan by applying the policy summarization method proposed in [12]. Figure 4.1(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.

The user may not desire the presented plan and raise an alternative query. The user query does not



(a)

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

(b)

Figure 4.1: Example MARL domain and a high-level plan.

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: 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.

To tackle this problem, we present an approach as illustrated in Algorithm 5. We describe the construction of a policy abstraction (line 1) in Section 4.3, the encoding and checking of the user query (lines 2-5) in Section 4.4, guided rollout (lines 6-9) in Section 4.5, and explanation generation (lines 10-11) in Section 4.6. Additionally, we analyze the correctness and complexity of the approach in Section 4.7.

4.3 Policy Abstraction

Given a trained MARL policy π, we construct a multi-agent Markov decision process (MMDP) abstraction following the policy abstraction method described in 3.2. We denote an MMDP as a tuple

Algorithm 5 Checking the feasibility of a user query

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

```
construct a policy abstraction MMDP M given π
encode the temporal query ρ as a PCTL* formula φ
if M satisfies φ then
    return YES
else
    M' ← update M via guided rollout (Algorithm 6)
    if M' satisfies φ then
        return YES
    else
        generate explanations E (Algorithm 7)
        return E
```

$\mathcal{M} = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{T}, \mathcal{L})$ with a set of joint abstract states $\mathcal{S}$, an initial state $s_0 \in \mathcal{S}$, 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 $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ starting from the initial state s_0.

The state space $\mathcal{S} = \{s = (s^1, \ldots, s^N)\}$ is defined over a set of Boolean predicates indicating whether a task $g \in G$ has been completed by agent i. The initial state s_0 represents that none of the tasks has been completed. In the example MMDP shown in Figure 4.2, the initial state is $s_0 = (000, 000, 000)$. State $s_1 = (000, 100, 100)$ represents that the fire task has been completed by robotic agents II and III, which is labeled with $\mathcal{L}(s_1) = \{\mathsf{fire\_robotII\_robotIII}\}$. The next state $s_2 = (010, 110, 100)$ is labeled with $\mathcal{L}(s_2) = \{\mathsf{obstacle\_robotI\_robotII}\}$, which only contains the newly completed obstacle task.

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

4.4 Query Checking with Temporal Logic

We encode a temporal user query $\rho = \langle \tau_1, \tau_2, \cdots \rangle$ as a PCTL* logic [50] formula φ with a "sequencing" specification template as follows.

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

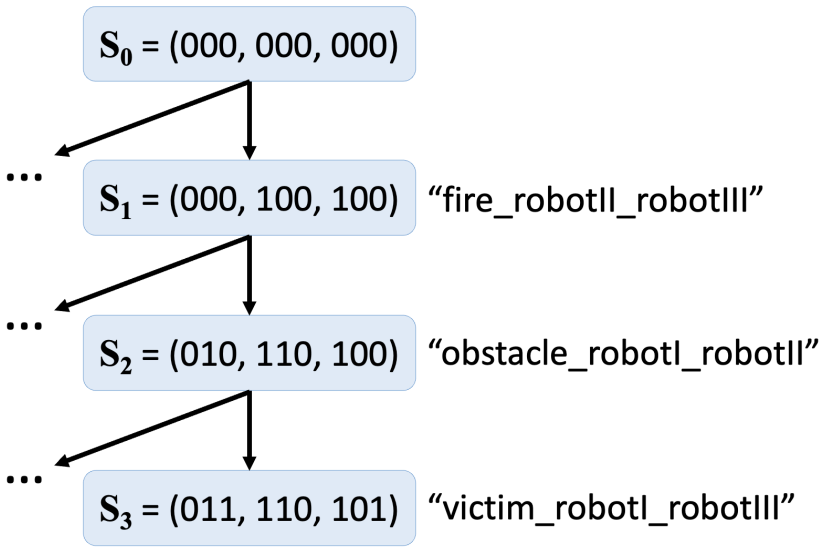


Figure 4.2: Fragment of an example MMDP.

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 4.2 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* [49] 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 5 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.

4.5 Guided Rollout

Algorithm 6 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 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 $\langle \tau_1 =$

Algorithm 6 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: $\quad s \leftarrow \mathcal{N}$.pop(0)
5: $\quad x \leftarrow$ pick a corresponding joint state from $X(s)$
6: $\quad \delta \leftarrow$ a rollout execution of π from x with `DepthLimit`
7: $\quad$ update the MMDP with samples in δ
8: **return** the updated MMDP $\mathcal{M}'$

$\mathsf{fire\_robotII\_robotIII}, \tau_2 = \mathsf{obstacle\_robotII}\rangle$, unfolding the MMDP in Figure 4.2 yields $U(s_0) = 0$, $U(s_1) = 1$ for conforming with τ_1, and $U(s_2) = -\infty$ 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 $C(s)$. 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 $s \in \mathcal{N}$, we (randomly) pick a corresponding joint state $x \in X(s)$ and generate a rollout execution $\delta = x \xrightarrow{a} x' \xrightarrow{a'} \cdots$ of the policy π starting from 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 6). When the number of rollout executions hits a predefined parameter `RolloutNum`, Algorithm 6 terminates with an updated MMDP, denoted by $\mathcal{M}'$.

We check if $\mathcal{M}'$ satisfies a PCTL* formula φ encoding the user query ρ (line 7 of Algorithm 5) as described in Section 4.4. 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.

4.6 Explanation Generation

Algorithm 7 shows the explanation generation procedure. Given the updated MMDP $\mathcal{M}'$ resulting from Algorithm 6, we unfold $\mathcal{M}'$ as a search tree and assign a U value to each tree node following Section 4.5. 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 $\langle \tau_1 = \mathsf{obstacle\_robotI\_robotII}, \tau_2 =$

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

$\mathsf{victim\_robotI}, \tau_3 = \mathsf{fire\_robotII\_robotIII}\rangle$, 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 [48], 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 s_2 in Figure 4.2 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 $\phi = \mathsf{fire\_robotII} \wedge \mathsf{fire\_robotIII} \wedge \mathsf{obstacle\_robotI} \wedge \mathsf{obstacle\_robotII}$. Recall our assumption in Section 4.2 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 4.4. If the model checker yields YES, then Algorithm 7 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 s_3 in Figure 4.2 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 $s_0 \to s_1 \to s_2 \to s_3$ in Figure 4.2 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 7.

4.7 Correctness and Complexity

Correctness. The correctness of our proposed approach, with respect to the problem formulated in Section 4.2, is proved below.

Proposition 6. *Given a temporal user query ρ and a trained MARL policy π, if Algorithm 5 returns YES, then the query ρ must be feasible under the policy π; otherwise, Algorithm 5 generates correct and complete explanations $\mathcal{E}$.*

Proof. We prove the following two cases.

Case 1: When Algorithm 5 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 (s, a, s') in $\mathcal{M}$ or $\mathcal{M}'$ with non-zero probability maps to at least one sampled decision (x, a, 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 5 returns explanations $\mathcal{E}$ generated via Algorithm 7. As described in Section 4.6, Algorithm 7 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 7 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). □

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 4.4 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 [51]. 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 3.2), given a well-trained MARL policy.

- The time complexity of guided rollout (Algorithm 6) is given by $\mathcal{O}(\texttt{RolloutNum} \cdot \texttt{DepthLimit})$. 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 7) 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 [52].

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.

4.8 Computational Experiments

To demonstrate the scalability of our approach, we developed a prototype implementation and applied it to four benchmark MARL domains [2].

(1) *Search and Rescue (SR)*, where multiple robotic agents cooperate to complete tasks such as fighting fires and rescuing victims [12].

(2) *Level-Based Foraging (LBF)*, where agents play a mixed cooperative-competitive game to collect food scattered in a gridworld [11].

(3) *Multi-Robot Warehouse (RWARE)*, where robots collaboratively move and deliver requested goods [11].

(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 [53].

Our prototype implementation used the Shared Experience Actor-Critic [10] for MARL policy training and evaluation. To simulate a centralized policy, each agent views the entire global state and has access to the actions of other agents. All models were trained and evaluated until converging to the expected reward, or up to 10,000 steps, whichever occurred first. The PRISM probabilistic

[2] 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 4.1: Experimental results on four benchmark MARL domains.

model checker [54] was applied for checking the feasibility of user queries. We set the guided rollout parameters as `RolloutNum` $= 10$ and `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 4.1 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 5. For infeasible queries, we also report the number of failures λ, which were controlled to grow with the environment size as the longer the 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

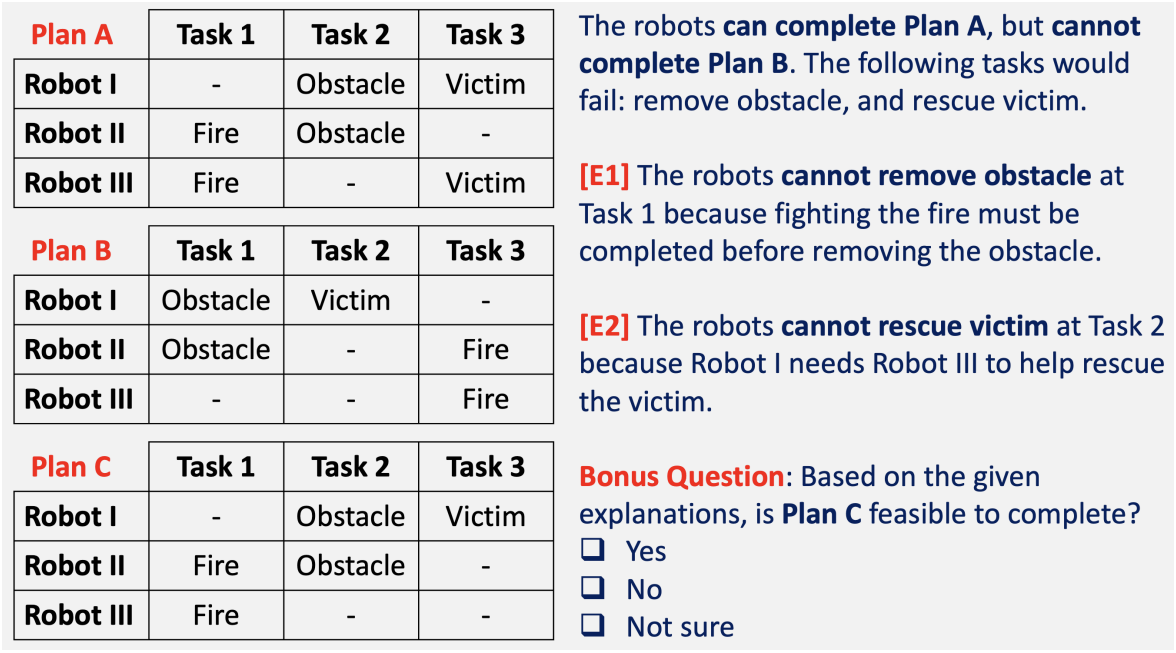
to various benchmark MARL domains with a large number of agents (e.g., up to 9 agents in the PLATE domain), for checking the feasibility of temporal user queries and generating explanations when needed.

4.9 User Study

We evaluate the quality of generated reconciliation explanations via a user study. [3] Section 4.9.1 describes the study design and Section 4.9.2 analyzes the results.

4.9.1 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 4.3 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.



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

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

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

Figure 4.3: Example of the user study interface displaying explanations generated by the proposed approach.

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

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.

Baseline. We adapted the explanation generation method in [29], 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 4.3 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 4.3) 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 4.3 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 [46].

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

4.9.2 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 4.4 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

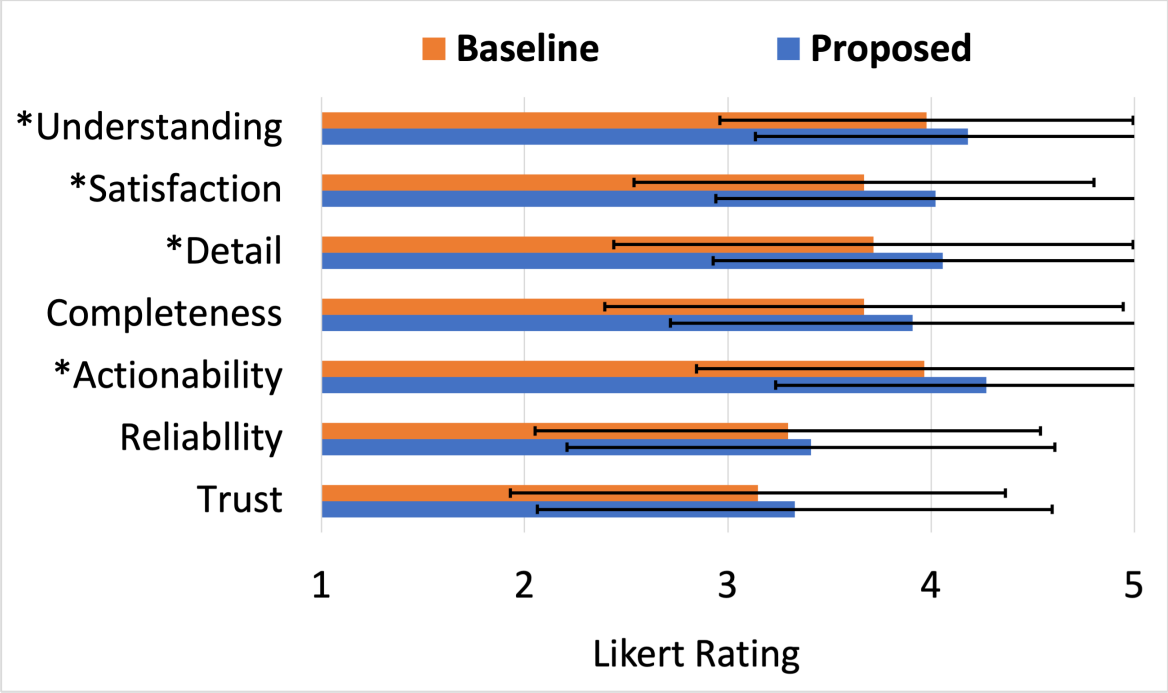


Figure 4.4: Mean and SD of participant ratings on explanation goodness metrics ("*" indicates statistically significant difference with the significant level set as $\alpha = 0.05$).

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.

4.10 Summary

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.

Chapter 5

Policy Summarizations for Decentralized MARL

5.1 Overview

Now that we have discussed how to generate summarizations and explanations for centralized MARL, in this chapter we present how to generate summarizations for decentralized MARL, which is significantly more challenging. To tackle this challenge, we develop a novel approach that generates a Hasse diagram as a policy summarization from a set of agents' trajectories resulting from executing decentralized MARL policies. We show that our approach can generate *correct* and *complete* policy summarizations. Correctness means that the generated policy summarization does not include spurious behaviors (e.g., tasks, agent cooperation) not present in the actual agents' trajectories, while completeness means that the policy summarization captures every agent's execution.

We evaluate the scalability and efficiency of our proposed approach through computational experiments across a variety of benchmark MARL domains, involving different numbers of agents and tasks in gridworlds. We also employ two distinct decentralized MARL algorithms to demonstrate the compatibility of our approach.

Finally, we assess the the generated policy summarizations through several user studies. First, we evaluate the effectiveness of the generated policy summarizations, citing improved user performance. Then, we specifically explore using an AR technique called position overlay to enhance the presentation of policy summarizations in complex, multi-agent environments. The study results highlight benefits such as improved question-answering performance, reduced cognitive load, and enhanced user satisfaction.

5.2 Problem Formulation

Decentralized MARL policies. Consider a group of N MARL agents, each with a trained policy $\pi^i : s^i \to \Delta(a^i)$ mapping local states s^i to a distribution over actions a^i for agent $i \in [1, N]$. Decentralized execution of these policies for an episode yields a set of trajectories $\{\omega^i\}_{i=1}^N$, where $\omega^i = s_0^i, a_0^i, r_0^i, s_1^i, \cdots$ is a trajectory for agent i. Starting from an initial state s_0^i, agent i samples an action $a_t^i \sim \pi^i(\cdot|s_t^i)$ from its policy at time t, receives a reward r_t^i, and observes a new state s_{t+1}^i from

the environment *which could be influenced by other agents' executions.* A trajectory ends when the terminal criterion of an episode is satisfied (e.g., all tasks have been completed, or after a maximum number of allowed time steps).

Based on an agent's execution trajectory ω^i, we can extract a sequence of tasks completed by agent i, denoted by $\mathsf{trace}(\omega^i) = \tau_1^i, \tau_2^i, \cdots$ where τ_k^i is the k-th task completed by agent i. For example, agent i receives a positive reward $r_t^i > 0$ after an action $a_t^i = \mathit{remove\_obstacle}$, and observes that the obstacle has been removed through the transition from state s_t^i to state s_{t+1}^i, then we say that agent i completes the task of removing the obstacle at time t.

Different agents' trajectories may be asynchronous due to their decentralized execution and lack of global clock, which introduces nondeterminism into the ordering of tasks completed by different agents. We assume that if a task is completed through the cooperation of multiple agents, they must complete it simultaneously, and each agent receives a portion of the reward, but they only observe their own portion of that reward.

Policy summarization via Hasse diagrams. To help users understand complex agents' behaviors under decentralized MARL policies, we propose to generate a policy summarization as a Hasse diagram [55]. Formally, a Hasse diagram is a directed acyclic graph $\mathcal{D} = (V, E)$ with vertices V and edges E, representing a finite partially ordered set, in the form of a drawing of its transitive reduction (i.e., with the minimal number of edges). Each vertex $v \in V$ is defined as a hash table mapping tasks to agents who cooperate to complete them. Tasks completed simultaneously are grouped within the same vertex. Each edge $(v, v') \in E$ represents a preorder relation $v \prec v'$, indicating that tasks in vertex v are completed before tasks in vertex v'. A path through the Hasse diagram is a finite sequence of edges starting from the initial vertex v_0, which represents an empty set indicating that no task has been completed yet.

Figure 5.1(e) shows an example Hasse diagram with seven vertices, representing a policy summarization of four agents completing six tasks together. Each vertex in the diagram, excluding the initial vertex v_0, contains one task and agents that cooperate to complete the task. Each edge represents a preorder relation of task completion time. For example, the edge $v_1 \to v_2$ indicates that task A is completed before task B. The branching of edges in the Hasse diagram illustrates the nondeterminism of task orders resulting from agents' decentralized execution of policies. For example, there are nondeterministic choices of successor vertices at v_1, indicating that both tasks B and C should occur after Task A. However, the order between tasks B and C, which are completed by different groups of agents, is uncertain.

Given a path $\rho = v_0 \to v_1 \to \cdots$ through the Hasse diagram, we define its projection onto an agent i, denoted by ρ^i, as a sequence of tasks obtained by iterating through each vertex in the path ρ

and retaining only the tasks completed by agent i. A path projection ρ^i conforms to the task sequence $\mathsf{trace}(\omega^i)$ for agent i, denoted by $\rho^i \sqsubseteq \mathsf{trace}(\omega^i)$, if and only if every task $\tau \in \rho^i$ is present in $\mathsf{trace}(\omega^i)$ and the preorder relations between the completion times of tasks in ρ^i are preserved in $\mathsf{trace}(\omega^i)$. For example, consider the path $\rho = v_0 \to v_1 \to v_2 \to v_4 \to v_6$ shown in Figure 5.1(e). Its projection onto agent 1 yields $\rho^1 = A, B, F$, which exactly matches the task sequence for agent 1. The projection of this path onto agent 2 yields $\rho^2 = A, D$, which conforms to the task sequence $\mathsf{trace}(\omega^2) = A, C, D$ for agent 2, that is, $\rho^2 \sqsubseteq \mathsf{trace}(\omega^2)$.

Problem. Given a set of trajectories $\{\omega^i\}_{i=1}^N$ yielded by an episode execution of decentralized MARL policies $\{\pi^i\}_{i=1}^N$, we say that a Hasse diagram $\mathcal{D} = (V, E)$ is a *correct* policy summarization if, for every path ρ through the Hasse diagram, when its projection ρ^i onto an agent i is non-empty, ρ^i conforms to the task sequence $\mathsf{trace}(\omega^i)$ for agent i, that is, $\rho^i \sqsubseteq \mathsf{trace}(\omega^i)$. On the other hand, we say that a Hasse diagram $\mathcal{D} = (V, E)$ is a *complete* policy summarization if, for every agent trajectory ω^i, there exists at least one path ρ through $\mathcal{D}$ such that its projection ρ^i matches the task sequence $\mathsf{trace}(\omega^i)$, that is, $\rho^i = \mathsf{trace}(\omega^i)$. This work aims to solve the problem of generating correct and complete policy summarization for decentralized MARL policies.

5.3 Approach

We present an approach, as illustrated in Algorithm 8, for generating a Hasse diagram $\mathcal{D} = (V, E)$ as a correct and complete policy summarization, given a set of trajectories $\{\omega^i\}_{i=1}^N$ produced by an episode execution of decentralized MARL policies $\{\pi^i\}_{i=1}^N$. These policies can be learned using either CTDE or DTDE algorithms. Algorithm 8 is agnostic to the underlying decentralized MARL algorithms, provided that a set of agent execution trajectories $\{\omega^i\}_{i=1}^N$ is available.

First, the algorithm initializes the Hasse diagram with an initial vertex v_0 (line 1). Then, it loops through every trajectory ω^i (line 2) and extracts a task sequence $\mathsf{trace}(\omega^i) = \tau_1^i, \tau_2^i, \cdots$ for agent i (line 3), as described previously. For each task $\tau_k^i \in \mathsf{trace}(\omega^i)$, if that task is already included in some vertex $v \in V$ in the Hasse diagram, then agent i is added to the list of cooperative agents for completing the task in v (lines 4-6). When $k > 1$ and no edge exists connecting a vertex $\bar{v} \in V$ containing the previous task τ_{k-1}^i completed by agent i to the current vertex v, a new edge $\bar{v} \to v$ is inserted into E (lines 7-10). If task τ_k^i is not included in any existing vertices in V, a new vertex v' containing task τ_k^i and agent i is created (lines 11-12). When $k = 1$, a new edge connecting the initial vertex v_0 to v' is inserted into E (line 13); and when $k > 1$, a new edge $\bar{v} \to v'$ is created to connect v' from a vertex $\bar{v} \in V$ containing the previous task τ_{k-1}^i (lines 15-17).

The algorithm yields a directed acyclic graph (DAG) $\mathcal{D} = (V, E)$ once all trajectories $\{\omega^i\}_{i=1}^N$ have been processed. Next, the algorithm continues with a procedure of transitive reduction on the

Algorithm 8 HDS - Hasse diagram-based summaries

Input: Trajectories $\{\omega^i\}_{i=1}^N$ yielded by an episode execution of decentralized MARL policies $\{\pi^i\}_{i=1}^N$
Output: Hasse diagram $\mathcal{D} = (V, E)$

1: $v_0 \leftarrow \emptyset$; $V \leftarrow \{v_0\}$; $E \leftarrow \emptyset$
2: **for** $(i = 1; i < N; i{+}{+})$ **do**
3: $\quad \mathsf{trace}(\omega^i) \leftarrow$ extract task sequence from ω^i
4: $\quad$ **for** $(k = 1; k < |\mathsf{trace}(\omega^i)|; k{+}{+})$ **do**
5: $\quad\quad$ **if** task τ_k^i is contained in a vertex $v \in V$ **then**
6: $\quad\quad\quad$ Append i to the list of agents in $v[\tau_k^i]$
7: $\quad\quad\quad$ **if** $k > 1$ **then**
8: $\quad\quad\quad\quad$ Find a vertex $\bar{v} \in V$ containing task τ_{k-1}^i
9: $\quad\quad\quad\quad$ **if** $(\bar{v}, v) \notin E$ **then**
10: $\quad\quad\quad\quad\quad$ Insert to E a new edge $\bar{v} \to v$
11: $\quad\quad$ **else**
12: $\quad\quad\quad$ Insert to V a new vertex v' with $v'[\tau_k^i] = \{i\}$
13: $\quad\quad\quad$ **if** $k = 1$ **then**
14: $\quad\quad\quad\quad$ Insert to E a new edge $v_0 \to v'$
15: $\quad\quad\quad$ **else**
16: $\quad\quad\quad\quad$ Find a vertex $\bar{v} \in V$ containing task τ_{k-1}^i
17: $\quad\quad\quad\quad$ Insert to E a new edge $\bar{v} \to v'$
18: **for all** $(v, v') \in E$ **do**
19: $\quad$ **if** there is a path $v \to v'$ excluding edge (v, v') **then**
20: $\quad\quad E = E - (v, v')$
return $\mathcal{D} = (V, E)$

DAG [56], ensuring the resulting Hasse diagram has the minimal number of edges. For every edge $(v, v') \in E$, if there is a path from v to v' in the (updated) DAG $\mathcal{D} = (V, E)$ that does not use the edge, then this edge is removed from E (lines 18-20). The aforementioned "if" condition can be checked via any linear time graph traversal algorithm like breadth-first search or depth-first search. There also exist faster procedures to compute the transitive reduction of a DAG, such as leveraging the topological ordering of vertices. We skip the details here and refer readers to [57].

Example. Now we describe a running example to show how Algorithm 8 works. Consider four agents with decentralized MARL policies $\{\pi^i\}_{i=1}^4$, each having a trajectory ω^i from one episode of policy execution. Looping through lines 2-17 of Algorithm 8, we first consider agent $i = 1$. Figure 5.1(a) shows the resulting DAG after processing the task sequence $\mathsf{trace}(\omega^1) = A, B, F$ for agent 1, where a chain of new vertices is created. Next, consider agent $i = 2$ whose task sequence is $\mathsf{trace}(\omega^2) = A, C, D$. Figure 5.1(b) shows the resulting DAG after processing $\mathsf{trace}(\omega^2)$. Since there is an existing vertex containing task A, agent 2 is appended to that vertex's list of cooperative agents. A new vertex is created for task C, and an edge is added to connect it to the vertex containing the previous task A completed by agent 2. Similarly, a new vertex is created for task D and connected to the vertex containing task C. Repeating the above process for agents 3 and 4 yields the DAGs shown in Figure 5.1(c-d). Next, the algorithm continues with a transitive reduction on the final DAG shown

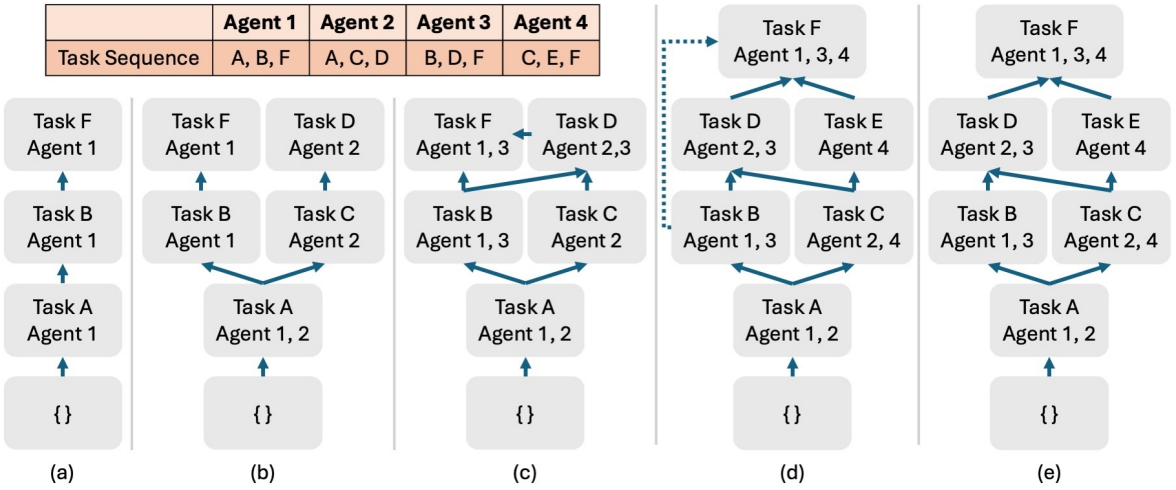


Figure 5.1: Running example for Algorithm 8.

in Figure 5.1(d). Consequently, the dashed edge $v_2 \to v_6$ is removed from the DAG since there is another path, $v_2 \to v_4 \to v_6$, connecting these two vertices. The algorithm terminates and outputs a Hasse diagram as shown in Figure 5.1(e).

Complexity. The worst case time complexity of Algorithm 8 is given by $\mathcal{O}(N \cdot T^4)$, where N is the number of agents and T is the number of tasks given by the environment.

Correctness and Completeness. The correctness and completeness of our proposed approach, with respect to the problem formulated in Section 4.2, is stated below.

Proposition 7. *Given a set of trajectories $\{\omega^i\}_{i=1}^N$ yielded by an episode execution of decentralized MARL policies $\{\pi^i\}_{i=1}^N$, the Hasse diagram $\mathcal{D} = (V, E)$ generated by Algorithm 8 is a correct and complete policy summarization.*

Proof. (Correctness) Suppose for the sake of contradiction that the Hasse diagram $\mathcal{D} = (V, E)$ generated by Algorithm 8 is not a correct policy summarization. Then, by definition, there must exist a path ρ through the Hasse diagram whose projection ρ^i onto some agent i does not conform to the task sequence $\mathsf{trace}(\omega^i)$. Suppose the non-conformance arises from a path fragment from a vertex v containing task τ_k^i to a vertex v' with task $\tau_{k'}^i$, where $k > k'$, indicating that task τ_k^i is completed after task $\tau_{k'}^i$. By the construction of the Hasse diagram, there is a preorder relation $v \prec v'$ such that tasks contained in v are completed before those in v'. Thus, k should be smaller than k', which is a contradiction.

(Completeness) By construction, Algorithm 8 loops through every task sequence $\mathsf{trace}(\omega^i)$ extracted from all trajectories $\{\omega^i\}_{i=1}^N$, and there exists an edge in the original DAG $\mathcal{D} = (V, E)$ connecting each pair of consecutive tasks in $\mathsf{trace}(\omega^i)$. When the transitive reduction occurs, an edge (v, v') is only removed if there is a path connecting v to v'. Thus, for every trajectory ω^i, there exists

at least one path ρ through $\mathcal{D}$ such that its projection ρ^i matches $\mathsf{trace}(\omega^i)$. By definition, the Hasse diagram $\mathcal{D} = (V, E)$ generated by Algorithm 8 is a complete policy summarization. □

Remark. When there are many agents and tasks, the generated policy summarization can overwhelm the user. In real-world applications like the search and rescue example described in the introduction, the human operator may only be interested in a subset of agents and tasks (e.g., nearby robots). Our approach can be easily adapted to generate policy summarizations for a selected group of agents and tasks as follows: The input to Algorithm 8 would only include trajectories of selected agents, and when extracting the task sequence from a trajectory (line 3), a filter can be applied to preserve only the relevant tasks.

Moreover, as our experiments will show, multiple episodes of policy execution can result in different Hasse diagrams. To summarize tasks already completed by agents, we can use the actual execution trajectories. For summarizing possible future behaviors, we can simulate several episodes of policy execution and report one of the resulting Hasse diagrams, such as the most frequent.

5.4 Computational Experiments

		CTDE						
		HDS				Baseline		
Domain	$\|N\|$,$\|T\|$	$\|V\|$	$\|E\|$	$\|\{\mathcal{D}\}\|$	Time (s)	$\|V\|$	$\|E\|$	Time (s)
SR	2,3	4.0±0.0	3.0±0.0	3	0.02	7.0±2.0	6.0±2.0	0.007
	9,7	8.0±0.0	7.88±0.78	100	0.1	59.33±10.21	58.33±10.12	0.05
LBF	5,5	6.0±0.0	5.54±0.67	98	0.09	67.2±5.49	66.2±5.49	0.04
	9,9	10.0±0.0	10.83±0.97	100	0.12	80.33±28.01	79.33±28.01	0.09
RW	3,4	5.0±0.0	4.0±0.0	35	0.04	23.0±5.66	22.0±5.66	0.01
	4,19	20.0±0.0	19.0±0.0	100	0.2	318.5±32.66	317.5±32.66	0.2
PP	4,4	5.0±0.0	4.0±0.0	64	0.07	19.0±4.47	18.0±4.47	0.01
	7,6	7.0±0.0	6.0±0.0	99	0.1	37.86±5.25	36.86±5.25	0.03

		DTDE						
		HDS				Baseline		
Domain	$\|N\|$,$\|T\|$	$\|V\|$	$\|E\|$	$\|\{\mathcal{D}\}\|$	Time (s)	$\|V\|$	$\|E\|$	Time (s)
SR	2,3	4.0±0.0	3.0±0.0	3	0.01	7.0±2.0	6.0±2.0	0.007
	9,7	8.0±0.0	7.79±0.77	100	0.1	38.89±18.80	37.89±18.89	0.05
LBF	5,5	6.0±0.0	5.62±0.64	71	0.07	30.2±11.16	29.2±11.16	0.03
RW	3,4	5.0±0.0	4.0±0.0	28	0.04	23.0±10.71	22.0±10.71	0.01
PP	4,4	5.0±0.0	4.0±0.0	37	0.04	16.5±8.67	15.5±8.67	0.01
	7,6	7.0±0.0	6.0±0.0	40	0.06	15.29±17.88	14.29±17.88	0.02

Table 5.1: Results of computational experiments on HDS summarization method

MARL domains. We evaluate the proposed approach through computational experiments on four benchmark MARL domains: (1) *Search and Rescue (SR)*, where multiple agents cooperate to complete assigned search and rescue tasks [12]; (2) *Level-Based Foraging (LBF)*, a mixed cooperative-competitive game where agents collect food [11]; (3) *Multi-Robot Warehouse (RW)*, where multiple agents cooperate to collect and deliver items [11]; and (4) *Pressure Plate (PP)*, where some agents press switches to open doorways in a maze, enabling other agents to navigate to a goal [53]. All these domains are based on gridworld. To highlight the need for decentralized execution, we assume agents can observe only one neighboring grid cell in each cardinal direction for the first three domains and up to four grid cells in any direction in the Pressure Plate domain.

Baseline. Since there are currently no summarization or explanation methods for decentralized MARL policies, we extend [58] (a single-policy summarization method) as our baseline summarization method. To do so, we generate one abstract policy graph per given agent policy and aggregate them together as suggested in the multi-agent framework in [26].

Setup. To demonstrate the proposed approaches' agnosticism to decentralized MARL algorithms, we utilized two MARL algorithms: Shared Experience Actor-Critic (SEAC) [10] for the CTDE type and Independent Advantage Actor-Critic (IA2C) [11] for the DTDE type. All models were trained until converging to the expected return or up to 400 million steps. The experiments were run on a machine with a 2.1 GHz Intel CPU, 132 GB of memory, and Ubuntu 22.04 operating system. Due to the lack of quality policies (policies did not converge) learned by the IA2C algorithm in the larger environments, LBF(9,9) and RW(4,19), results for these cases are absent.

Summarization results analysis. We execute each trained policy for 100 episodes, generating 100 Hasse diagrams. Table 5.1 shows the total run time in seconds for generating those policy summarizations using the proposed approach and the baseline. Additionally, we report the average (and standard deviation) number of vertices and edges produced in both the Hasse diagrams generated by our method and abstract policy graphs generated by the baseline. Finally, we show the number of unique Hasse diagrams generated among the 100 summarizations, for different MARL environments with varying numbers of agents N and tasks T.

The results illustrate the efficiency and scalability of the proposed approach. Although the baseline is slightly faster than our proposed method, this increase in run time is expected due to the generation of combined agent information which is not present in the baseline. However, it takes less than 1 second to generate 100 summarizations in all cases, even for the largest environment, RW(4,19), with 4 agents and 19 tasks. So, the increase in run time is not overly problematic. Furthermore, the DTDE policies take less or equal time to summarize compared to the CTDE policies because the DTDE policies have less possible combinations of agent assignment and task order, as these policies can't share information

between agents during training.

Additionally, we find that the average number of nodes and edges in the single agent diagrams of the baseline are significantly greater than the average number of nodes and edges shown in each Hasse diagram. This suggests that the summarization produced by the baseline method is significantly larger, which may cause undue cognitive burden when the user must combine the given information from multiple agents themselves.

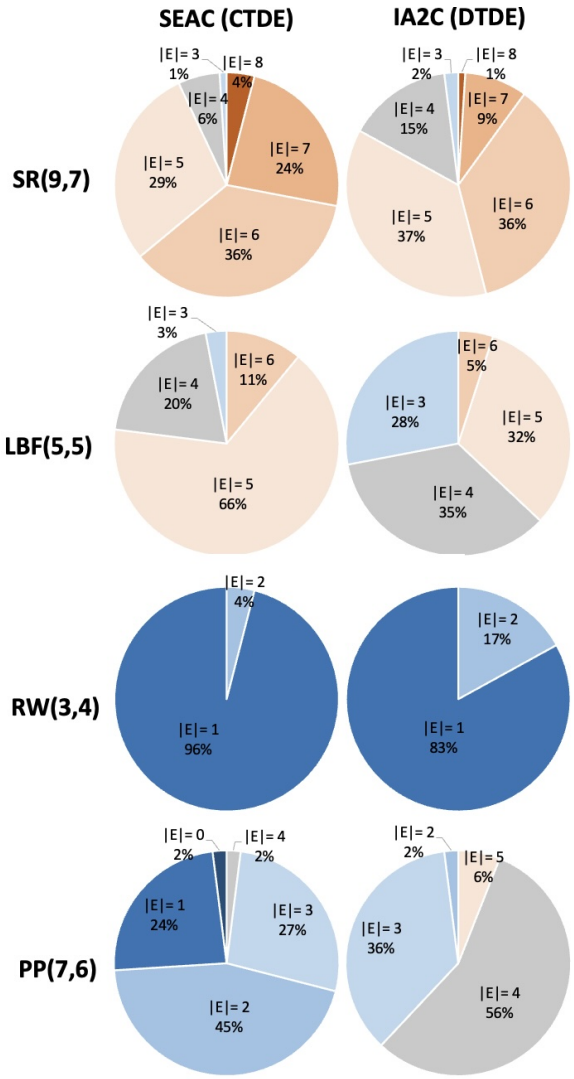


Figure 5.2: Types of generated Hasse diagrams for SR(9,7), LBF(5,5), RW(3,4), and PP(7,6).

Finally, the number of unique Hasse diagrams often increases with the environment size, likely due to more possible combinations of agent cooperation and task completion in larger environments with more agents and tasks. We also observe that more unique diagrams are generated for CTDE policies compared to DTDE policies. In some cases, like SR(9,7), every generated Hasse diagram for

each episode of policy execution is unique. However, as shown in Figure 5.2, out of 100 unique Hasse diagrams generated for SR(9,7), there are only 6 types of diagrams categorized by the number of edges (all diagrams have the same number of nodes determined by the number of tasks) for both CTDE and DTDE policy summarizations. This is because each edge in the Hasse diagram represents a pre-order relation between task completion times. For a Hasse diagram with 7 nodes, each representing a task, there is a limited number of ways to connect these nodes via edges. We observe similar patterns for other domains: for example, only 4 types for LBF(5,5) and 2 types for RW(3,4). Detailed pie charts for these domains are available in Figure 5.2.

In summary, the proposed summarization approach can be efficiently applied to various benchmark MARL domains, scaled for environments with many agents and tasks, and is compatible with policies learned via either CTDE or DTDE algorithms.

5.5 Summarization Effectiveness User Study

We conducted a user study[4] to evaluate the effectiveness of generated policy summarizations with real-world users.

5.5.1 Study Design

User interface. Users were presented with a survey via Qualtrics to evaluate the effectiveness of summarizations. Figures 5.3 and 5.4 show examples of the user interface participants were shown with a generated summarization on top and a question regarding it below.

Participants. We recruited 20 participants (10 males, 9 females, 1 other) via university mailing lists. Those eligible included fluent English speakers over 18 years old. The average age was 22.55 years (SD = 2.89). Participants were incentivized with bonus payments to answer questions correctly based on the provided summarizations.

Baseline. Since there are no summarization methods specifically for decentralized MARL, we adapted the summarization generation method found in [58], which was originally intended for a single agent domain, as a baseline. We extended the method (using suggestions from [26]) to the multi-agent by generating one abstract policy graph for each independent agent policy, showing all of them to the user. We then highlight the most common states (or most likely agent behavior) per graph for the user in red. The abstract features (task completion, agent assignment) chosen are the same for both methods so that the summarizations can be compared one-to-one.

Summarizations generated by the baseline do not portray any agent cooperation on tasks, just tasks (and their order) completed by a single agent. So, users must combine information from multiple

[4]This study was approved by the UVA Institutional Review Board with IRB-SBS #7237.

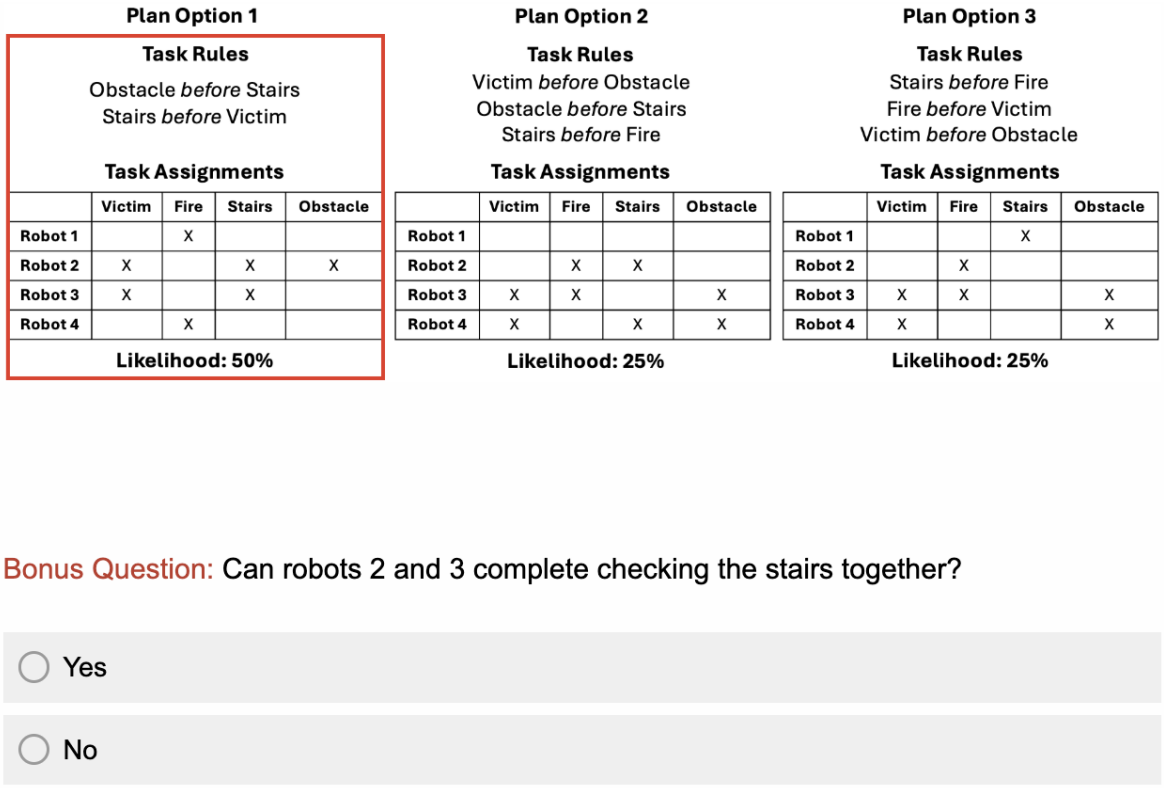


Figure 5.3: Example of user interface with HDS summarization method.

agents themselves to understand cooperation on the same tasks which may not always be clear. Furthermore, the baseline may not accurately portray the most likely set of agent behaviors, as the most likely behaviors for a single agent may not be its most likely set of behaviors when cooperating with all agents. Finally, the baseline makes it difficult to tell the true order of tasks since users are shown the task order by agent. Overall, users may not be aware of the order of tasks (or lack thereof) completed by two separate agents.

Independent variables. For the summarization study, the single independent variable was summarization type: HDS (shown in Figure 5.3) or baseline (shown in Figure 5.4). The baseline summarization is displayed as described in the selected baseline [58]. The Hasse diagram-based summarization displays the partial order represented by the Hasse diagram as a set of natural language rules and the agent-task assignment from the nodes in a simple table. One summarization is generated for each unique Hasse diagram produced by Algorithm 8 and the frequency of the diagram is shown below it as the behavior likelihood. The most likely summarization is highlighted in red for the user.

Procedures. We began each study trial (aggregation, Hasse diagram) by demonstrating how to utilize the summarization. To ensure data quality, the participant answered several attention-check questions

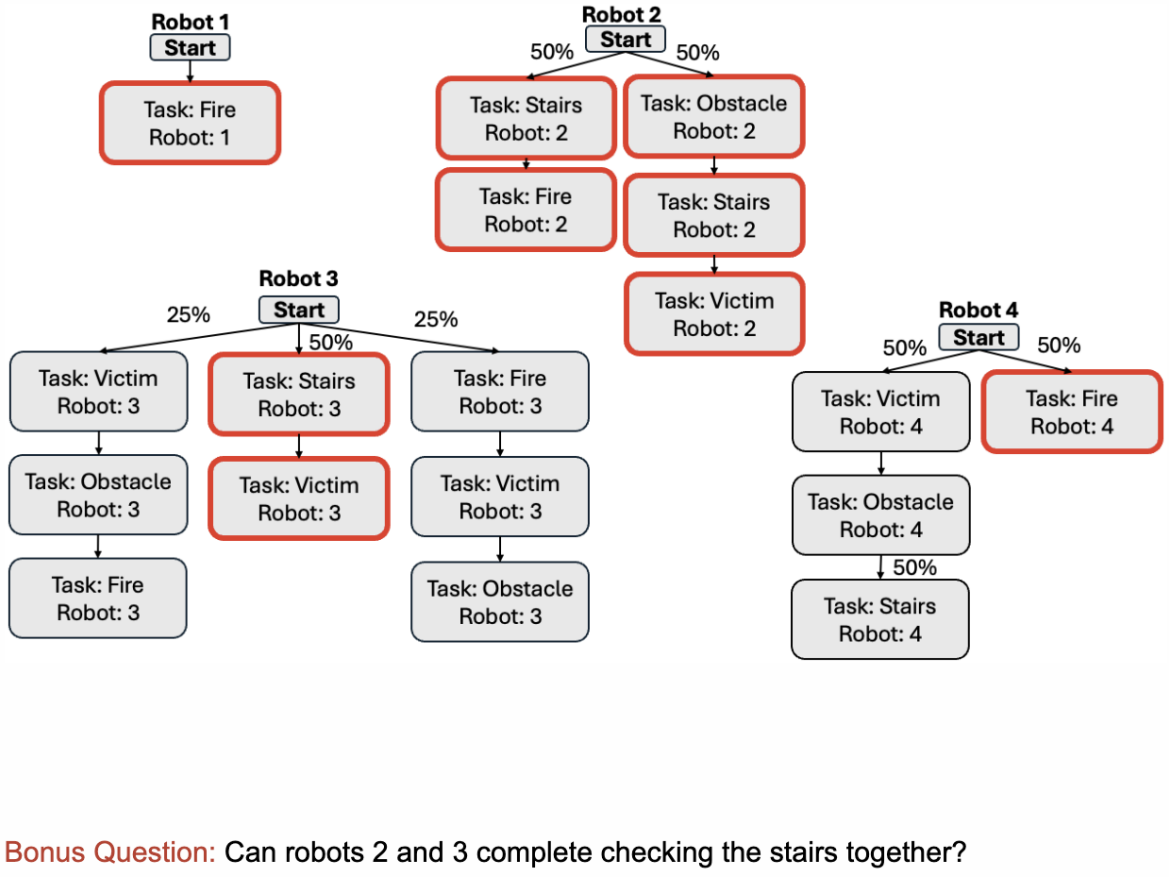


Figure 5.4: Example of user interface with baseline summarization method.

after the demonstration. During the within-subject summarization study, the participants completed two trials (one for each method), each consisting of two summarizations with three questions each (four summarizations and 12 questions in total). Participants were randomly assigned to two groups (aggregation first or Hasse diagram first) to counterbalance any ordering effects. All questions within the trials were also randomized. A demonstration was given, attention checks were injected, bonus payments were offered, and the time to complete the survey was tracked to ensure data quality.

Dependent variables. Users were asked three different types of questions regarding the given summarization: assignment (Can [robot(s)] complete [task]?), likelihood (What are the most likely robot(s) to complete [task]?), and order (Must [task 1] always be completed before [task 2]?). Since all three aspects must be understood to utilize the summarization, we counted the total number of

questions correctly answered by a participant as a performance measure. However, we also report the performance by question type. We also tracked the time to submit an answer to each question as the response time.

At the end of each trial, participants were instructed to rate the summarization on several goodness metrics on a 5-point Likert scale [46]. Participants were informed of the number of questions they answered correctly before rating the summaries. The questions presented are as follows:

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

Hypotheses. We tested three hypotheses stated below:

- **H1:** HDS summaries lead to *better user performance* than the baseline summaries.
- **H2:** HDS summaries have an *equal response time* compared to the baseline summaries.
- **H3:** HDS summaries lead to *higher ratings on summarization goodness metrics* than the baseline summaries.

5.5.2 Study Results

Question-Answering Performance. Users were able to answer more questions correctly using the HDS method (M=4.25 out of 6, SD=0.83) compared to the baseline one (M=3.1 out of 6, SD=1.04). A paired t-test ($\alpha = 0.05$) shows a statistically significant difference (t(19)=4.2, p $\leq$ 0.01, d=0.96).

Regarding each individual question type, we find no significant difference in user performance for assignment (HDS: M=1.15 out of 2, SD=0.36, Baseline: M=1.2 out of 2, SD=0.75) and order (HDS: M=1.6 out of 2, SD=0.58, Baseline: M=1.5 out of 2, SD=0.5) questions. For likelihood questions (HDS: M=1.5 out of 2, SD=0.59, Baseline: M=0.4 out of 2, SD=0.49), the HDS method leads to better performance than the baseline. A paired t-test ($\alpha = 0.05$) shows these findings for assignment

(t(19)=-0.27, p $\leq$ 0.79, d=0.06, not significant), likelihood (t(19)=6.85, p $\leq$ 0.01, d=1.57, significant), and order (t(19)=0.7, p $\leq$ 0.49, d=0.16, not significant) questions. *Thus, the data supports H1.*

Response Time. Participants spend an equal amount of time responding to questions for both the HDS (M=25.69 seconds, SD=12.47) and baseline methods (M=25.33 seconds, SD=7.88). A paired t-test ($\alpha = 0.05$) shows no statistically significant difference (t(16)=0.12, p $\leq$ 0.9, d=0.03) in the recorded response times.

Regarding each individual question type, we also find no significant difference in response time for assignment (HDS: M= 22.79 seconds, SD=10.77, Baseline: M= 19.6 seconds, SD=4.14) and order (HDS: M= 23.77 seconds, SD=13.76, Baseline: M= 24.84 seconds, SD=7.48) questions. For likelihood questions (HDS: M= 17.06 seconds, SD=4.75, Baseline: M= 25.45 seconds, SD=9.25), the HDS method is significantly faster than the baseline. A paired t-test ($\alpha = 0.05$) shows these findings for assignment (t(15)=1.17, p $\leq$ 0.26, d=0.03, not significant), likelihood (t(14)=-3.07, p $\leq$ 0.01, d=0.82, significant), and order (t(15)=-0.38, p $\leq$ 0.71, d=0.1, not significant) questions. We removed any outliers using the inter-quartile range method before the paired t-tests were performed. *Thus, the data supports H2.*

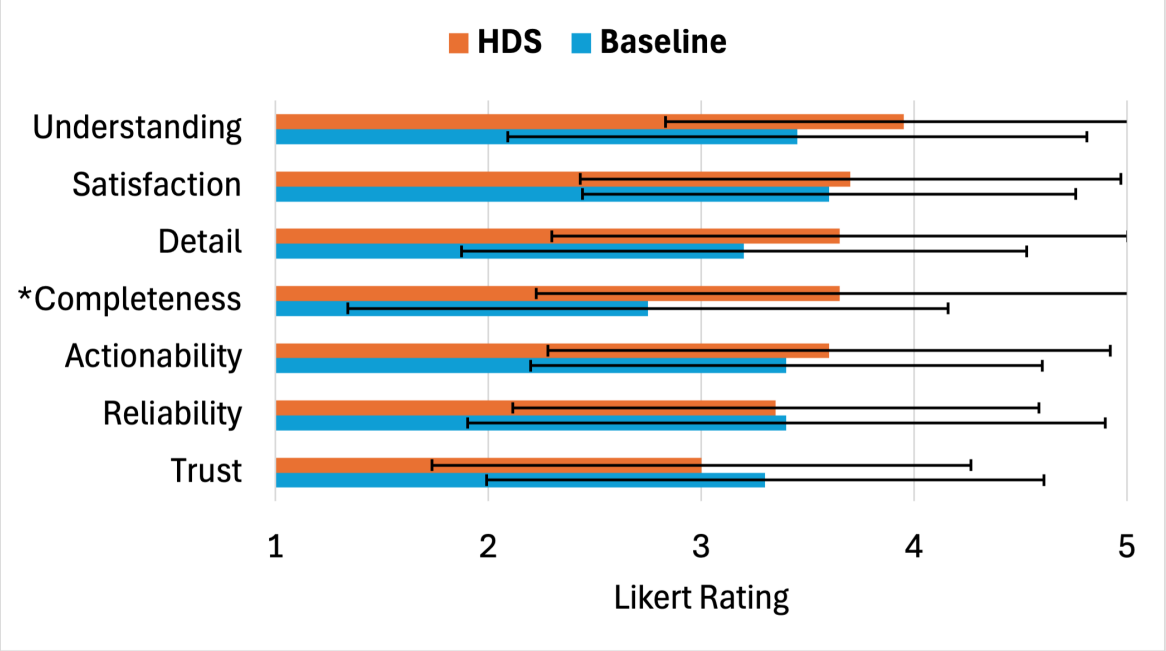


Figure 5.5: Mean and SD of participant ratings about policy summarizations ("*" indicates statistically significant difference).

Summarization Goodness Rating. Regarding summarization goodness ratings, participants only find the proposed HDS method as slightly more complete, as shown in Figure 5.5. We used the Wilcoxon signed-rank test ($\alpha = 0.05$) to evaluate hypothesis H3. As stated, there is a statistically

significant different regarding *completeness* (W=16.0, Z=-2.07, p $\leq$ 0.04, r=-0.33), but not *understanding* (W=18.0, Z=-1.12, p $\leq$ 0.26, r=-0.18), *satisfaction* (W=40.0, Z=-0.38, p $\leq$ 0.7, r=-0.06), *detail* (W=30.5, Z=-1.15, p $\leq$ 0.25, r=-0.18), *actionability* (W=28.0, Z=-0.6, p $\leq$ 0.55, r=-0.09), *reliability* (W=36.5, Z=0.27, p $\leq$ 0.79, r=0.04), or *trust* (W=22.5, Z=0.73, p $\leq$ 0.47, r=0.12). *Thus, the data rejects H3.*

Discussion. In summary, the data supports two of the three provided hypotheses. The HDS method can increase user performance without increasing response time. However, regarding summarization goodness ratings, users do not prefer the HDS method compared to the baseline.

The increased overall user performance using the HDS method is most likely due to the focus on agent cooperation and multi-agent task order that is not present in the baseline method. However, users are still able to retrieve the needed information from the baseline method as shown by the comparable performance on assignment and order questions. Yet, our data suggests that the HDS method excels at providing information about task likelihood as this is the most difficult information for users to glean on their own from disparate agent information.

Furthermore, the increased complexity of combining information from multiple agents, thus introducing possible uncertainty in the task order, does not increase response time. Our method actually may decrease user response time when cooperative information is needed as suggested by the response time shown for likelihood questions. So, users do not have increased difficulty understanding the partial order of tasks or agent assignments present in the HDS summarizations.

Finally, users show no strong preference regarding summarization goodness ratings between the two methods, only finding the HDS method as slightly more complete. This suggests that users are aware our method is more complete (containing both certain and uncertain information) compared to the baseline. Furthermore, users may also have a preference for the ease of the flowchart-based presentation of the aggregation method, influencing the subjective evaluation of goodness ratings. However, the presented information is disparate and requires users to combine information from multiple agents on their own, which may not always be accurate.

5.6 Summarization Presentation User Study

We also conducted a user study[5] to evaluate the use of augmented reality (AR) in the presentation of generated policy summarizations. We describe the study design in Section 5.6.1, present the results in Section 5.6.2, and discuss the findings in Section 5.6.3.

[5]This study was approved by the UVA Institutional Review Board with IRB-SBS #6292.

5.6.1 Study Design

User interface. The user was placed into a virtual reality (VR) environment, which simulates a real-world search and rescue environment with four simulated robotic agents following decentralized MARL policies. The user acted as a field operator who was provided a policy summarization generated by the proposed approach, helping them understand how robots cooperate to complete search and rescue tasks (e.g., rescuing victims, fighting fires).

Figure 5.6(a) shows an example of the user's first-person view through the VR headset (Meta Quest 2). For simplicity, policy summarizations used in the study were Hasse diagrams with a single path (i.e., the task order is deterministic), visualized as a chart with columns indicating the order of tasks and rows representing the agents' task sequences. It could be difficult for the user to locate agents and tasks mentioned in the policy summarization, especially in environments with a large number of agents and tasks. To address this issue, we apply an augmented reality technique called position overlay (PO), which overlays connecting lines between the actual locations of agents (or tasks) and their names in the policy summarization, as shown in Figure 5.6(b).

Additionally, the user can choose to see a static 2D blueprint of the environment through the VR headset. However, the static 2D blueprint may be inaccurate and outdated due to the complex and dynamic nature of the search and rescue environment.

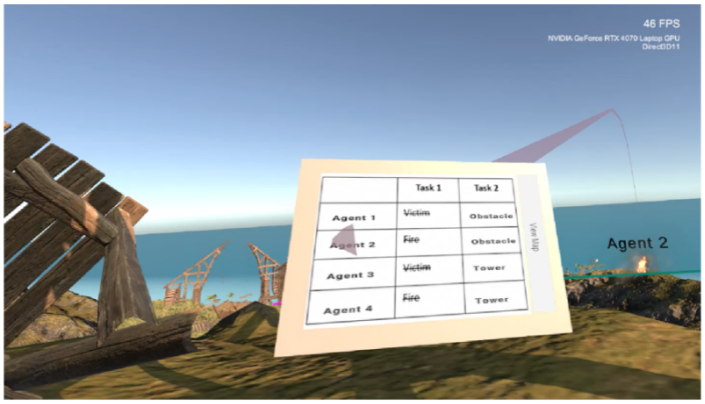
Independent variables. For this study, the single independent variable was the policy summarization presentation condition: with position overlay (called WithPO) or without (called NoPO).

Participants. We recruited 23 participants (12 males, 11 females) via university mailing lists. Eligibility included fluent English speakers over 18 years old with clear vision and no history of motion or VR sickness. The average age was 24.4 years (SD = 4.6).

Procedures. We began the study by demonstrating the user interface for both NoPO and WithPO conditions. To ensure data quality, the participant answered several attention-check questions after the demonstration. During the within-subject study, the participant completed two trials, each consisting of three maps (six maps in total). One trial used the NoPO condition, and the other used the WithPO condition. Both trials used the same three maps but with different agent locations and policy summarizations to ensure similar difficulty. Maps were selected randomly for each trial, and participants were randomly assigned to two groups (NoPO first or WithPO first) to counterbalance any ordering effects.

Dependent variables. For each map, we first asked the participant to orient themselves to the environment (e.g., identifying their location and the robots' locations). The participant then answered the following four questions via multiple choices, based on the provided policy summarization and static 2D blueprint of the environment.

(a) NoPO: Without Position Overlay



(b) WithPO: With Position Overlay

Figure 5.6: Examples of the user's first-person view through the VR headset, without or with position overlay.

- Q1: What is the next task agent x will complete?
- Q2: Can agents x and y complete task τ?
- Q3: Why can't agents x and y complete task τ?
- Q4: Is agent x or y physically closer to task τ?

where x, y, and τ were instantiated with the names of various agents and tasks for different maps. To track a participant's performance, we measured the time taken to orient themselves, the time taken to answer each question, the correctness of their answers, and their confidence ratings on a seven-point scale.

At the end of each trial, the participant was asked to complete an unweighted NASA TLX [59] on a seven-point scale to measure cognitive load. Additionally, the participant rated several statements about the policy summarization based on explanation goodness metrics (e.g., user satisfaction) adapted from [46]. The questions presented are as follows:

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

Hypotheses. Utilizing position overlay to support the presentation of policy summarization leads to:

- **H1:** reduced response time;
- **H2:** more correctly answered questions;
- **H3:** lower cognitive load;
- **H4:** higher summarization goodness ratings.

5.6.2 Study Results

Response time. Participants were able to orient themselves more quickly under the WithPO condition than with NoPO. A paired T-test ($\alpha = 0.05$) showed a statistically significant difference ($t(22)$=2.4, $p\leq$0.03, d=0.5). Moreover, as shown in Figure 5.7(a), there is a significant difference in the question response time between the WithPO and NoPO conditions: Q1 ($t(22)$=4.2, $p\leq$0.01, d=0.9), Q2 ($t(22)$=5.3, $p\leq$0.01, d=1.1), Q3 ($t(22)$=2.5, $p\leq$0.03, d=0.5), and Q4 ($t(22)$=2.4, $p\leq$0.03, d=0.5). *Thus, the data supports H1.*

Correct answers. Figure 5.7(b) shows the number of correct answers for each question type given unlimited time. A paired T-test ($\alpha = 0.05$) revealed no significant difference between the WithPO and NoPO conditions: Q1 ($t(22)$=-1.4, $p\leq$0.2, d=0.3), Q2 ($t(22)$=-1.6, $p\leq$0.2, d=0.3), Q3 ($t(22)$=-1.7, $p\leq$0.1, d=0.4), and Q4 ($t(22)$=2.1, $p\leq$0.06, d=0.4). *Thus, the data rejects H2.*

However, participants gave higher confidence ratings for WithPO: an average of 6.2 (SD=0.6) compared to NoPO: 5.6 (SD=0.7). A Wilcoxon signed-rank test ($\alpha = 0.05$) showed a significant difference (W=14.5, Z=-2.4, $p\leq$0.02, r=-0.4).

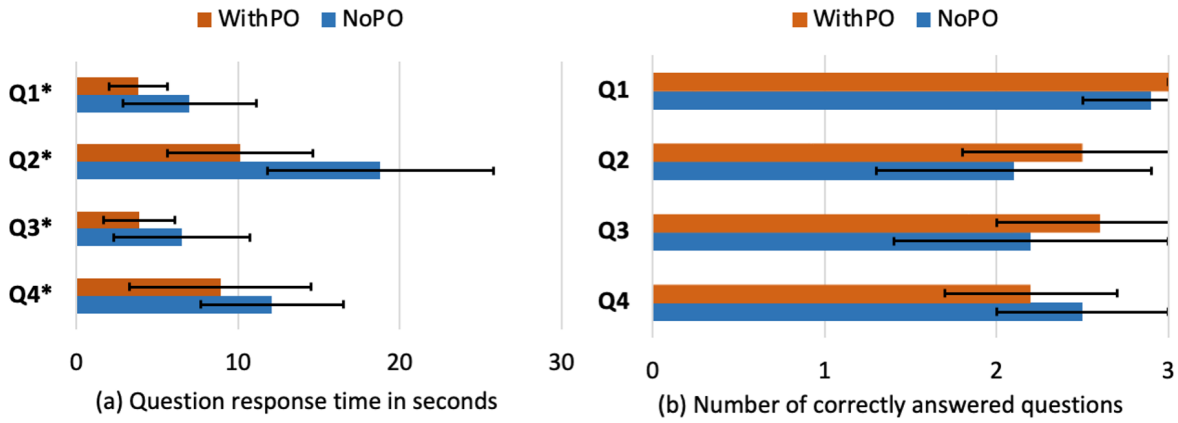


Figure 5.7: Mean and SD of question-answering performance (* indicates statistically significant difference).

Cognitive load. Participants reported an average cognitive load index of 17.8 (SD=4.8) when utilizing WithPO, compared to 22.8 (SD=7.1) with NoPO. The reduced cognitive load under the WithPO condition may be attributed to the ease of locating agents and tasks. A Wilcoxon signed-rank test ($\alpha = 0.05$) showed a significant difference (W=0.0, Z=2.6, $p\leq$0.01, r=0.4). *Thus, the data supports H3.*

Summarization goodness. Figure 5.8 shows the percentage of participant ratings on each summarization goodness statement. We found a significant difference using a Wilcoxon signed-rank test ($\alpha = 0.05$) in user understanding (W=0.0, Z=-2.3, $p\leq$0.03, r=-0.3), satisfaction (W=0.0, Z=-2.4, $p\leq$0.02, r=-0.3), detail (W=4.0, Z=-2.7, $p\leq$0.01, r=-0.4), completeness (W=4.0, Z=-2.5, $p\leq$0.02, r=-0.4), and actionability (W=0.0, Z=-2.8, $p\leq$0.01, r=-0.4) when using the WithPO condition. No significant difference was found in reliability (W=18.0, Z=-1.7, $p\leq$0.09, r=-0.3) and trust (W=14.5, Z=-1.6, $p\leq$0.11, r=-0.2). *Thus, the data partially supports H4.*

5.6.3 Discussion

The results show that participants responded faster in both orienting themselves and answering questions under the WithPO condition compared to the NoPO condition. This is likely due to the ease of locating agents and processing their surroundings with position overlay.

As expected, with unlimited time, participants could figure out the correct answers in both conditions. However, it took significantly longer under the NoPO condition, and participants were more confident in their responses under the WithPO condition due to reduced uncertainty about the agents' positions and environmental effects.

Since unlimited time is not always available in fast-paced, high-pressure search and rescue missions, we tracked the number of questions participants correctly answered in under five seconds. We found

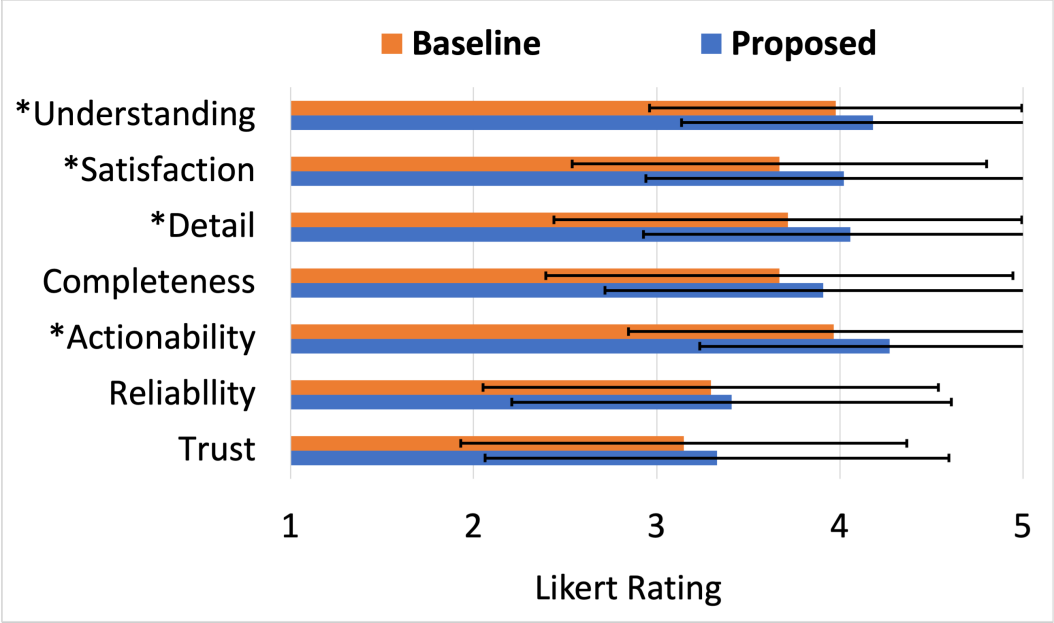


Figure 5.8: Distribution of participant ratings on summarization goodness metrics (* indicates statistically significant difference).

that participants answered significantly fewer questions correctly with NoPO compared to WithPO. However, more testing is needed, as participants were not originally given this time limit in the study, which could affect their behavior.

The majority of summarization goodness metrics were rated higher under the WithPO condition. However, because participants did not actually observe task completion, metrics like reliability and trust showed no statistically significant difference between WithPO and NoPO conditions.

In summary, the user study results suggest that the generated policy summarizations are effective on their own but are more effective when presented with position overlay.

5.7 Summary

This chapter presents a novel approach for summarizing decentralized MARL policies. Computational experiments on four MARL domains show that our approach is scalable and compatible with various decentralized MARL algorithms. Our user studies demonstrate the effectiveness of the generated policy summarizations, improving user performance. Additionally, we highlight the benefits of summary presentation with position overlay such as improved question-answering performance, reduced cognitive load, and enhanced user satisfaction.

Chapter 6

Policy Explanations for Decentralized MARL

6.1 Overview

In addition to the decentralized summarization method found in the previous chapter, we develop query-based explanations for decentralized MARL. We focus on three types of user queries: "When do [agents] do [task]?", "Why don't [agents] do [task] under [conditions]?", and "What do the agents do after [task]? inspired by those found in [21]. Drawing information from the generated Hasse diagram-based summarizations found in Chapter 5, we compute abstract states representing features such as agent cooperation and completed tasks. A minimal Boolean logic expression that covers all states matching the given user query is then determined, similar to the work found in Chapter 3. We convert the boolean expression into a natural language explanation using Large Language Models (LLMs) and present it to the user in response to their query.

Using two distinct MARL algorithms, we show the scalability of our proposed approach for all three query types across four MARL domains. Our method can be applied to large MARL domains with a significant number of agents and tasks, producing reasonable explanations quickly and effectively.

Additionally, we assess the effectiveness of the generated explanations through a user study, measuring user performance, completion time, and goodness metrics [46]. Results show that our proposed approach significantly improves user performance and increases subjective ratings without needing significantly more time than the more naive baseline.

6.2 Query-based Explanations

A policy summarization cannot provide a local view of agent behavior, limiting user understanding of specific actions. Thus, we also develop Hasse diagram-based methods (HDE) to generate explanations for three user queries:

- "When do [agents] do [task]?"
- "Why don't [agents] do [task] under [conditions]?"
- "What do the agents do after [task]?

Algorithm 9 HDE-When - Hasse diagram-based explanations for "when" queries

Input: user query "When do agents G_q do task T_q?", Hasse diagrams $\mathcal{D}$ from simulations, abstract features F

Output: Explanation E

1: $\mathcal{F} \leftarrow$ insert all relevant features for T_q from F
2: $U \leftarrow \{\}$
3: **for** $(j = 1; j < |simulations|; j{+}{+})$ **do**
4: **if** $\mathcal{D}_j$ contains task T_q completed by agents G_q **then**
5: **for** $v \in V_j$ **do**
6: **if** BFS(v,T_q) = False & BFS(T_q,v) = False **then**
7: $U[\mathcal{D}_j] \leftarrow f \in \mathcal{F}$ related to v
8: **for** $(j = 1; j < |simulations|; j{+}{+})$ **do**
9: **for** $v \in V_j$ **do**
10: **if** v contains task T_q completed by agents G_q **then**
11: $\mathbf{V} \leftarrow v$
12: **else**
13: $\mathcal{V} \leftarrow v$
14: $B_1 \leftarrow$ NodeToBoolean($\mathbf{V}$,$\mathcal{D}_j$,U)
15: $B_0 \leftarrow$ NodeToBoolean($\mathcal{V}$,$\mathcal{D}_j$,U)
16: $\phi \leftarrow$ Quine-McCluskey(ones = B_1,zeros = B_0)
17: Translate ϕ into explanation E via LLM
18: **return** E
19: **function** NODETOBOOLEAN(V,$\mathcal{D}_j$,U)
20: $B \leftarrow$ [ ]
21: **for** $v \in V$ **do**
22: $C \leftarrow$ feature predicate valuations of v with $\mathcal{D}_j$, $U[\mathcal{D}_j]$
23: **for** $f \in \mathcal{F}$ **do**
24: Insert $C(v, f)$ to B
return B

With these queries, our goal is to generate a set of conditions for, isolate missing conditions of, and generate a list of behaviors after task completion by one or multiple agents.

6.2.1 Explanations for When Queries

Algorithm 9 shows the steps for answering the query "When do [agents] do [task]?" The variables G_q and T_q represent the user-selected agents and task, respectively. We also receive a set of Hasse diagrams produced from Algorithm 8 and a set of abstract features F.

Using domain knowledge, the algorithm begins by isolating the relevant features $\mathcal{F}$ for task T_q from the total set of abstract features F. We follow the method described in [12], shown in Chapter 3, to isolate relevant features via domain knowledge. For our method, these abstract features contain information regarding the MARL domain selected by experts such as agent actions (i.e. agent 1 completes task A) and task completion (i.e. task A is completed) (line 1).

Example 8. For example, the user may ask "When do agents 2 and 4 do task C?" We will assume

that our total abstract features include agent actions (agent 2 completes task C) and task completions (task A complete) for all agents and tasks. From domain knowledge, we know that only agents 2 and 4 have the ability to complete task C and that completing task C is not a precondition for any other task. So, the relevant features are agent 2 completing task C, agent 4 completing task C, and the completion of any other task (A, B, D, E, F).

Since all the Hasse diagrams from Algorithm 8 are generated by simulations of the same underlying decentralized policies, which are also trained in the same environment, all the diagrams have the same underlying conditions for the execution of task T_q. So, we combine the information from all the diagrams together to generate a set of conditions for T_q (line 3).

If a diagram contains the selected task T_q, completed by the selected agents G_q, we generate a partial comparability graph [60] focused on the node containing task T_q. A compatibility graph indicates the nodes in a Hasse diagram that have a total ordering. For a full comparability graph, a search algorithm is run between all pairs of nodes to determine if a path exists between them in the Hasse diagram, indicating a known ordering. However, our interest is only in nodes (and their contained tasks) connected to the node containing task T_q. So, we run a breath-first search from the node containing task T_q to all other nodes and vice versa, generating a partial graph (lines 4-6).

If a node v is not connected to the node containing task T_q in the comparability graph, we connect any features in $\mathcal{F}$ related to v to diagram D_j in our dictionary tracking uncertain features (lines 2, 7). As an unconnected node v indicates an unknown ordering of that node in relation to the node containing T_q. So, it is uncertain if the task contained in v occurred previously and is required for the execution of task T_q.

Example 9. Figure 6.1 contains an example of a diagram and its resulting comparability graph. The given task (task C) is marked in blue. Connected tasks (certain ordering) are marked in gray and unconnected tasks (uncertain ordering) are marked in green. Due to node containing task B being unconnected, we insert the feature “task B complete" into our uncertainty dictionary.

We now isolate the features needed for task T_q to be completed by agents G_q. For each diagram D_j, we view each node v (lines 8-9). If a node v contains the completion of the selected task T_q by agents G_q, the node is inserted into the set of target nodes $\mathbf{V}$. Otherwise, it is inserted into $\mathcal{V}$, a set of non-target nodes (lines 9-12).

Example 10. In Figure 6.1, the nodes in blue (given task) are target nodes and all other nodes are non-target nodes based on our given query. We do this for all received diagrams.

Once our sets of nodes are generated, we convert each node into a boolean formula, such that each bit of the formula corresponds to a relevant feature contained in $\mathcal{F}$ (lines 14-15). This conversion is

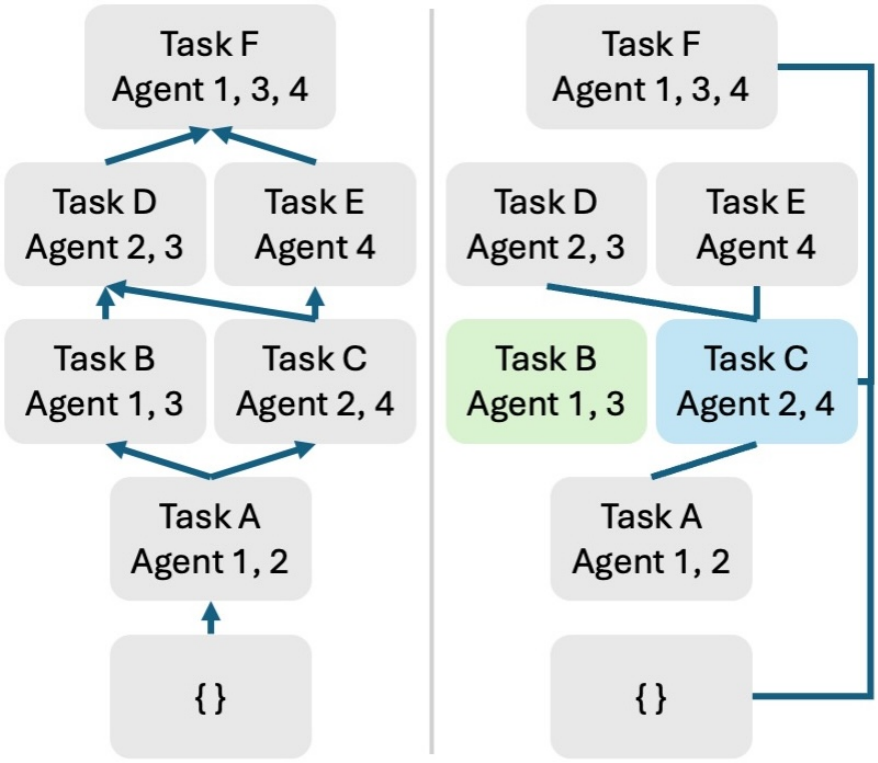


Figure 6.1: Hasse diagram and resulting partial comparability graph

adapted from the one found in [12], shown in Chapter 3. For each node v, if a feature f is true in the given Hasse diagram, the value of the corresponding bit is set to one, $C(v, f) = 1$. An abstract feature f is said to be true for a node $v \in D_j$, if there exists a path $p \in D_j$ satisfying f such that $v \in p$. For example, the feature "task A complete" is true if task A precedes node v in the Hasse diagram. If the value of the feature f is false, the value of the bit is set to zero, $C(v, f) = 0$.

Since each agent is only aware of its own internal state, we may not see all underlying conditions for task T_q in all diagrams. So, a feature f for a node v containing T_q in D_j is uncertain if f is contained in U in relation to D_j. The value of the bit representing f is set to one to ensure the feature is not discounted, $C(v, f) = 1$. For example, the bit for the feature "task B complete" is set to one for the node containing task T_q since the feature is contained in the uncertainty dictionary for the given diagram.

We then generate a minimized boolean formula representing the difference between the two sets (target vs. non-target) via the Quine-McCluskey algorithm [48] (line 16). However, due to uncertain features, we generate a super-set of underlying conditions. Yet, uncertain conditions are still tracked via U to ensure they are reported to the user, indicating which features of the formula may not be a part of the original set of conditions (lines 19-25).

Example 11. For example, the blue node in the diagram shown in Figure 6.1 is converted into the boolean formula [1111000] since agent 2 completes task C, agent 4 completes task C, task A is completed, task B is present in the uncertainty dictionary for this diagram, and tasks D, E, and F are not completed. We convert all nodes across all diagrams using this method and run the Quine-McCluskey algorithm to receive a minimized boolean formula covering all target states.

Finally, we use a large language model to convert the returned minimized boolean expression from Quine-McCluskey into a natural language explanation (line 17). Specifically, we use a template such that certain features are explained with the phrases "must" (true) and "must not" (false) based on their value. However, if a feature is an uncertain feature in any generated diagram (in dictionary U), it is explained with the phrases "may" (true) or "may not" (false). We present the resulting explanation to the user (line 18). The LLM prompts can be found below and resulting natural language explanations can be seen in Table 6.1.

"When" Query LLM Prompt. An example of the prompt given to the LLM to produce a query in the form "When do [agents] do [task]?" is given below:

Here is a sentence: "For agents 1 and 2 to do task A, agent 1 must complete task A, agent 2 must complete task A, and task B needs to be completed. Additionally, task C may need to be completed and agent 3 may need to complete task A. This sentence is generated from the following information: agents:[1,2], task: [complete task A], certain features:[agent 1 completes task A, agent 2 completes task A, task B completed], uncertain features:[task C completed, agent 3 completes task A]. Generate a sentence like the one above with the following information: agents:[2,4], task: [complete task C], certain features:[agent 2 completes task C, agent 4 completes task C, task A completed], uncertain features:[task B complete]. Remove any reference to the information.

6.2.2 Explanations for Why Not Queries

To answer the query "Why don't [agents] do [task] under [conditions]?", we alter Algorithm 9 to capture the minimal missing features for agents G_q to complete task T_q in a node with given conditions. First, we update lines 9-12, so that if node v contains the completion of task T_q by agents G_q it is inserted into the non-target set $\mathcal{V}$. Furthermore, the target set $\mathbf{V}$, will now only contain a boolean formula representing the conditions Γ_q given by the query. The new algorithm can be seen in Algorithm 10.

Example 12. For example, the user may ask "Why don't agents 2 and 4 do task C in the state where agents 2 and 4 are trying to complete task C and only task A is completed?" In Figure 6.1, the nodes in blue are non-target nodes and the target node has the following boolean formula [1110000] since agents 2 and 4 could complete task C and only task A is completed in the given conditions. The LLM prompts can be seen below and the resulting natural language explanation can be seen in Table 6.1.

Algorithm 10 HDE-WhyNot - Hasse diagram-based explanations for "why not" queries

Input: user query "Why don't do agents G_q do task T_q under conditions Γ_q?", Hasse diagrams $\mathcal{D}$ from simulations, abstract features F

Output: Explanation E

$\mathcal{F} \leftarrow$ insert all relevant features for T_q from F
$U \leftarrow \{\}$
for $(j = 1; j < |simulations|; j{+}{+})$ **do**
    if $\mathcal{D}_j$ contains task T_q completed by agents G_q **then**
        for $v \in V_j$ **do**
            if BFS(v,T_q) = False & BFS(T_q,v) = False **then**
                $U[\mathcal{D}_j] \leftarrow f \in \mathcal{F}$ related to v
for $(j = 1; j < |simulations|; j{+}{+})$ **do**
    for $v \in V_j$ **do**
        if v contains task T_q completed by agents G_q **then**
            $\mathcal{V} \leftarrow v$
$B_1 \leftarrow$ Convert Γ_q to boolean expression
$B_0 \leftarrow$ NodeToBoolean($\mathcal{V}$,$\mathcal{D}_j$,U)
$\phi \leftarrow$ Quine-Mccluskey(ones = B_1,zeros = B_0)
Translate ϕ into explanation E via LLM
return E
function NODETOBOOLEAN(V,$\mathcal{D}_j$,U)
$B \leftarrow$ [ ]
for $v \in V$ **do**
    $C \leftarrow$ feature predicate valuations of v with $\mathcal{D}_j$, $U[\mathcal{D}_j]$
    for $f \in \mathcal{F}$ **do**
        Insert $C(v, f)$ to B
return B

"Why Not" Query LLM Prompt. An example of the prompt given to the LLM to produce a query in the form "Why don't [agents] do [task] under [conditions]?" is given below:

Here is a sentence: "For agents 1 and 2 to do task A, agent 1 must complete task A, agent 2 must complete task A, and task B needs to be completed. Additionally, task C may need to be completed and agent 3 may need to complete task A. This sentence is generated from the following information: agents:[1,2], task: [complete task A], certain features:[agent 1 completes task A, agent 2 completes task A, task B completed], uncertain features:[task C completed, agent 3 completes task A]. Generate a sentence like the one above with the following information: agents:[2,4], task: [complete task C], certain features:[none], uncertain features:[task B complete]. Remove any reference to the information.

6.2.3 Explanations for What Queries

The query "What do the agents do after [task]?" can be answered using a simple set of steps, shown in Algorithm 11. After generating our set of Hasse diagrams using Algorithm 8 for each simulation, we isolate any nodes where task T_q is completed. For those nodes, we generate a list of children, counting the number of times each task appears in a node in that list. This gives us the frequency of the next

possible tasks that are certain. For each node containing T_q, we also generate a partial comparability graph focused on it. We then count the number of times each task occurs in an unconnected node. This gives us the frequency of possible next tasks that are uncertain. We then present the user with a list of certain and uncertain next tasks in order of frequency in a natural language template containing both certain and uncertain tasks: After the [task] is completed, [certain tasks] are completed. Additionally, [uncertain tasks] may be completed.

Algorithm 11 HDE-What - Hasse diagram-based explanations for "what" queries

Input: user query "What do agents do after task T_q?", Hasse diagrams $\mathcal{D}$ from simulations
Output: Explanation E

$\mathcal{C}, U = []$
for $(j = 1; j < |simulations|; j{+}{+})$ **do**
  for $v \in V_j$ **do**
    if v contains task T_q completed by agents G_q **then**
      $\mathcal{C} \leftarrow$ children of v
      for $v' \in V_j$ **do**
        if BFS(v,v') = False & BFS(v',v) = False **then**
          U.append(v')
Translate $\mathcal{C}$ and U into explanation E via templates
return E

Example 13. For example, the user may ask "What do agents do after task C is completed?" The resulting natural language explanation can be seen in Table 6.1.

Query	**Explanations Generated by HDE**	**Explanations Generated by Baseline**
When do agents 2 and 4 do task C?	For agents 2 and 4 to complete task C, agent 2 must complete task C, agent 4 must complete task C, and task A must be completed. Additionally, task B may need to be completed.	For agents 2 and 4 to complete task C, agent 2 must complete task C, agent 4 must complete task C, and task A must be completed.
Why don't agents 2 and 4 do task C in state where only task A is completed?	For agents 2 and 4 to complete task C, task B may need to be completed.	Agents can complete task C.
What do the agents do after task A is completed?	After task C is completed, tasks D and E are completed. Additionally, task B may be completed.	After task C is completed, tasks D and E are completed.

Table 6.1: Examples of query-based explanations

6.2.4 Properties

When Queries. The time complexity of HDE-When follows the exponential time complexity of Quine-McCluskey and is bounded by $\mathcal{O}(3^{N\cdot|\mathcal{F}|}/ln(N\cdot|\mathcal{F}|))$. It relies on the number of agents and the number of relevant features.

Why Not Queries. Since the Algorithm 10 is a modified version of Algorithm 9, it follows the same time complexity.

What Queries. The time complexity of Algorithm 11 is significantly smaller, bounded by $\mathcal{O}(4\mathcal{D}V^2(V+E))$ since the algorithm does not call Quine-McCluskey. It relies on the number of diagrams as well as the number of tasks (nodes) and the number of pre-order rules (edges) in each Hasse diagram.

6.3 Computational Experiments

		CTDE						DTDE					
		HDE-When			Baseline			HDE-When			Baseline		
Domain	$\|N\|$,$\|T\|$	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)
SR	2,3	6	1	0.02	15	0	0.02	7	1	0.03	16	0	0.03
	9,7	9	2	0.06	54	0	0.04	5	5	0.07	45	0	0.04
LBF	5,5	15	1	0.04	58	0	0.03	14	4	0.04	41	0	0.03
	9,9	13	11	0.1	104	0	0.06	-	-	-	-	-	-
RW	3,4	0	3	0.02	4	0	0.01	0	2	0.02	2	0	0.01
	4,19	0	153	0.29	267	0	0.07	-	-	-	-	-	-
PP	4,4	5	3	0.03	14	0	0.01	5	3	0.03	13	0	0.01
	7,6	8	3	0.04	20	0	0.01	8	3	0.05	19	0	0.03

Table 6.2: Results of computational experiments on HDE explanation method for “when" queries

		CTDE						DTDE					
		HDE-WhyNot			Baseline			HDE-WhyNot			Baseline		
Domain	$\|N\|$,$\|T\|$	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)	$\|\phi_{\not U}\|$	$\|\phi_U\|$	Time (s)
SR	2,3	1	0	0.04	2	0	0.07	1	0	0.05	2	0	0.08
	9,7	1	0	1	2	0	1.8	1	0	1.12	2	0	1.9
LBF	5,5	1	0	0.29	2	0	0.56	1	0	0.3	2	0	0.56
	9,9	1	0	12.07	2	0	27	-	-	-	-	-	-
RW	3,4	1	0	0.01	2	0	0.01	1	0	0.01	1	0	0.01
	4,19	1	0	461.09	1	0	449.65	-	-	-	-	-	-
PP	4,4	1	0	0.05	2	0	0.04	1	0	0.05	3	0	0.04
	7,6	1	0	1.01	2	0	0.94	1	0	1	1	0	0.95

Table 6.3: Results of computational experiments on HDE explanation method for “why not" queries

MARL domains. We evaluate the proposed approach through computational experiments on four benchmark MARL domains:

- *Search and Rescue (SR)* - A gridworld environment where agents cooperate to complete assigned

		CTDE						DTDE					
		HDE-What			Baseline			HDE-What			Baseline		
Domain	$\|N\|$,$\|T\|$	$\|\mathcal{C}\|$	$\|U\|$	Time (s)	$\|\mathcal{C}\|$	$\|U\|$	Time (s)	$\|\mathcal{C}\|$	$\|U\|$	Time (s)	$\|\mathcal{C}\|$	$\|U\|$	Time (s)
SR	2,3	2	1	0.01	2	0	0.01	1	1	0.01	1	0	0.01
	9,7	3	0	0.02	2	0	0.02	1	7	0.02	3	0	0.02
LBF	5,5	4	0	0.02	4	0	0.01	4	0	0.02	3	0	0.01
	9,9	8	0	0.02	6	0	0.02	-	-	-	-	-	-
RW	3,4	1	3	0.01	1	0	0.01	1	3	0.01	2	0	0.004
	4,19	9	11	0.04	10	0	0.01	-	-	-	-	-	-
PP	4,4	1	3	0.02	1	0	0.01	1	3	0.02	1	0	0.01
	7,6	1	5	0.02	2	0	0.01	2	3	0.02	2	0	0.01

Table 6.4: Resultsof computational experiments on HDE explanation method for “what" queries

search and rescue tasks.

- *Level-Based Foraging (LBF)* - A mixed cooperative-competitive game in a gridworld environment where agents collect food [11];
- *Multi-Robot Warehouse (RW)* - A gridworld environment where multiple agents cooperate to collect and deliver items [11];
- *Pressure Plate (PP)* - A gridworld environment where agents press switches to open doorways in a maze, enabling other agents to navigate to a goal [53].

To highlight decentralized execution, we set each environment so agents can observe only one neighboring grid cell in each cardinal direction for the first three domains and up to four grid cells in any direction in the Pressure Plate domain.

Baseline. We adapt a single-agent explanation method and suggestions from [26] to create an aggregation-based method as a possible baseline. After generating an abstract policy graph for each agent policy, we produce an explanation for each agent by selecting target and non-target states and generating a minimized boolean formula [21], adapting this method to contain only relevant features like our own. Then, following suggestions in [26] we combine the returned information from each agent through simple aggregation.

Setup. The experiments were run on a machine with a 2.1 GHz Intel CPU, 132 GB of memory, and Ubuntu 22.04 operating system. All models were trained until converging to the expected return or up to 400 million steps. We utilized two MARL algorithms, Shared Experience Actor-Critic (SEAC) [10] for the CTDE type and Independent Advantage Actor-Critic (IA2C) [11] for the DTDE type, to demonstrate the proposed approaches' agnosticism to decentralized MARL algorithms. Due to the lack of quality policies (policies did not converge) learned by the IA2C algorithm in the larger environments, LBF(9,9) and RW(4,19), results for these cases are absent.

Explanation results analysis. To generate explanations for our experiments, we execute each trained policy for 100 episodes to generate 100 Hasse diagrams, then we generate an explanation. Tables 6.2 and 6.3 show the total run time in seconds and feature amount (number of certain, uncertain features) for both CTDE and DTDE policies, respectively. Table 6.4 reports the run time and the number of both certain and uncertain subsequent tasks.

First, run times to generate all three types of queries are comparable across both our proposed method and the baseline. This suggests that there is no additional time needed to produce the extra uncertain features for "when" queries and "why not" queries or uncertain tasks for "what" queries, which can add to the user's understanding of the decentralized policy.

The baseline method produces larger abstract policy graphs compared to our proposed method's Hasse diagrams and naively aggregates information from multiple agents through the simple addition of generated single-agent explanations. So, we find that for "when queries" and "why not" queries significantly more features are produced. This increase in features can cause a significant cognitive burden on users as they must sort through, combine, and reduce features themselves to understand agent actions. Furthermore, as expected, there are no uncertain features produced by the baseline leaving the resulting explanations incomplete.

Regarding "what" queries our method can isolate more tasks compared to the baseline, as it considers both certain and uncertain tasks, giving the user a more complete set of subsequent tasks. Additionally, the combined information from multiple agents results in fewer certain tasks, as the combined information is able to limit what tasks are certain to happen and what tasks are uncertain. The naive aggregation of subsequent tasks often leads to a long list of subsequent tasks that may not be accurate. Thus, the HDE method is more accurate and complete. It allows users to better understand all possible options for subsequent agent actions.

In summary, the proposed explanation approach shows efficient run times and more effective explanations compared to the baseline. These explanations can be scaled for large environments with a significant number of agents and tasks for both CTDE and DTDE algorithms.

6.4 User Study

Additionally, we conducted a user study[6] to evaluate the effectiveness of generated policy explanations for three types of queries (when?, why not?, what?).

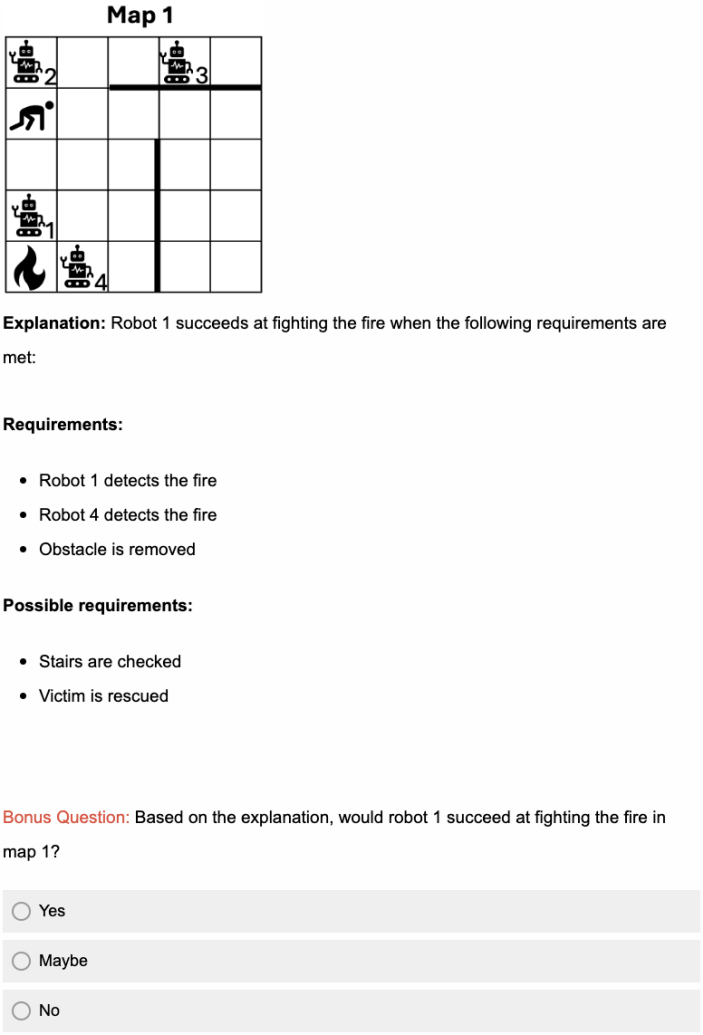


Figure 6.2: Question based on explanations for a "when" query.

6.4.1 Study Design

User interface. To evaluate the produced explanations, users were presented with a survey via Qualtrics. Users were first presented with a map of a search and rescue mission containing four agents and four tasks. They were then given an explanation containing the requirements for when a task occurs, the violated requirements causing a task to not occur, or what tasks could occur based on the given query (when?, why not?, what?) for the question. Participants were then asked to predict if the task would occur in the given map or what tasks could happen next. Figure 6.2 shows an example of the basic user interface for a question based on an explanation for a"when" query. Figures 6.3 and 6.4 show the user interface for a "why not" and "what" query, respectively. Our method is

[6]This study was approved by the UVA Institutional Review Board with IRB-SBS #7237.

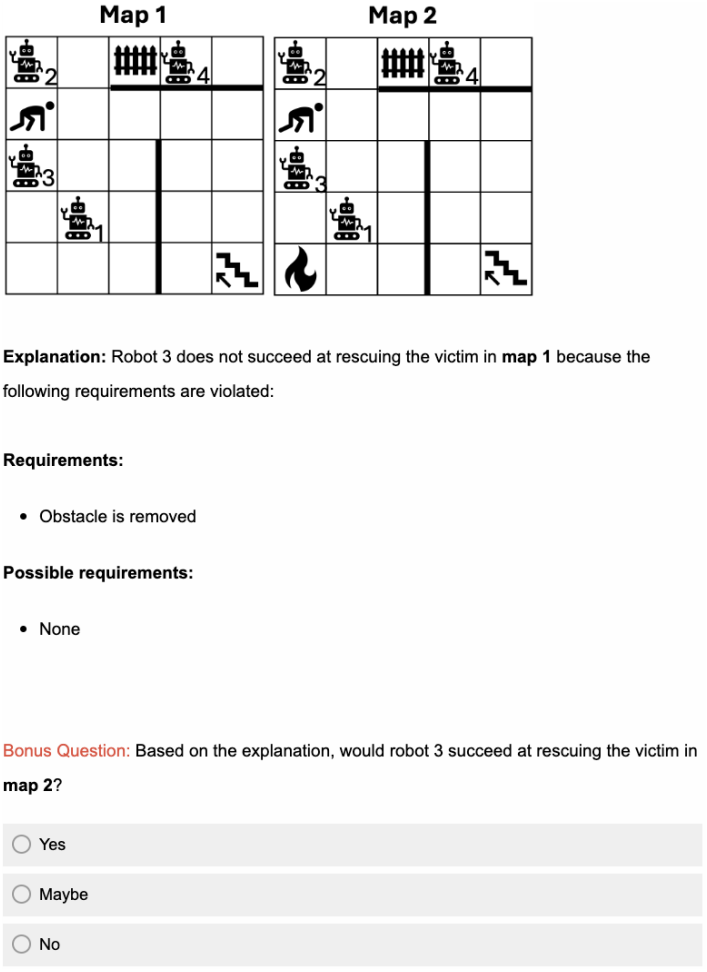


Figure 6.3: Question based on explanations for a “why not” query.

able to produce both certain features (Requirements) and uncertain features (Possible requirements) since our method is built specifically for decentralized policies. However, the baseline is only able to produce certain features (Requirements), so any uncertain features (Possible requirements) are listed as unknown. This is also true for certain and uncertain tasks in the “what" query.

Participants. We recruited 21 participants (14 males, 6 females, 1 other) via university mailing lists. Those eligible included fluent English speakers over 18 years old. The average age was 24 years (SD = 3.95). Participants were incentivized with bonus payments to answer questions correctly based on the provided explanations.

Baseline. Since there are no explanation methods specifically for decentralized MARL, we generate a baseline utilizing a common single agent method found in [21] combined with multi-agent suggestions from [26]. Overall, we generate an explanation for each agent given in the user query using an abstract policy graph from that agent’s policy as presented in [21], then we aggregate the generated features

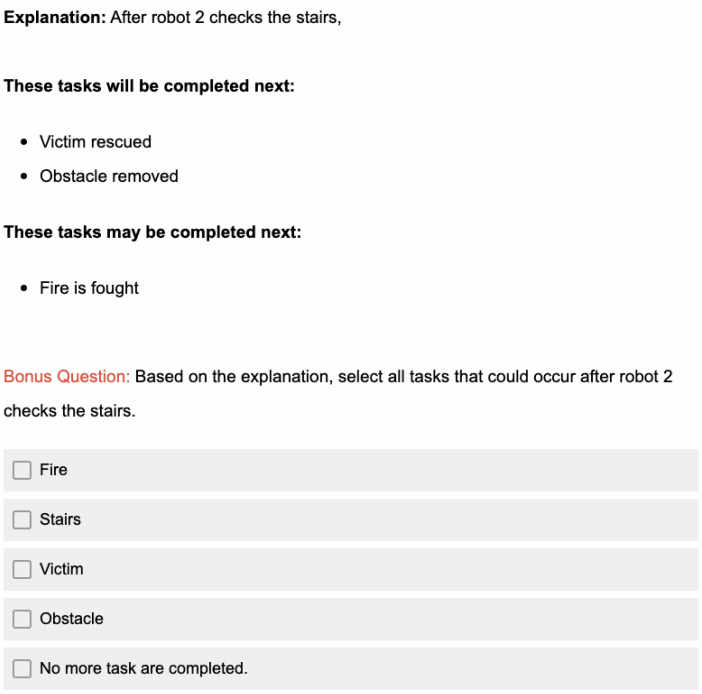


Figure 6.4: Question based on explanations for a "what" query.

(Requirements) together as suggested in [26].

By naively aggregating explanations from each agent, explanations do not provide information regarding agent cooperation on tasks or tasks completed by other agents outside of the given query. So, agents may miss task requirements regarding agent-task assignment or preconditions (i.e. fire must be fought before the victim is rescued) that could be provided by other agents. Furthermore, the produced single agent explanation may add additional information always observed by one agent that could be discounted by another.

Independent variables. For the explanation study, the single independent variable was explanation type: HDE or baseline.

Procedures. We began the study by demonstrating how to utilize the explanations. To ensure data quality, the participant answered several attention-check questions after the demonstration. During the within-subject summarization study, the participant completed two trials (one for each method), each consisting of two questions for each query type (12 questions in total). Participants were randomly assigned to two groups (aggregation first or Hasse diagram first) to counterbalance any ordering effects. All questions within the trials were also randomized. A demonstration was given, attention checks were injected, bonus payments were offered, and the time to complete the survey was tracked to ensure data quality.

Dependent variables. We counted the total number of prediction questions correctly answered by a

participant for each query type as a performance measure. The user's response time for each question was also recorded.

At the end of each trial, participants were instructed to rate the explanations on several goodness metrics on a 5-point Likert scale [46]. Participants were informed of the number of prediction questions they answered correctly before rating the explanations. The questions presented are as follows:

- The explanations help me *understand* how the robots complete the mission.
- The explanations are *satisfying*.
- 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 three hypotheses stated below:

- **H1:** HDE explanations lead to *better user performance* than the baseline explanations across all three query types.
- **H2:** HDE explanations have an *equal question response time* compared to the baseline explanations across all three query types.
- **H3:** HDE explanations lead to *higher ratings on summarization goodness metrics* than the baseline explanations across all three query types.

6.4.2 Study Results

Question-Answering Performance. As shown in Figure 6.5, users were able to answer more questions correctly using the HDE method compared to the baseline one for all three query types. A paired t-test ($\alpha = 0.05$) shows a statistically significant difference for all three queries: *when* ($t(20)=9.65$, $p \leq 0.01$, $d=2.16$), *why not* ($t(20)=13.23$, $p \leq 0.01$, $d=2.96$), and *what* ($t(20)=12.05$, $p \leq 0.01$, $d=2.69$). *Thus, the data supports H1.*

Response Time. Figure 6.6 shows that participants spend an equal amount of time responding to all three types of questions for both the HDE and baseline methods. A paired t-test ($\alpha = 0.05$) shows no statistically significant difference for all three queries: *when* ($t(20)=0.57$, $p \leq 0.58$, $d=0.13$),

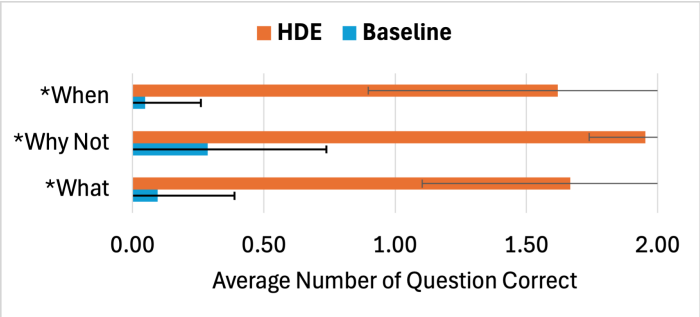


Figure 6.5: Mean and SD of participant performance about policy explanations ("*" indicates statistically significant difference).

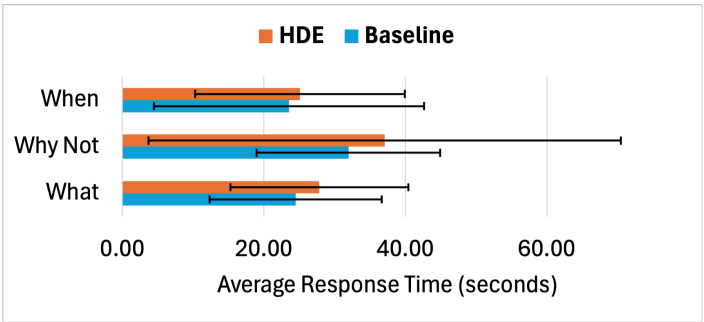


Figure 6.6: Mean and SD of participant response time about policy explanations ("*" indicates statistically significant difference).

why not (t(20)=0.7, $p \leq 0.49$, d=0.17), and *what* (t(20)=0.91, $p \leq 0.38$, d=0.21). We removed any outliers using the inter-quartile range method before the paired t-tests were performed. *Thus, the data supports H2.*

Summarization Goodness Rating. Participants find the proposed HDE method better than the baseline method in all reported explanation goodness metrics, shown in Figure 6.7. We used the Wilcoxon signed-rank test ($\alpha = 0.05$) to show the statistically significant difference in *understanding* (W=3.5, Z=-3.02, $p \leq 0.01$, r=-0.47), *satisfaction* (W=5.0, Z=-3.01, $p \leq 0.01$, r=-0.46), *detail* (W=4.5, Z=-3.02, $p \leq 0.01$, r=-0.47), *completeness* (W=0.0, Z=-3.21, $p \leq 0.01$, r=-0.49), *actionability* (W=4.0, Z=-2.53, $p \leq 0.02$, r=-0.39), *reliability* (W=3.0, Z=-2.78, $p \leq 0.01$, r=-0.43), and *trust* (W=0.0, Z=-2.68, $p \leq 0.01$, r=0.41). *Thus, the data supports H3.*

Discussion. In summary, the data supports all three hypotheses. This is most likely because our proposed HDE method presents the user not just with certain features (Requirements), but also uncertain features (Possible Requirements). So, participants can better predict if and what tasks will occur. Furthermore, there was no increase in response time when these extra uncertain features were added, indicating that users do not have increased difficulty understanding the uncertain features or

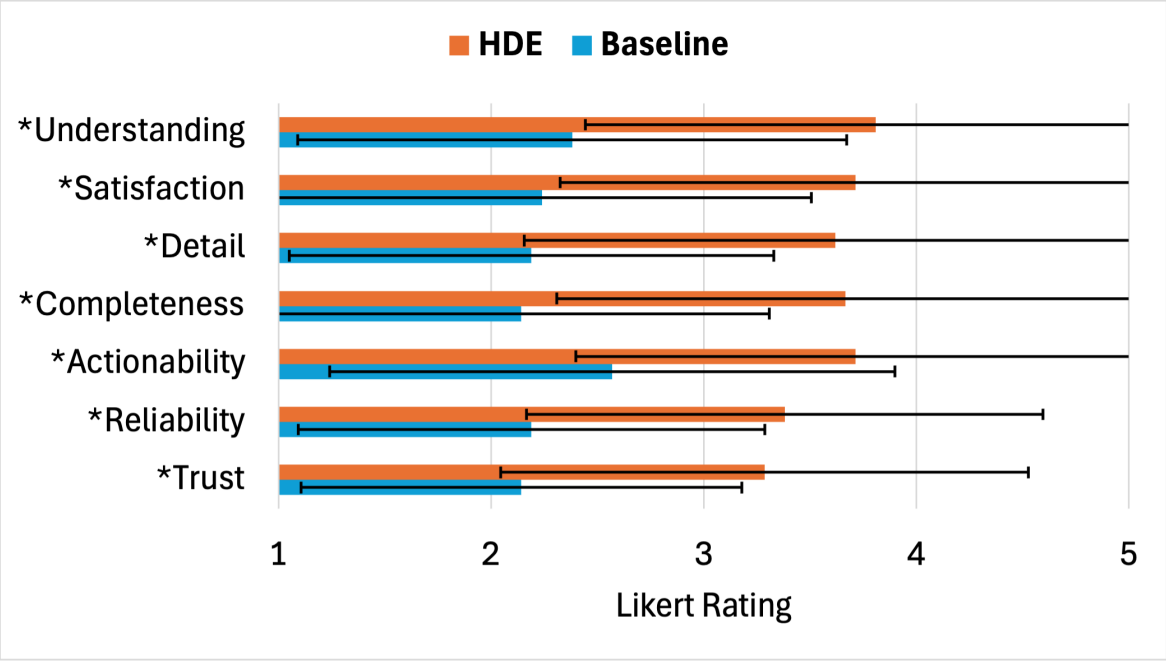


Figure 6.7: Mean and SD of participant ratings about policy explanations ("*" indicates statistically significant difference).

applying the often larger number of features quickly to predict outcomes. Finally, participants find the HDE explanations as better across all goodness metrics, indicating that the participants prefer the method focused on providing indications of decentralized information, even if it is more complex.

6.5 Summary

This chapter presents a novel approach to explain decentralized MARL policies. Computational experiments show that our method can be applied to large MARL domains with a significant number of agents and tasks, producing reasonable explanations quickly and effectively across two different decentralized algorithms and four MARL domains. Results of our user study show that our proposed approach significantly improves user performance and increases subjective ratings without needing significantly more time than the more naive baseline.

Chapter 7

Conclusion

In conclusion, this dissertation focuses on generating summarizations and explanations for multi-agent reinforcement learning. First, in Chapter 3 we provide policy summarizations and query-based explanations for centralized MARL. This allows users to understand both global agent behaviors and local agent actions for policies with joint agent states and actions. Not only are our methods effective, showing significant scalability across several MARL domains, but they also show improved user performance and increased goodness ratings.

Then, we present contrastive temporal explanations for centralized MARL in Chapter 4, allowing for more advanced queries that compare expected and actual agent behavior. The produced method is not only scalable and efficient for both feasible and infeasible queries but also improves user performance, understanding, and satisfaction.

Finally, we produce policy summarizations and query-based explanations for decentralized MARL in Chapters 5 and 6, respectively. These methods utilize Hasse-diagrams to represent the uncertainty and non-determinism of decentralized MARL policies to users in an accurate and understandable way. Furthermore, we demonstrate the effectiveness of augmented reality techniques like position overlay to represent decentralized summarizations. Much like their centralized counterparts, the produced methods provide effective and efficient ways to increase user performance and other goodness metrics.

In summary, this dissertation produces novel approaches to generate summarizations and explanations for both centralized and decentralized multi-agent reinforcement learning, improving user understanding, satisfaction, and performance.

7.1 Limitations and Future Directions

The work presented in this dissertation has several limitations. First, this work has only been tested on grid world examples utilizing simulations. So, we cannot comment on any real-world deployment with users. Future work includes applying the proposed methods to a broader range of MARL domains and environments. Specifically, domains and environments with a huge number of agents or real-world environments containing uncertainty and time constraints.

Second, more advanced queries may be needed to fully understand agent decisions and fill user

needs. For example, users may provide abstract references to agents or objects (Why do those agents not do that task?), inquire about broader categories of agents or tasks (e.g., Why do those agents perform any task except task y?), ask the agent to explain any uncertainty in its policy (Why might the agent try to do task y, but not succeed?), or consider subjective measures of the agent's decision (Why was the decision fair?). Future work would explore the possible development of these explanations as our work currently provides only the most basic summarizations and explanations.

Moreover, our system does not provide subjective explanations, which consider the perspectives of both the user and other agents to deliver the most relevant and accurate information. Future work should gather insights from agents with heterogeneous capabilities and reward systems while filtering out information the user is likely already familiar with to produce these more advanced explanations.

Finally, all the presented methods utilize abstract features to improve human understanding of explanations. However, these features are manually selected by domain experts which could produce an incomplete, biased, poorly defined, or ambiguous set of features. To improve this, we need to develop methods for automatically selecting a minimal, yet complete and human-understandable set of abstract features, while minimizing bias and uncertainty.

7.2 Social and Technological Impact

However, the proposed methods have the potential for significant technological advancement to ensure safety and satisfaction with MARL systems in complex situations. Furthermore, these summarizations and explanations offer decision support for human-agent cooperation, allowing users to make more informed decisions, faster in safety-critical situations while deterring the potential misuse of the system by mistake. Finally, this work can be applied to many different MARL fields as the produced methods are generalizable and not tied to any one MARL algorithm or policy type. This work provides a significant base of summarization and explanation methods that can be applied to fields such as search and rescue, transportation, health care, and manufacturing, allowing our work to significantly impact the lives of everyday users.

Bibliography

[1] A. Dafoe, E. Hughes, Y. Bachrach, *et al.*, "Open problems in cooperative ai," *arXiv preprint arXiv:2012.08630*, 2020.

[2] B. R. Kiran, I. Sobh, V. Talpaert, *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[3] S. Kraus, A. Azaria, J. Fiosina, *et al.*, "Ai for explaining decisions in multi-agent environments," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 534–13 538.

[4] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable automated planning & decision making.," in *IJCAI*, 2020, pp. 4803–4811.

[5] L. Wells and T. Bednarz, "Explainable ai and reinforcement learning—a systematic review of current approaches and trends," *Frontiers in artificial intelligence*, vol. 4, p. 48, 2021.

[6] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106 685, 2021.

[7] E. Puiutta and E. Veith, "Explainable reinforcement learning: A survey," in *International cross-domain conference for machine learning and knowledge extraction*, Springer, 2020, pp. 77–95.

[8] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati, "Plan explanations as model reconciliation: Moving beyond explanation as soliloquy," in *IJCAI*, 2017.

[9] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2023. [Online]. Available: https://www.marl-book.com.

[10] F. Christianos, L. Schäfer, and S. V. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," in *Thirty-fourth Conference on Neural Information Processing Systems*, Curran Associates Inc, 2020, pp. 10 707–10 717.

[11] G. Papoudakis, F. Christianos, L. Schäfer, and S. 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.

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

[13] K. Boggess, S. Kraus, and L. Feng, "Explainable multi-agent reinforcement learning for temporal queries," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[14] S. Chen, K. Boggess, and L. Feng, "Towards transparent robotic planning via contrastive explanations," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 6593–6598.

[15] S. Chen, K. Boggess, D. Parker, and L. Feng, "Multi-objective controller synthesis with uncertain human preferences," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2022, pp. 170–180.

[16] S. Kraus, K. Boggess, R. Kim, B. H. Choi, and L. Feng, "Towards computational foreseeability," in *2025 AAAI conference on artificial intelligence (AAAI)*, 2025.

[17] K. Boggess, "Explanations for multi-agent reinforcement learning," in *2025 AAAI conference on artificial intelligence (AAAI)*, 2025.

[18] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable reinforcement learning: A survey and comparative review," *ACM Computing Surveys*, 2023.

[19] N. Topin and M. Veloso, "Generation of policy-level explanations for reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2514–2521.

[20] D. Amir and O. Amir, "Highlights: Summarizing agent behavior to people," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 1168–1176.

[21] B. Hayes and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2017, pp. 303–312.

[22] S. Milani, Z. Zhang, N. Topin, *et al.*, "Maviper: Learning decision tree policies for interpretable multi-agent reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2022, pp. 251–266.

[23] D. Kazhdan, Z. Shams, and P. Liò, "Marleme: A multi-agent reinforcement learning model extraction library," in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–8.

[24] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Collective explainable ai: Explaining cooperative strategies and agent contribution in multiagent reinforcement learning with shapley values," *IEEE Computational Intelligence Magazine*, vol. 17, no. 1, pp. 59–71, 2022.

[25] J. Kottinger, S. Almagor, and M. Lahijanian, "Maps-x: Explainable multi-robot motion planning via segmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 7994–8000.

[26] Y. Mualla, I. Tchappi, T. Kampik, *et al.*, "The quest of parsimonious xai: A human-agent architecture for explanation formulation," *Artificial intelligence*, vol. 302, p. 103 573, 2022.

[27] N. Topin, S. Milani, F. Fang, and M. Veloso, "Iterative bounding mdps: Learning interpretable policies via non-interpretable methods," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 9923–9931.

[28] M. Landajuela, B. K. Petersen, S. Kim, *et al.*, "Discovering symbolic policies with deep reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2021, pp. 5979–5989.

[29] S. Sreedharan, U. Soni, M. Verma, S. Srivastava, and S. Kambhampati, "Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=o-1v9hdSult.

[30] M. L. Olson, R. Khanna, L. Neal, F. Li, and W.-K. Wong, "Counterfactual state explanations for reinforcement learning agents via generative deep learning," *Artificial Intelligence*, vol. 295, p. 103 455, 2021.

[31] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 2493–2500.

[32] A. Atrey, K. Clary, and D. Jensen, "Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning," in *International Conference on Learning Representations*, 2019.

[33] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in *IJCAI/ECAI Workshop on explainable artificial intelligence*, 2019.

[34] M. H. Danesh, A. Koul, A. Fern, and S. Khorram, "Re-understanding finite-state representations of recurrent policy networks," in *International Conference on Machine Learning*, PMLR, 2021, pp. 2388–2397.

[35] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[36] Z. Lin, K.-H. Lam, and A. Fern, "Contrastive explanations for reinforcement learning via embedded self predictions," in *International Conference on Learning Representations*, 2021.

[37] M. Finkelstein, L. Liu, Y. Kolumbus, D. C. Parkes, J. Rosenschein, S. Keren, *et al.*, "Explainable reinforcement learning via model transforms," in *Advances in Neural Information Processing Systems*, 2022.

[38] A. Hudon, T. Demazure, A. Karran, P.-M. Léger, and S. Sénécal, "Explainable artificial intelligence (xai): How the visualization of ai predictions affects user cognitive load and confidence," in *Information Systems and Neuroscience: NeuroIS Retreat 2021*, Springer, 2021, pp. 237–246.

[39] J. C. Kim, T. H. Laine, and C. Åhlund, "Multimodal interaction systems based on internet of things and augmented reality: A systematic literature review," *Applied Sciences*, vol. 11, no. 4, p. 1738, 2021.

[40] A. J. Karran, T. Demazure, A. Hudon, S. Senecal, and P.-M. Léger, "Designing for confidence: The impact of visualizing artificial intelligence decisions," *Frontiers in neuroscience*, vol. 16, p. 883 385, 2022.

[41] A. R. Dennis and T. A. Carte, "Using geographical information systems for decision making: Extending cognitive fit theory to map-based presentations," *Information systems research*, vol. 9, no. 2, pp. 194–203, 1998.

[42] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–33.

[43] X. Xu, A. Yu, T. R. Jonker, *et al.*, "Xair: A framework of explainable ai in augmented reality," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–30.

[44] M. B. Luebbers, A. Tabrez, and B. Hayes, "Augmented reality-based explainable ai strategies for establishing appropriate reliance and trust in human-robot teaming," in *5th International Workshop on Virtual, Augmented, and Mixed Reality for HRI*, 2022.

[45] X. Wu, H. Zhao, Y. Zhu, *et al.*, "Usable xai: 10 strategies towards exploiting explainability in the llm era," *arXiv preprint arXiv:2403.08946*, 2024.

[46] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable ai: Challenges and prospects," *arXiv preprint arXiv:1812.04608*, 2018.

[47] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[48] W. V. Quine, "The problem of simplifying truth functions," *The American mathematical monthly*, vol. 59, no. 8, pp. 521–531, 1952.

[49] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic model checking: Advances and applications," *Formal System Verification: State-of the-Art and Future Trends*, pp. 73–121, 2018.

[50] A. Aziz, V. Singhal, F. Balarin, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "It usually works: The temporal logic of stochastic systems," in *International Conference on Computer Aided Verification*, Springer, 1995, pp. 155–165.

[51] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[52] A. K. Chandra and G. Markowsky, "On the number of prime implicants," *Discrete Mathematics*, vol. 24, no. 1, pp. 7–11, 1978.

[53] T. McInroe and F. Christianos, *Pressureplate*, https://github.com/uoe-agents/pressureplate, Accessed: 2022-11-22, 2022.

[54] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*, Springer, 2011, pp. 585–591.

[55] S. K. Sarkar, "A textbook of discrete mathematics," in 9th. S. Chand Publishing, 2017, ch. 9.4 "Hasse Diagram", pp. 339–341.

[56] A. V. Aho, M. R. Garey, and J. D. Ullman, "The transitive reduction of a directed graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 131–137, 1972.

[57] K. Simon, "An improved algorithm for transitive closure on acyclic digraphs," *Theoretical Computer Science*, vol. 58, no. 1-3, pp. 325–346, 1988.

[58] J. McCalmon, T. Le, S. Alqahtani, and D. Lee, "Caps: Comprehensible abstract policy summaries for explaining reinforcement learning agents," in *nt'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

[59] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, Elsevier, 1988, pp. 139–183.

[60] D. Kelly, "Comparability graphs," *Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Its Applications*, pp. 3–40, 1985.