

# Explaining Decentralized Multi-Agent Reinforcement Learning Policies

Kayla Boggess<sup>1</sup>, Sarit Kraus<sup>2</sup>, Lu Feng<sup>1</sup>

<sup>1</sup>University of Virginia

<sup>2</sup>Bar-Ilan University

{kjb5we, lu.feng}@virginia.edu, sarit@cs.biu.ac.il

## Abstract

Multi-Agent Reinforcement Learning (MARL) has gained significant interest in recent years, enabling sequential decision-making across multiple agents in various domains. However, most existing explanation methods focus on centralized MARL, failing to address the uncertainty and non-determinism inherent in decentralized settings. We propose methods to generate policy summarizations that capture task ordering and agent cooperation in decentralized MARL policies, along with query-based explanations for “When,” “Why Not,” and “What” types of user queries about specific agent behaviors. We evaluate our approach across four MARL domains and two decentralized MARL algorithms, demonstrating its generalizability and computational efficiency. User studies show that our summarizations and explanations significantly improve user question-answering performance and enhance subjective ratings on metrics such as understanding and satisfaction.

## 1 Introduction

Multi-Agent Reinforcement Learning (MARL) has gained significant interest in recent years, enabling multi-agent sequential decision-making across various domains such as autonomous driving (Dinneweth et al. 2022) and multi-robot warehousing (Krnjaic et al. 2022). Recent works have explored generating explanations for MARL policies to enhance system transparency, improve user understanding, and foster human-agent collaboration (Boggess, Kraus, and Feng 2022, 2023). However, these prior efforts are primarily limited to *centralized* MARL frameworks, where joint policies are learned and executed with full observability. Such methods cannot adequately address the uncertainty, non-determinism, and limited observability inherent in *decentralized* MARL settings, which are common in real-world applications with communication or scalability constraints.

This work addresses this gap by introducing methods for generating policy summarizations and query-based explanations for decentralized MARL policies. Our approach is the first to summarize and explain agent coordination and task ordering under decentralized execution, enabling users to interpret multi-agent behavior even when individual agents act independently and only have local observations.

For example, consider a search and rescue mission where multiple cooperative robots follow a decentralized MARL policy. A human operator in the field receives decision-making support via an explainer that provides high-level policy summaries and answers user queries using real-time trajectory data. The summaries help the operator understand general robot behaviors—task completion, agent cooperation, task order—while the query-based explanations answer specific questions, such as: “When do [agents] complete [task]?”, “Why don’t [agents] complete [task] under [conditions]?”, or “What do the agents do after [task]?”. With this information, the operator can make informed decisions, such as prioritizing urgent tasks or allocating resources more effectively.

A key challenge in supporting such explanations is representing the uncertain, asynchronous execution of decentralized policies. Each agent’s policy governs only its local behavior, possibly unaware of others’ actions, making it difficult to infer global task order or cooperation from raw trajectories alone.

To tackle this, we develop a novel algorithm that constructs *Hasse diagram*-based summarizations from trajectories generated under decentralized execution. Each diagram is a directed acyclic graph where nodes represent tasks (annotated with the agents that completed them), and edges encode partial-order constraints over task completion times. The resulting diagrams compactly capture both coordination and uncertainty: branching edges represent nondeterminism in task order, while nodes annotated with multiple agents indicate cooperation on shared tasks.

Building on this, we develop query-based explanation methods for three types of user queries: “When?”, “Why not?”, and “What?”. Given a set of Hasse diagrams, we derive abstract states that encode key features such as task completions and agent involvement. To capture uncertainty, we introduce an *uncertainty dictionary* derived from partial comparability graphs that summarize unordered task dependencies across episodes. We then apply the Quine-McCluskey algorithm (Quine 1952) to extract minimal Boolean formulas, which are translated into natural language explanations using structured templates, with uncertain features explicitly expressed using “may” conditions.

We evaluate our method’s generalizability and computational efficiency across four benchmark MARL domains,

scaling to settings with up to 19 tasks and 9 agents. To demonstrate the algorithm-agnostic nature of our method, we apply it to two distinct MARL algorithms that both yield decentralized policies but differ in their training paradigms: centralized training versus decentralized training.

Finally, we assess the effectiveness of our summarizations and explanations via two user studies measuring objective task performance and subjective ratings. Results show that our approach significantly improves user question-answering accuracy and boosts subjective ratings such as understanding and satisfaction.

Together, these contributions bridge the gap between opaque decentralized MARL policies and interpretable, human-centered explanations, enabling effective human-agent collaboration in multi-agent environments.

## 2 Related Work

**Multi-Agent Reinforcement Learning.** MARL algorithms are commonly categorized by their training and execution paradigms. *Centralized training and centralized execution* (CTCE) methods train a single policy using full observability of the environment and agent states (Albrecht, Christianos, and Schäfer 2023). *Centralized training with decentralized execution* (CTDE) methods, such as SEAC (Christianos, Schäfer, and Albrecht 2020), use global information during training but deploy policies that operate on local observations at execution time. *Decentralized training and decentralized execution* (DTDE) methods, including independent learning approaches (Papoudakis et al. 2021), train each agent’s policy independently, treating other agents as part of the environment. This work focuses on post-hoc summarization and explanation of both CTDE and DTDE policies in cooperative settings.

**Policy Summarization.** Explainable RL (XRL) has received increasing attention, as surveyed in (Milani et al. 2023; Wells and Bednarz 2021), though most prior work targets single-agent settings. For instance, (Topin and Veloso 2019) introduces *abstract policy graphs*, which represent agent behavior as Markov chains over abstract states, and (McCalmon et al. 2022) improves their comprehensibility. (Amir and Amir 2018) visualizes agent behavior via representative trajectory videos. In the multi-agent domain, (Milani et al. 2022) uses intrinsically interpretable decision trees, while (Boggess, Kraus, and Feng 2022) abstracts centralized MARL policies into macro-actions over joint state-action trajectories to identify common agent behaviors. However, all of these methods assume a single input policy, whether from a single agent or a centralized multi-agent controller. In contrast, we consider decentralized settings where each agent has its own policy and actions may require inter-agent coordination. We develop a scalable method to summarize such decentralized executions using compact, structured representations.

**Query-Based Explanations.** Post-hoc explanations in single-agent RL often rely on abstract state representations (Hayes and Shah 2017; Sreedharan et al. 2022), saliency maps (Atrey, Clary, and Jensen 2019), causal models (Madumal et al. 2020), or reward decompositions (Juoza-

paitis et al. 2019). Several works have extended these ideas to multi-agent systems, but typically assume centralized control or non-cooperative agents. For example, (Boggess, Kraus, and Feng 2022) applies abstract policy graphs to centralized MARL, (Heuillet, Couthouis, and Díaz-Rodríguez 2022; Mahjoub et al. 2024; Chen et al. 2025) compute agent contributions to joint policies, and (Kottinger, Almagor, and Lahijanian 2021) visualizes action assignments in joint plans. (Mualla et al. 2022) proposes a framework for generating parsimonious explanations for teams of BDI agents. However, these methods do not support inter-agent cooperation under decentralized policies and often aggregate agent behavior in a naive or disjointed manner.

*To our knowledge, this is the first work to generate both policy summarizations and query-based explanations for decentralized MARL policies.*

## 3 Policy Summarization

**Decentralized MARL Policies.** Consider  $N$  agents, each with a decentralized MARL policy  $\pi^i : s^i \rightarrow \Delta(a^i)$  mapping local state  $s^i$  to a distribution over actions  $a^i$ . Agents act asynchronously due to decentralized execution without a global clock. Joint tasks are assumed to be completed simultaneously, with each agent observing only its own contribution and reward. Executing these policies yields trajectories  $\{\omega^i\}_{i=1}^N$ , where  $\omega^i = s_0^i, a_0^i, r_0^i, s_1^i, \dots$  records transitions observed by agent  $i$ . A task sequence  $\text{trace}(\omega^i) = \tau_1^i, \tau_2^i, \dots$  can be extracted from each trajectory  $\omega^i$ , where a completed task  $\tau$  is inferred from reward signals and state transitions.

**Problem Statement.** Given decentralized MARL policies and their execution trajectories, how can we generate a compact, interpretable summary that captures both individual and joint agent behaviors? We seek a representation that ensures *correctness*, meaning each agent’s behavior in the summary aligns with its actual task sequence, and *completeness*, meaning each agent’s full task sequence is captured in at least one path in the summary.

**Hasse Diagram Summarization.** We propose to summarize decentralized agent behavior using a Hasse diagram  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ , a directed acyclic graph that represents a partial order over task completions (Sarkar 2017). Each vertex denotes a set of tasks completed simultaneously and the agents that perform them. Edges encode precedence:  $v \prec v'$  indicates that tasks in  $v$  must precede those in  $v'$ . A path  $\rho = v_0 \rightarrow v_1 \rightarrow \dots$  through  $\mathcal{D}$  defines a possible task ordering. The projection of  $\rho$  onto agent  $i$ , denoted  $\rho^i$ , retains only tasks performed by agent  $i$ . We say  $\rho^i$  conforms to the agent’s trajectory  $\text{trace}(\omega^i)$  if  $\rho^i \sqsubseteq \text{trace}(\omega^i)$ —i.e., the task order is preserved.

Formally, the diagram  $\mathcal{D}$  is *correct* if, for all paths  $\rho$  and agents  $i$ , either  $\rho^i = \emptyset$  or  $\rho^i \sqsubseteq \text{trace}(\omega^i)$ . It is *complete* if, for every agent  $i$ , there exists a path  $\rho$  such that  $\rho^i = \text{trace}(\omega^i)$ .

We present Algorithm 1, which constructs a correct and complete Hasse diagram  $\mathcal{D}$  from a single episode of decentralized execution trajectories  $\{\omega^i\}_{i=1}^N$ , agnostic to how the policies are trained (e.g., CTDE, DTDE). The algorithm iterates through task sequences, creates nodes for new tasks,

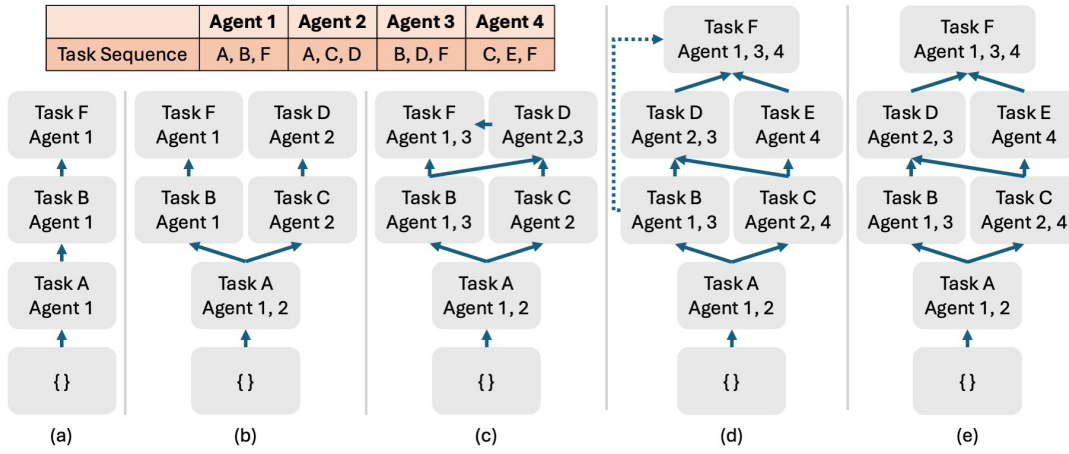


Figure 1: Example of Algorithm 1 constructing a Hasse diagram incrementally: steps (a)–(d) incorporate each agent’s task sequence, and step (e) applies transitive reduction.

---

**Algorithm 1** Hasse Diagram Summarization (HDS)

---

**Input:** Agent trajectories  $\{\omega^i\}_{i=1}^N$   
**Output:** Hasse Diagram  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$

- 1: Initialize:  $v_0 \leftarrow \emptyset$ ;  $\mathcal{V} \leftarrow \{v_0\}$ ;  $\mathcal{E} \leftarrow \emptyset$
- 2: **for** each agent  $i = 1$  to  $N$  **do**
- 3:    $T^i \leftarrow \text{trace}(\omega^i)$
- 4:   **for** each task index  $k = 1$  to  $|T^i|$  **do**
- 5:      $\tau \leftarrow T_k^i$
- 6:     **if**  $\tau$  exists in some vertex  $v \in \mathcal{V}$  **then**
- 7:       Add agent  $i$  to  $v[\tau]$
- 8:     **else**
- 9:       Create new vertex  $v'$  with  $v'[\tau] = \{i\}$ ;  $\mathcal{V} \leftarrow \mathcal{V} \cup \{v'\}$
- 10:     **if**  $k = 1$  **then**
- 11:       Add edge  $(v_0 \rightarrow v)$  or  $(v_0 \rightarrow v')$  to  $\mathcal{E}$
- 12:     **else**
- 13:       Let  $\tau_{\text{prev}} \leftarrow T_{k-1}^i$
- 14:       Find vertex  $\bar{v} \in \mathcal{V}$  containing  $\tau_{\text{prev}}$
- 15:       Add edge  $(\bar{v} \rightarrow v)$  or  $(\bar{v} \rightarrow v')$  to  $\mathcal{E}$  if not present
- 16: **for** each edge  $(u \rightarrow v) \in \mathcal{E}$  **do**
- 17:   **if** a path from  $u$  to  $v$  exists excluding edge  $(u \rightarrow v)$  **then**
- 18:     Remove edge  $(u \rightarrow v)$  from  $\mathcal{E}$
- 19: **return**  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$

---

identifies agents for shared tasks, and inserts edges to maintain local task order. Finally, a transitive reduction is applied to eliminate redundant edges.

Figure 1 shows an example of applying Algorithm 1, illustrating the incremental construction of a Hasse diagram summarizing agents’ behavior.

**Complexity and Guarantee.** The worst-case time complexity of Algorithm 1 is  $\mathcal{O}(N \cdot |T|^2 + |T|^4)$ , where  $N$  is the number of agents and  $|T|$  is the number of tasks. The  $\mathcal{O}(N \cdot |T|^2)$  term covers task-wise updates across  $N$  trajectories, and the  $\mathcal{O}(|T|^4)$  term arises from transitive reduction over a graph with up to  $|T|^2$  edges.

**Theorem 1.** *Given a set of agent trajectories  $\{\omega^i\}_{i=1}^N$  produced by executing decentralized MARL policies  $\{\pi^i\}_{i=1}^N$  in a single episode, the Hasse diagram  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$  constructed*

*by Algorithm 1 is both a correct and complete policy summarization. (Proof provided in Appendix A.)*

**Practical Considerations.** In large environments, users are often interested in only a subset of agents or tasks (e.g., nearby robots in a search and rescue scenario). Our method supports selective summarization by restricting input to relevant agent trajectories and applying task filters during sequence extraction.

Because decentralized execution is stochastic, different episodes may yield different Hasse diagrams. To summarize observed behaviors, we rely on actual trajectories; to capture potential future behaviors, we can simulate multiple episodes and report a representative diagram, such as the most frequent one.

## 4 Query-Based Explanations

While Hasse diagrams summarize global behavior, they do not explain local decisions—such as *when* agents choose to perform a task, *why* they fail to do so under certain conditions, or *what* they do next. To address this gap, we develop methods that generate language-based explanations in response to user queries. Our approach builds on prior work in query-based explanations for single-agent (Hayes and Shah 2017) and centralized multi-agent settings (Boggess, Kraus, and Feng 2022), but introduces new techniques to address the uncertainty and partial observability inherent in decentralized execution.

### 4.1 Answering “When” Queries

We consider queries of the form: “When do agents  $\mathcal{G}_q$  perform task  $\tau_q$ ?”, aiming to identify the necessary conditions under which  $\tau_q$  is completed by agent group  $\mathcal{G}_q$  across multiple simulated executions. Algorithm 2 outlines our method for generating language-based explanations for such queries.

We begin by extracting a subset of features  $\mathcal{F}_q \subseteq \mathcal{F}$  that are relevant to the query task, using domain knowledge. For example, for the query “When do agents 2 and 4 do task C?”, relevant features may include boolean predicates indicating

---

**Algorithm 2** “When” Query-Based Explanation

---

**Input:** Agent group  $\mathcal{G}_q$ , query task  $\tau_q$ , Hasse diagrams  $\{\mathcal{D}_j\}$ , and feature set  $\mathcal{F}$

**Output:** Language-based explanation  $\mathcal{X}$

```
1: Extract relevant features  $\mathcal{F}_q \subseteq \mathcal{F}$  for  $\tau_q$ 
2: Initialize uncertainty dictionary  $U \leftarrow \emptyset$ 
3: for each diagram  $\mathcal{D}_j$  do
4:   if  $\tau_q$  completed by  $\mathcal{G}_q$  in  $\mathcal{D}_j$  then
5:     Let  $v_\tau$  be the node where  $\tau_q$  is completed
6:     Compute partial comparability graph from  $v_\tau$ 
7:     for each node  $v$  not reachable to/from  $v_\tau$  do
8:       Add features associated with  $v$  to  $U[\mathcal{D}_j]$ 
9: Label nodes as targets (containing  $\tau_q$  by  $\mathcal{G}_q$ ) or non-targets
10: Encode nodes as boolean vectors over  $\mathcal{F}_q$  using  $U$ 
11: Apply Quine-McCluskey to distinguish targets/non-targets
12: Translate resulting formula into explanation  $\mathcal{X}$ 
13: return  $\mathcal{X}$ 
```

---

whether agents 2 or 4 complete task C, as well as whether potentially prerequisite tasks (e.g., task A, B, etc.) have been completed.

Given a set of Hasse diagrams  $\{\mathcal{D}_j\}$  summarizing multiple episodes of decentralized policy execution, we check, for each diagram  $\mathcal{D}_j$ , whether the query task  $\tau_q$  is completed by the queried agent group  $\mathcal{G}_q$ . If so, we identify the corresponding node  $v_\tau$  and construct a *partial comparability graph* (Kelly 1985) centered at  $v_\tau$ , which includes only nodes with a known ordering relative to  $v_\tau$  (i.e., those reachable via the Hasse diagram’s edges in either direction).

To handle partial observability and structural ambiguity in decentralized execution, we introduce an *uncertainty dictionary*  $U$ . Any node that is not reachable to or from  $v_\tau$  is considered unordered with respect to the query task. Features associated with such nodes are marked as *uncertain* and stored in  $U[\mathcal{D}_j]$ . For example, if task B appears in a node unconnected to the node where agents 2 and 4 complete task C, we cannot determine whether task B occurred before or after task C. As a result, the feature “task B completed” is added to  $U[\mathcal{D}_j]$  and treated as a possible—but not confirmed—precondition for task C.

We then label nodes as *targets* if they satisfy the query (i.e., they contain  $\tau_q$  completed by  $\mathcal{G}_q$ ), and as *non-targets* otherwise. Each node is encoded as a boolean vector over the relevant feature set  $\mathcal{F}_q$ , where each bit indicates whether the corresponding feature is satisfied along a path to that node. To avoid underestimating dependencies, features marked as uncertain in  $U$  are conservatively treated as true in the boolean encoding.

To identify distinguishing conditions, we apply Quine-McCluskey (Quine 1952) to derive a minimal boolean formula that separates targets from non-targets. Finally, we translate the resulting formula into a language explanation using a structured language template, mapping certain features to “must” and uncertain ones to “may”.

An example of a generated explanation is: “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.”

While prior methods (Hayes and Shah 2017; Bogges,

---

**Algorithm 3** “What” Query-Based Explanation

---

**Input:** Query task  $\tau_q$ , Hasse diagrams  $\{\mathcal{D}_j\}$

**Output:** Language-based explanation  $\mathcal{X}$

```
1: Initialize sets  $\mathcal{T}^c \leftarrow \emptyset, \mathcal{T}^u \leftarrow \emptyset$ 
2: for each diagram  $\mathcal{D}_j$  do
3:   for each node  $v \in \mathcal{D}_j$  do
4:     if  $v$  contains  $\tau_q$  completed by any agent group then
5:       Add tasks from all immediate children of  $v$  to  $\mathcal{T}^c$ 
6:       Compute partial comparability graph from  $v$ 
7:       for each node  $v'$  not reachable to/from  $v$  do
8:         Add tasks from  $v'$  to  $\mathcal{T}^u$ 
9: Translate  $\mathcal{T}^c$  and  $\mathcal{T}^u$  into explanation  $\mathcal{X}$ 
10: return  $\mathcal{X}$ 
```

---

Kraus, and Feng 2022) also use Quine-McCluskey minimization followed by language translation, they do not account for the uncertainty introduced by decentralized execution. In contrast, Algorithm 2 incorporates partial comparability graphs and an uncertainty dictionary to capture unordered task dependencies, enabling “may” conditions in the resulting explanations.

**Complexity.** The dominant cost of Algorithm 2 is Quine-McCluskey minimization with worst-case complexity  $\mathcal{O}(3^{|\mathcal{F}_q|}/\ln |\mathcal{F}_q|)$ . Other steps scale linearly with the number of Hasse diagrams and nodes. The method is tractable in practice for moderate feature sizes.

## 4.2 Answering “Why Not” Queries

To answer queries of the form: “Why don’t agents  $\mathcal{G}_q$  do task  $\tau_q$  under conditions  $\Phi_q$ ?”, we adapt the procedure used for “When” queries. Instead of identifying preconditions for successful completions, the goal is to isolate the minimal set of missing conditions that prevent  $\tau_q$  from occurring under the given scenario.

The key difference lies in how we define the target and non-target sets: the user-provided condition  $\Phi_q$  is encoded as the target (i.e., a case where the task did not occur), while nodes from Hasse diagrams where  $\tau_q$  is successfully completed by  $\mathcal{G}_q$  serve as non-targets. As in the “When” query, we construct partial comparability graphs to identify ordering uncertainty and maintain an uncertainty dictionary to track ambiguous dependencies. These are incorporated into the boolean encoding, allowing us to apply Quine-McCluskey minimization to identify which missing features distinguish the query condition from successful executions. The full algorithm pseudocode is provided in Appendix B and shares the same complexity as Algorithm 2.

For instance, for the query “Why don’t agents 2 and 4 complete task C when only task A is completed?”, the resulting explanation could be: “Task B *may* need to be completed for agents 2 and 4 to complete task C.”

## 4.3 Answering “What” Queries

To answer queries of the form: “What do the agents do after task  $\tau_q$ ?”, we analyze the successors of  $\tau_q$  across multiple Hasse diagrams. Our goal is to identify which tasks occur after  $\tau_q$ , distinguishing between those that are certainly

ordered afterward and those that may follow, but whose ordering is ambiguous due to decentralized execution.

Given a set of Hasse diagrams  $\{\mathcal{D}_j\}$  generated from simulated episodes, we first locate all nodes where  $\tau_q$  is completed. For each such node, we add the tasks from its immediate children, representing actions that are explicitly ordered after  $\tau_q$ , to a set of certain successors  $\mathcal{T}^c$ .

To identify uncertain successors  $\mathcal{T}^u$ , we construct a partial comparability graph rooted at each node where  $\tau_q$  is completed. We then collect tasks from nodes that are not ordered with respect to it. These tasks are added to a set  $\mathcal{T}^u$  as possible, but not guaranteed, successors of  $\tau_q$ .

We generate an explanation using a language template that reports both the certain and uncertain successor sets. For example, for the query “What do agents do after task C is completed?”, the explanation could be: “After task C is completed, tasks D and E are completed. Additionally, task B *may* be completed.”

**Complexity.** The worst-case time complexity of Algorithm 3 is  $\mathcal{O}(|\{\mathcal{D}_j\}| \cdot |\mathcal{V}|^2(|\mathcal{V}| + |\mathcal{E}|))$ , where  $|\{\mathcal{D}_j\}|$  is the number of Hasse diagrams, and  $|\mathcal{V}|$  and  $|\mathcal{E}|$  are the number of nodes and edges in each diagram, respectively.

## 5 Computational Experiments

**MARL Domains.** We evaluate our approaches on four benchmark domains: (1) *Search and Rescue (SR)* (Boggess, Kraus, and Feng 2022), where agents rescue victims and fight fires; (2) *Level-Based Foraging (LBF)* (Papoudakis et al. 2021), where agents collect food; (3) *Multi-Robot Warehouse (RW)* (Papoudakis et al. 2021), where agents pick up and deliver items; and (4) *Pressure Plate (PP)* (McInroe and Christianos 2022), where agents open doors to enable others’ navigation. All domains are gridworld-based. Agents observe nearby grid cells only: up to four per direction in PP and one per direction in other domains, reflecting the partial observability of decentralized execution.

**Experimental Setup.** We train policies using two MARL algorithms: Shared Experience Actor-Critic (SEAC) (Christianos, Schäfer, and Albrecht 2020) for CTDE, and Independent Advantage Actor-Critic (IA2C) (Papoudakis et al. 2021) for DTDE. Each model is trained until convergence or for up to 400 million steps. All experiments are conducted on a machine with a 2.1 GHz Intel CPU, 132 GB RAM, and Ubuntu 22.04.

### 5.1 Evaluation on Policy Summarization

**Summarization Baseline.** Since no existing methods summarize decentralized MARL policies, we adapt the single-agent approach from (McCalmon et al. 2022) as a baseline. For each agent, we construct an *abstract policy graph* that summarizes task sequences observed over 100 episodes, using the same abstract features as our HDS method for fair comparison. The resulting agent-specific graphs are displayed side-by-side and annotated with task sequence probabilities (see Figure 5 in Appendix C).

**Results.** Table 1 compares CTDE policy summarization sizes (number of nodes and edges) produced by our HDS method and the baseline for the largest configuration in each

Domain ( $N,  T $ )	HDS		Baseline	
	$ \mathcal{V} $	$ \mathcal{E} $	$ \mathcal{V} $	$ \mathcal{E} $
SR (9,7)	8	7.88	534	525
LBF (9,9)	10	10.83	723	714
RW (4,19)	20	19	1,274	1,270
PP (7,6)	7	6	265	258

Table 1: Summarization sizes for HDS and the baseline on the largest setting in each domain, based on 100 episodes executed using CTDE policies trained with SEAC.

Domain ( $N,  T $ )	HD-When		Baseline	
	$ \mathcal{F}^c $	$ \mathcal{F}^u $	$ \mathcal{F}^c $	$ \mathcal{F}^u $
SR (9,7)	9	2	54	0
LBF (9,9)	13	11	104	0
RW (4,19)	0	153	267	0
PP (7,6)	8	3	20	0

Table 2: Explanations sizes for our method and the baseline on the largest setting in each domain, based on 100 episodes executed using CTDE policies trained with SEAC.

domain (i.e., with the most agents and tasks). HDS generates one Hasse diagram per episode, each representing a complete summary of all agents’ behavior in that episode; we report average size over 100 episodes. In contrast, the baseline displays all observed task sequences across episodes for all agents, resulting in large visualizations with hundreds of nodes and edges, which are significantly harder to interpret than the compact Hasse diagrams. Similar results hold for DTDE policies trained with IA2C (see Appendix C).

Both HDS and the baseline are computationally efficient, each processing 100 episodes and generating summarizations in under one second across all domains.

Additionally, while HDS often produces unique Hasse diagrams across episodes, they typically fall into a small number of structural types based on edge counts. For example, SR(9,7) yields 100 unique diagrams but only 6 distinct edge counts. Figure 6 in Appendix C illustrates the distribution of diagram types across domains, highlighting HDS’s ability to capture both behavioral diversity and structural regularity.

### 5.2 Evaluation on Query-Based Explanations

**Explanation Baseline.** Since no existing methods generate query-based explanations for decentralized MARL, we adapt the single-agent explanation approach from (Hayes and Shah 2017) by applying it independently to each agent’s abstract policy graph (described in Section 5.1). The resulting per-agent explanations are then merged using simple union-based aggregation as a baseline.

**Results.** Table 2 compares explanation sizes for “When” queries on CTDE policies, measured by the number of certain ( $|\mathcal{F}^c|$ ) and uncertain ( $|\mathcal{F}^u|$ ) features extracted by our method and the baseline. These features are derived from boolean formulas obtained through Quine-McCluskey min-

imization. In all cases, both methods generate explanations in under one second.

The baseline includes only certain features, as it does not capture task ordering uncertainty. It also produces significantly larger explanations due to the size of its aggregated policy graphs and union-based merging of per-agent results. In contrast, our HD-When method yields more compact and informative explanations, balancing certain and uncertain features. In RW(4,19), all features are marked uncertain due to highly asynchronous execution.

Similar trends are observed for DTDE policies and for other query types (see Appendix C).

## 6 User Studies

We conducted two user studies (with IRB approval) to evaluate the effectiveness of our proposed policy summarizations and query-based explanations.

For both studies, we recruited participants via university mailing lists to answer surveys via Qualtrics. Eligible participants were fluent English speakers over 18 years old and were incentivized with bonus payments for correctly answering questions based on the provided summarizations or explanations. The summarization study included 20 participants (10 males, 9 females, 1 other), with an average age of 22.55 years ( $SD = 2.89$ ). The explanation study included 21 participants (14 males, 6 females, 1 other), with an average age of 24 years ( $SD = 3.95$ ).

We describe the study design and results for each study in Sections 6.1 and 6.2, respectively.

### 6.1 Summarization Study

**Independent Variables.** The independent variable in this study was the summarization generation approach: our HDS method or the baseline described in Section 5.1. The baseline displays side-by-side abstract policy graphs for each agent, with each agent’s most likely task sequence highlighted (Figure 5, Appendix C). To aid interpretation, the HDS interface presents a table of agent-task assignments and a list of task-ordering rules converted from each Hasse diagram. It shows the top three most frequent plans from 100 episodes, annotated with empirical likelihoods, and highlights the most likely one in red (Figure 7, Appendix D).

**Procedures.** The study followed a within-subject design where each participant completed two trials, one per summarization method (HDS and baseline). Each trial consisted of two summarizations, and each summarization was followed by three questions (12 questions total). Participants were randomly assigned to one of two groups to counterbalance ordering effects (HDS first or baseline first). All questions were randomized. Prior to each trial, participants received a brief tutorial and passed attention-check questions to ensure engagement. Bonus incentives and timing were used to promote data quality.

**Dependent Measures.** We assessed user performance based on the number of correctly answered questions in three categories: *assignment* (e.g., “Can [robot(s)] complete [task]?”), *likelihood* (e.g., “What are the most likely robot(s) to complete [task]?”), and *order* (e.g., “Must [task 1] always

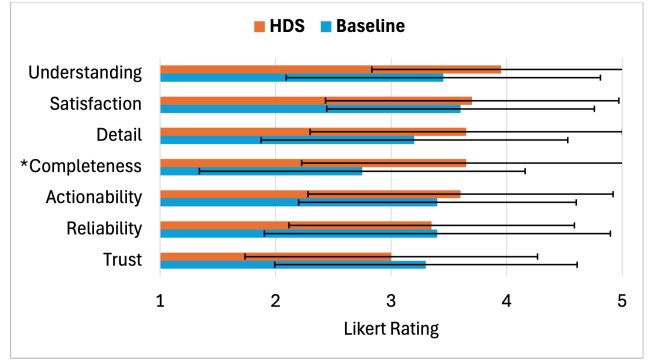


Figure 2: Mean and standard deviation of participant ratings on policy summarizations (\* indicates statistically significant difference).

be completed before [task 2]?”). Response time was also recorded for each question.

At the end of each trial, participants rated the summarization quality on a 5-point Likert scale across seven metrics (Hoffman et al. 2018): *understanding*, *satisfaction*, *detail*, *completeness*, *actionability*, *reliability*, and *trustworthiness*. Participants were informed of their accuracy before providing ratings.

**Hypotheses.** We hypothesized that, compared to baseline summarizations, HDS would (**H1**) improve user question-answering performance and (**H2**) receive higher ratings on summarization quality metrics.

**Results.** We found that users answered significantly more questions correctly using HDS ( $M=4.25$  out of 6,  $SD=0.83$ ) than with the baseline ( $M=3.1$  out of 6,  $SD=1.04$ ). A paired t-test confirms this difference is statistically significant ( $t(19)=4.2$ ,  $p \leq 0.01$ ,  $d=0.96$ ). *The data supports H1.*

Regarding user-perceived summarization quality, HDS was rated slightly higher in *completeness* (Wilcoxon signed-rank,  $W=16.0$ ,  $Z=-2.07$ ,  $p \leq 0.04$ ,  $r=-0.33$ ), but not significantly different in other dimensions (see Figure 2). *The data partially supports H2.*

**Discussion.** These results suggest that HDS improves objective user performance in answering questions that require understanding task coordination across agents. Because the baseline does not explicitly model inter-agent cooperation, users must compare across graphs to infer coordination, which can be cognitively demanding. Moreover, the baseline may mislead by showing each agent’s most likely task sequence independently, which may not reflect the most likely joint behavior in coordination.

Subjective ratings show limited preference for HDS, possibly due to user familiarity with the baseline’s flowchart-style layout despite its reduced clarity on coordination.

Finally, we note that users’ response times remained comparable between methods. In some cases, HDS enabled faster responses (e.g., for likelihood queries).

### 6.2 Explanation Study

**Independent Variables.** The independent variable was the



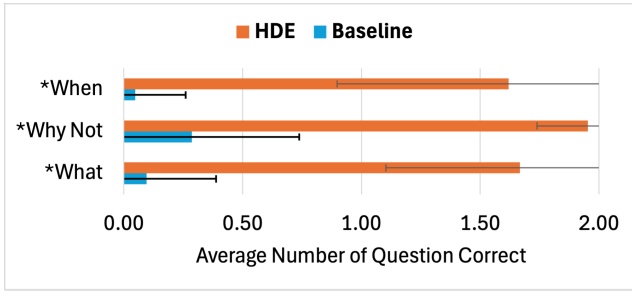


Figure 3: Mean and standard deviation of participant performance on query-based explanations (\* indicates statistically significant difference).

explanation generation method: our proposed approach versus the baseline described in Section 5.2. Example interfaces for all three query types (“When,” “Why Not,” and “What”) are shown in Appendix D.

For “When” queries, users viewed a map of a search-and-rescue scenario (with four agents and four tasks) and an accompanying explanation. Our method outputs both required (certain) and possible (uncertain) conditions, while the baseline includes only certain ones.

For “Why Not” queries, users were shown two maps: the first with a failed task and an explanation of the violated conditions, and the second used to test whether the explanation helped predict behavior.

For “What” queries, users received an explanation about what tasks could occur next. Our method distinguishes certain and uncertain tasks; the baseline lists only certain ones.

**Procedures.** The study followed a within-subject design, where each participant completed two trials—one using our method, and one using the baseline. Each trial included two questions per query type (“When,” “Why Not,” and “What”), totaling 12 questions. Method order was counter-balanced across participants to mitigate ordering effects. The study included a demonstration, attention-check questions, bonus incentives, and timing to ensure data quality.

**Dependent Measures.** User performance was measured by the number of correctly answered prediction questions, reported separately for each query type. Response time per question was also recorded. After each trial, participants rated explanation quality on a 5-point Likert scale (Hoffman et al. 2018) using the same seven metrics described in Section 6.1.

**Hypotheses.** We hypothesized that, compared to the baseline, our generated explanations would (**H3**) improve user question-answering performance and (**H4**) receive higher ratings on explanation quality metrics.

**Results.** As shown in Figure 3, users answered significantly more questions correctly using our HDE explanations compared to the baseline across all three query types. Paired t-tests ( $\alpha = 0.05$ ) confirm the improvement for *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* ( $t(20)=12.05$ ,  $p \leq 0.01$ ,  $d=2.69$ ). *The data supports H3.*

Figure 4 shows participant ratings on explanation qual-

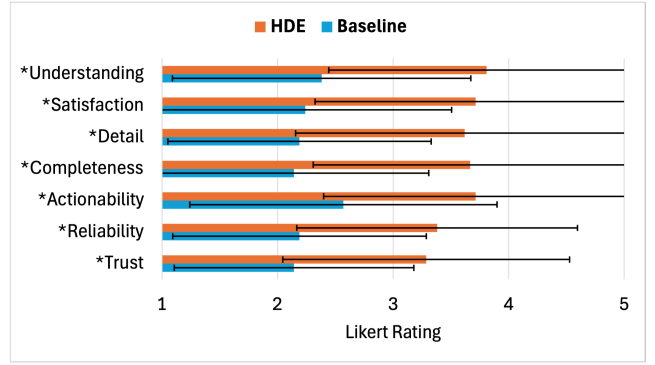


Figure 4: Mean and standard deviation of participant ratings on query-based explanations (\* indicates statistically significant difference).

ity. Wilcoxon signed-rank tests ( $\alpha = 0.05$ ) indicate significantly higher ratings for HDE across all seven metrics: *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* ( $W=0.0$ ,  $Z=-2.68$ ,  $p \leq 0.01$ ,  $r=0.41$ ). *The data supports H4.*

**Discussion.** The improvement in user performance is likely due to our explanation method’s ability to present both certain and uncertain features, enabling users to more accurately predict if and what tasks will occur. In contrast, the baseline captures only agent-specific behavior and lacks inter-agent context. It may omit shared task dependencies or include irrelevant conditions observed by individual agents, requiring users to reconcile fragmented information—reducing its effectiveness in decentralized settings.

Participants also gave higher subjective ratings to our explanations across all goodness metrics, suggesting they valued access to decentralized task dependencies and uncertainty, even if the explanations were more complex. Notably, the inclusion of uncertain features did not increase response time, indicating that users could efficiently interpret the richer information.

## 7 Conclusion

We presented novel approaches for summarizing and explaining decentralized MARL policies. Computational experiments across four MARL domains and two learning algorithms show that our method is scalable and efficient, generating compact summarizations and meaningful explanations even in large environments with many agents and tasks. User studies further demonstrate that our approach improves user performance and perceived explanation quality, without increasing response time.

Future work includes integrating these explanations into interactive human-agent systems, supporting more expressive user queries, and leveraging large language models to enhance explanation clarity and usability.

## Acknowledgements

This work was supported in part by the U.S. National Science Foundation grant CCF-1942836, and Israel Ministry of Innovation, Science & Technology grant 1001818511. The opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsoring agencies.

## References

- Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2023. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.
- Amir, D.; and Amir, O. 2018. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 1168–1176.
- Atrey, A.; Clary, K.; and Jensen, D. 2019. Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning. In *International Conference on Learning Representations*.
- Boggess, K.; Kraus, S.; and Feng, L. 2022. Toward Policy Explanations for Multi-Agent Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*.
- Boggess, K.; Kraus, S.; and Feng, L. 2023. Explainable Multi-Agent Reinforcement Learning for Temporal Queries. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*.
- Chen, J.; Wang, Y.; Wang, J.; Xie, X.; Hu, J.; Wang, Q.; and Xu, F. 2025. Understanding Individual Agent Importance in Multi-Agent System via Counterfactual Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 15785–15794.
- Christianos, F.; Schäfer, L.; and Albrecht, S. 2020. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33: 10707–10717.
- Dinneweth, J.; Boubezoul, A.; Mandiau, R.; and Espié, S. 2022. Multi-agent reinforcement learning for autonomous vehicles: A survey. *Autonomous Intelligent Systems*, 2(1): 27.
- Hayes, B.; and Shah, J. A. 2017. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 303–312.
- Heuillet, A.; Couthouis, F.; and Díaz-Rodríguez, N. 2022. Collective explainable AI: Explaining cooperative strategies and agent contribution in multiagent reinforcement learning with shapley values. *IEEE Computational Intelligence Magazine*, 17(1): 59–71.
- Hoffman, R. R.; Mueller, S. T.; Klein, G.; and Litman, J. 2018. Metrics for explainable AI: Challenges and prospects. *arXiv preprint arXiv:1812.04608*.
- Juozapaitis, Z.; Koul, A.; Fern, A.; Erwig, M.; and Doshi-Velez, F. 2019. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on explainable artificial intelligence*.
- Kelly, D. 1985. Comparability graphs. *Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Its Applications*, 3–40.
- Kottinger, J.; Almagor, S.; and Lahijanian, M. 2021. MAPS-X: Explainable multi-robot motion planning via segmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 7994–8000. IEEE.
- Krnjaic, A.; Stealeac, R. D.; Thomas, J. D.; Papoudakis, G.; Schäfer, L.; To, A. W. K.; Lao, K.-H.; Cubuktepe, M.; Haley, M.; Börsting, P.; et al. 2022. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. *arXiv preprint arXiv:2212.11498*.
- Madumal, P.; Miller, T.; Sonenberg, L.; and Vetere, F. 2020. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2493–2500.
- Mahjoub, O.; de Kock, R. J.; Singh, S.; Khelifi, W.; Vall, A.; ab Tessera, K.; Gorsane, R.; and Pretorius, A. 2024. Efficiently Quantifying Individual Agent Importance in Cooperative MARL. In *eXplainable AI approaches for Deep Reinforcement Learning*.
- McCalmon, J.; Le, T.; Alqahtani, S.; and Lee, D. 2022. Caps: Comprehensible abstract policy summaries for explaining reinforcement learning agents. In *nt’l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*.
- McInroe, T.; and Christianos, F. 2022. PRESSURE-PLATE. <https://github.com/uoe-agents/pressureplate>. Accessed: 2022-11-22.
- Milani, S.; Topin, N.; Veloso, M.; and Fang, F. 2023. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*.
- Milani, S.; Zhang, Z.; Topin, N.; Shi, Z. R.; Kamhoua, C.; Papalexakis, E. E.; and Fang, F. 2022. MAVIPER: Learning Decision Tree Policies for Interpretable Multi-agent Reinforcement Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 251–266.
- Mualla, Y.; Tchappi, I.; Kampik, T.; Najjar, A.; Calvaresi, D.; Abbas-Turki, A.; Galland, S.; and Nicolle, C. 2022. The quest of parsimonious XAI: A human-agent architecture for explanation formulation. *Artificial intelligence*, 302: 103573.
- Papoudakis, G.; Christianos, F.; Schäfer, L.; and Albrecht, S. V. 2021. 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)*.
- Quine, W. V. 1952. The problem of simplifying truth functions. *The American mathematical monthly*, 59(8): 521–531.
- Sarkar, S. K. 2017. *A Textbook of Discrete Mathematics*, chapter 9.4 “Hasse Diagram”, 339–341. S. Chand Publishing, 9th edition.
- Sreedharan, S.; Soni, U.; Verma, M.; Srivastava, S.; and Kambhampati, S. 2022. Bridging the Gap: Providing Post-Hoc Symbolic Explanations for Sequential Decision-Making Problems with Inscrutable Representations. In *International Conference on Learning Representations*.



Topin, N.; and Veloso, M. 2019. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2514–2521.

Wells, L.; and Bednarz, T. 2021. Explainable ai and reinforcement learning—a systematic review of current approaches and trends. *Frontiers in Artificial Intelligence*, 4: 48.

## A Proof of Theorem 1

**Theorem 1.** *Given a set of agent trajectories  $\{\omega^i\}_{i=1}^N$  produced by executing decentralized MARL policies  $\{\pi^i\}_{i=1}^N$  in a single episode, the Hasse diagram  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$  constructed by Algorithm 1 is both a correct and complete policy summarization.*

*Proof. (Correctness)* Suppose, for contradiction, that the Hasse diagram  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$  produced by Algorithm 1 is not a correct policy summarization. Then there exists a path  $\rho$  through  $\mathcal{D}$  such that its projection  $\rho^i$  onto some agent  $i$  does not conform to the agent’s task sequence trace( $\omega^i$ ).

This implies that in  $\rho^i$  there is a task  $\tau_k^i$  that precedes another task  $\tau_{k'}^i$ , with  $k > k'$ , yet  $\tau_k^i$  appears earlier in the path than  $\tau_{k'}^i$ . Let  $v$  and  $v'$  be the nodes in  $\rho$  that contain  $\tau_k^i$  and  $\tau_{k'}^i$ , respectively. Since  $\rho$  is a valid path, it must be that  $v \prec v'$ , meaning tasks in  $v$  occur before those in  $v'$ . But then  $\tau_k^i$  should occur before  $\tau_{k'}^i$  in trace( $\omega^i$ ), contradicting  $k > k'$ . Hence, all projections must conform, and the diagram is correct.

**(Completeness)** The algorithm explicitly iterates over each trajectory  $\omega^i$  and processes its task sequence trace( $\omega^i$ ) in order. For every consecutive pair of tasks  $(\tau_{k-1}^i, \tau_k^i)$ , an edge is added from the vertex containing  $\tau_{k-1}^i$  to the one containing  $\tau_k^i$ . Thus, for each agent, a path through the DAG is constructed that corresponds exactly to trace( $\omega^i$ ).

In the transitive reduction step, an edge  $(v, v')$  is removed only if there already exists an alternative path from  $v$  to  $v'$ . Hence, the path corresponding to trace( $\omega^i$ ) is preserved. Therefore, for every agent  $i$ , there exists at least one path  $\rho$  in  $\mathcal{D}$  such that  $\rho^i = \text{trace}(\omega^i)$ , proving completeness.  $\square$

## B Algorithm for “Why Not” Queries

The pseudocode below outlines our method for answering “Why don’t agents  $\mathcal{G}_q$  do task  $\tau_q$  under conditions  $\Phi_q$ ?” queries, as described in Section 4.2.

## C Details on Computational Experiments

**Baseline Summarization Visualization.** Figure 5 shows an example of the baseline summarization used throughout our evaluations. Each agent’s abstract policy graph summarizes its possible task sequences over 100 execution episodes, with sequence-level probabilities. These per-agent graphs are displayed side-by-side, requiring users to manually infer inter-agent task coordination, which is a key limitation compared to our HDS method.

**DTDE Policy Summarization Results.** Table 3 reports summarization statistics for HDS and the baseline using

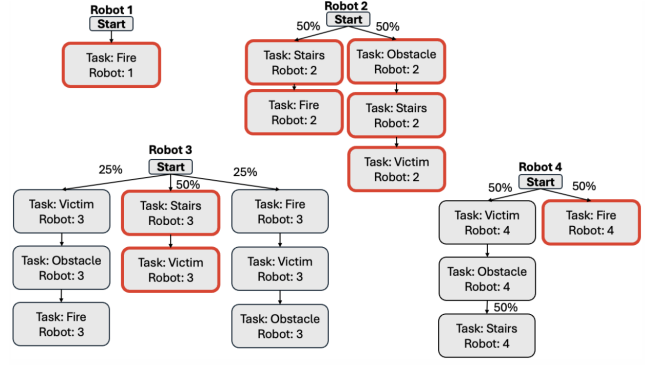


Figure 5: Example of baseline summarization.

DTDE policies trained with IA2C. Due to convergence issues of training IA2C in larger environments, we report results only for LBF(5,5) and RW(3,4), rather than LBF(9,9) and RW(4,19) used in the CTDE setting with SEAC. As in all CTDE cases, HDS yields substantially more compact summarizations, whereas the baseline produces significantly larger graphs.

Domain ( $N,  T $ )	HDS		Baseline	
	$ \mathcal{V} $	$ \mathcal{E} $	$ \mathcal{V} $	$ \mathcal{E} $
SR (9,7)	8	7.79	350	341
LBF (5,5)	6	5.62	151	146
RW (3,4)	5	4	69	66
PP (7,6)	7	6	107	100

Table 3: Summarization sizes for HDS and the baseline on selected domain settings, based on 100 episodes executed using DTDE policies trained with IA2C.

**Hasse Diagram Diversity.** Figure 6 shows the distribution of Hasse diagram types across 100 episodes for selected domain-policy pairs. Although individual executions often yield unique diagrams, they typically fall into a small num-

### Algorithm 4 “Why Not” Query-Based Explanation

**Input:** Agent group  $\mathcal{G}_q$ , query task  $\tau_q$ , query conditions  $\Phi_q$ , Hasse diagrams  $\{\mathcal{D}_j\}$ , and feature set  $\mathcal{F}$

**Output:** Language-based explanation  $\mathcal{X}$

- 1: Extract relevant features  $\mathcal{F}_q \subseteq \mathcal{F}$  for  $\tau_q$
- 2: Initialize uncertainty dictionary  $U \leftarrow \emptyset$
- 3: **for** each diagram  $\mathcal{D}_j$  **do**
- 4:   **if**  $\tau_q$  completed by  $\mathcal{G}_q$  in  $\mathcal{D}_j$  **then**
- 5:     Let  $v_\tau$  be the node where  $\tau_q$  is completed
- 6:     Compute partial comparability graph from  $v_\tau$
- 7:     **for** each node  $v$  not reachable to/from  $v_\tau$  **do**
- 8:       Add features associated with  $v$  to  $U[\mathcal{D}_j]$
- 9: Encode  $\Phi_q$  as boolean vector  $B_1$  over  $\mathcal{F}_q$
- 10: Encode non-target nodes as boolean vectors  $B_0$  using  $U$
- 11: Apply Quine-McCluskey to distinguish  $B_1$  from  $B_0$
- 12: Translate resulting formula into explanation  $\mathcal{X}$
- 13: **return**  $\mathcal{X}$

ber of structural categories based on edge count. For instance, SR(9,7) has 100 unique diagrams but only 6 distinct edge counts. This highlights HDS’s ability to preserve behavioral variation while maintaining structural compactness.

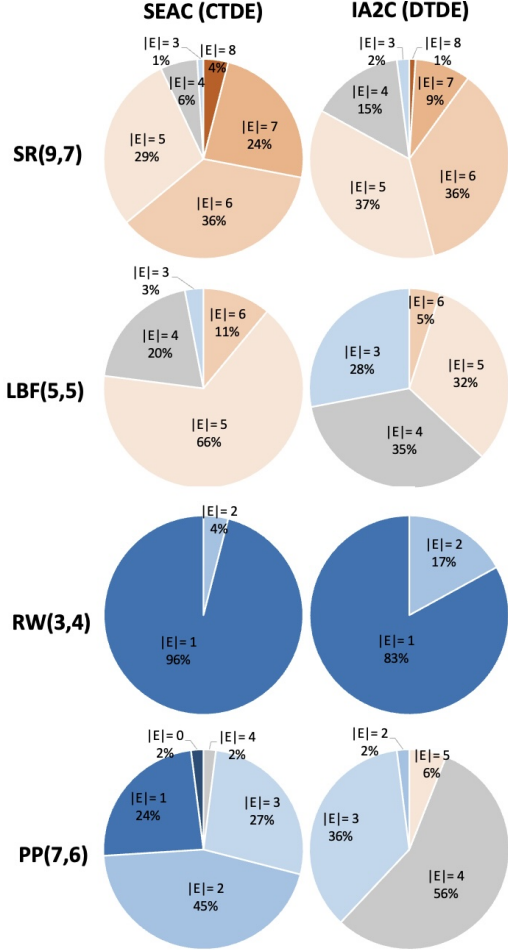


Figure 6: Distribution of Hasse diagram types across 100 episodes for each domain and training method. Each slice represents a unique diagram type categorized by edge count, illustrating structural diversity captured by HDS.

**Additional Explanation Results.** Tables 4, 5 and 6 provide additional explanation statistics for HD-When and HD-What under DTDE policies, and HD-What under CTDE policies. These tables report the number of certain and uncertain features or tasks included in the explanations, based on 100 episodes. Results in Table 4 show trends consistent with those observed in Table 2 for CTDE “When” queries: our methods produce compact, structured explanations with meaningful uncertainty, while the baseline includes more features but no uncertainty terms. Results in Tables 5 and 6 show explanations with increased information due to captured coordination and task order not present in the baseline.

Domain ( $N,  T $ )	HD-When		Baseline	
	$ \mathcal{F}^c $	$ \mathcal{F}^u $	$ \mathcal{F}^c $	$ \mathcal{F}^u $
SR (9,7)	5	5	45	0
LBF (5,5)	14	4	41	0
RW (3,4)	0	2	2	0
PP (7,6)	8	3	19	0

Table 4: Explanations sizes for our method and the baseline on selected domain settings, based on 100 episodes executed using DTDE policies trained with IA2C.

Domain ( $N,  T $ )	HD-What		Baseline	
	$ \mathcal{T}^c $	$ \mathcal{T}^u $	$ \mathcal{T}^c $	$ \mathcal{T}^u $
SR (9,7)	1	7	3	0
LBF (5,5)	4	0	3	0
RW (3,4)	1	3	2	0
PP (7,6)	2	3	2	0

Table 5: Explanations sizes for our method and the baseline on selected domain settings, based on 100 episodes executed using DTDE policies trained with IA2C.

## D Details on User Studies

**Summarization Study Interface Example.** Figure 7 shows an example of the user interface used in the summarization study for presenting HDS-generated summaries. Each summary consists of a table showing agent-task assignments and a list of natural language rules representing partial task order, converted from a Hasse diagram. The interface displays the top three most frequent summaries (based on 100 episodes), with empirical likelihoods shown beneath each. The most likely summary is visually emphasized with a red border.

Plan Option 1					Plan Option 2					Plan Option 3				
Task Rules					Task Rules					Task Rules				
Obstacle before Stairs					Victim before Obstacle					Stairs before Fire				
Stairs before Victim					Obstacle before Stairs					Fire before Victim				
Task Assignments					Task Assignments					Task Assignments				
	Victim	Fire	Stairs	Obstacle		Victim	Fire	Stairs	Obstacle		Victim	Fire	Stairs	Obstacle
Robot 1		X			Robot 1		X	X		Robot 1		X	X	
Robot 2	X		X	X	Robot 2					Robot 2		X		
Robot 3	X		X		Robot 3	X	X		X	Robot 3	X	X		X
Robot 4		X			Robot 4	X		X	X	Robot 4	X			X
Likelihood: 50%					Likelihood: 25%					Likelihood: 25%				

**Bonus Question:** Can robots 2 and 3 complete checking the stairs together?

☐ Yes

☐ No

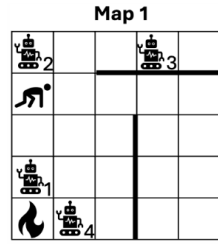
Figure 7: Example user interface displaying an HDS-generated summarization and associated question.

**Explanation Study Interface Examples.** Figures 8, 9 and 10 show example user interfaces used in the explanation

Domain ( $N,  T $ )	HD-What		Baseline	
	$ \mathcal{T}^c $	$ \mathcal{T}^u $	$ \mathcal{T}^c $	$ \mathcal{T}^u $
SR (9,7)	3	0	2	0
LBF (9,9)	8	0	6	0
RW (4,19)	9	11	10	0
PP (7,6)	1	5	2	0

Table 6: Explanations sizes for our method and the baseline on the largest setting in each domain, based on 100 episodes executed using CTDE policies trained with SEAC.

study for the “When,” “Why Not,” and “What” queries, respectively. Each interface presents a scenario with agents and tasks, along with an HD-generated explanation and a follow-up question for participants to answer.



**Explanation:** Robot 1 succeeds at fighting the fire when the following requirements are met:

**Requirements:**

- Robot 1 detects the fire
- Robot 4 detects the fire
- Obstacle is removed

**Possible requirements:**

- Stairs are checked
- Victim is rescued

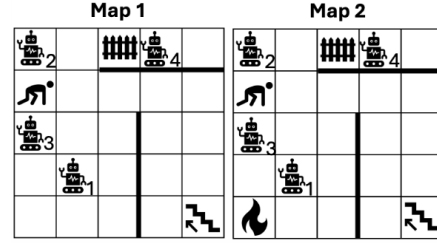
**Bonus Question:** Based on the explanation, would robot 1 succeed at fighting the fire in map 1?

☐ Yes

☐ Maybe

☐ No

Figure 8: Example user interface displaying an HD-When explanation and associated question.



**Explanation:** Robot 3 does not succeed at rescuing the victim in map 1 because the following requirements are violated:

**Requirements:**

- Obstacle is removed

**Possible requirements:**

- None

**Bonus Question:** Based on the explanation, would robot 3 succeed at rescuing the victim in map 2?

☐ Yes

☐ Maybe

☐ No

Figure 9: Example user interface displaying an HD-WhyNot explanation and associated question.

**Explanation:** After robot 2 checks the stairs,

**These tasks will be completed next:**

- Victim rescued
- Obstacle removed

**These tasks may be completed next:**

- Fire is fought

**Bonus Question:** Based on the explanation, select all tasks that could occur after robot 2 checks the stairs.

☐ Fire

☐ Stairs

☐ Victim

☐ Obstacle

☐ No more task are completed.

Figure 10: Example user interface displaying an HD-What explanation and associated question.