

Explanation and Knowledge Acquisition in Ad Hoc Teamwork

Hasra Dodampegama^[0000–0003–2302–1501] and
Mohan Sridharan^[0000–0001–9922–8969]

Intelligent Robotics Lab, School of Computer Science
University of Birmingham, Birmingham, UK
hhd968@student.bham.ac.uk
m.sridharan@bham.ac.uk

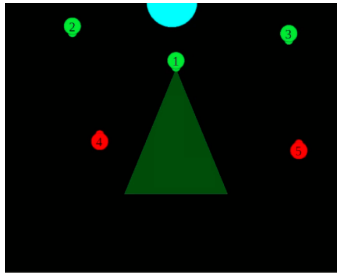
Abstract. State of the art frameworks for ad hoc teamwork i.e., for enabling an agent to collaborate with others “on the fly”, pursue a data-driven approach, using a large labeled dataset of prior observations to model the behavior of other agents and to determine the ad hoc agent’s behavior. It is often difficult to pursue such an approach in complex domains due to the lack of sufficient training examples and computational resources. In addition, the learned models lack transparency and it is difficult to revise the existing knowledge in response to previously unseen changes. Our prior architecture enabled an ad hoc agent to perform non-monotonic logical reasoning with commonsense domain knowledge and predictive models of other agents’ behavior that are learned from limited examples. In this paper, we enable the ad hoc agent to acquire previously unknown domain knowledge governing actions and change, and to provide relational descriptions as on-demand explanations of its decisions in response to different types of questions. We evaluate the architecture’s knowledge acquisition and explanation generation abilities in two simulated benchmark domains: Fort Attack and Half Field Offense.

Keywords: Non-monotonic logical reasoning · Ecological rationality · Knowledge acquisition · Explanation generation · Ad hoc teamwork.

1 Introduction

Ad hoc teamwork (AHT) refers to the problem of enabling an agent to collaborate with others without any prior coordination [27]. For example, consider the simulated multiagent domain *Fort Attack* (FA, Figure 1a), with a team of guards trying to defend a fort from a team of attackers [8], or *Half Field Offense* (HFO, Figure 1b), with a team of offense agents trying to score a goal against a team of defenders [16]. Agents in these domains have limited knowledge of each other, no prior experience of working as a team, and have to operate under partial observability and limited communication; these conditions also exist in many practical applications such as disaster rescue and surveillance.

The state of the art in AHT has moved from using predetermined policies for selecting actions in specific states to methods based on a key “data-driven”



(a) Fort Attack (FA)



(b) Half Field Offense (HFO)

Fig. 1: Screenshots from FA and HFO environments.

component [20]. This component uses a long history of prior experiences to build probabilistic or deep network methods that model the behavior of other agents (or agent types) and optimize the behavior of the ad hoc agent. However, in practical domains, it is difficult to gather such large training datasets of different situations and to adapt to unforeseen changes. Also, these methods lack transparency, and make it difficult to leverage commonsense domain knowledge to revise existing knowledge over time. Unlike existing work, we follow a *cognitive systems* approach, formulating AHT as a joint reasoning and learning problem. Our prior work developed a knowledge-guided architecture for AHT (**KAT**), which combined the principles of *refinement* and *ecological rationality*, enabling an ad hoc agent to determine its actions based on non-monotonic logical reasoning with prior domain knowledge and rapidly-learned predictive models of other agents' behaviors. In this paper, we significantly extend the architecture, enabling the ad hoc agent to:

1. Incrementally learn and revise previously unknown axioms and models that govern the dynamics of the domain; and
2. Provide on-demand relational descriptions of its decisions, and those of other agents, as *explanations* in response to different types of questions.

We demonstrate and evaluate these capabilities in the two simulated benchmark domains (FA, HFO). In particular, we illustrate how the interplay between representation, reasoning, and learning leads to incremental, reliable, and efficient knowledge acquisition and explanation generation.

2 Related Work

AHT has been researched under different names, as described in a recent survey [20]. Early work encoded specific protocols (or plays) for different scenarios, with an agent choosing specific protocols in specific states [6]. Subsequent work used sampling-based methods such as Upper Confidence bounds for Trees (UCT) to determine the ad hoc agent's action selection policy [5].

Many recent studies include a data-driven component that uses probabilistic, deep-network, and reinforcement learning (RL)-based methods to learn action

choice policies for the ad hoc agent based on a lengthy history or prior observations of different *types* of teammates or situations [4, 22]. For example, a RL method has been used to learn different policies for different teammate types, computing and using the best policy among the learned policies for a new teammate [4]. Also, attention-based deep neural networks have been used to jointly learn policies for different agent types [7], and to account for different team compositions [22]. Sequential and hierarchical variational auto-encoders have been used to model beliefs over other agents, and approximate belief inference has been achieved through meta-learning for a given prior [29]. Other work has combined learned policy methods with adversarial teammate prediction to account for changes in the agents’ behavior [24], and used Convolutional Neural Networks to detect and adapt to changing teammate types [23]. Sampling strategies have also been combined with such learning methods to optimize performance [28]. Such methods based on a data-driven learning component require considerable computation, memory, and training examples, build opaque models, and make it difficult to adapt to unexpected changes. Our architecture addresses these limitations by leveraging the complementary strengths of knowledge-based reasoning and data-driven learning methods.

Given the increasing use of AI methods in different applications, many architectures have utilized knowledge-based and data-derived methods to provide transparency in the operation of such methods [1, 19]. For example, prior work proposed a theory for explanation generation in human-robot interaction, using the declarative programming paradigm of Answer Set Prolog (ASP) to represent and reason with domain knowledge [25]. Other work has adapted this theory to support transparency in the decisions made by agents in scene understanding tasks [21]. In addition to transparency, agents operating in complex domains often need to revise their existing knowledge, and many different methods have been developed to support this ability. For example, a system based on inductive learning has been developed to learn new knowledge in the form of an ASP program [18]. Other approaches have used non-monotonic logical reasoning together with inductive learning and relational reinforcement learning to identify new rules for answer set programs [26].

The architecture developed in this work draws inspiration from the existing work to support reliable, efficient, and transparent reasoning and learning in ad hoc teamwork, enabling the ad hoc agent to adapt to different teammates and opponents, acquire and revise domain knowledge, and provide on-demand explanations in response to different types of questions.

3 Architecture

Figure 2 provides an overview of KAT, our architecture for ad hoc teamwork. Our ad hoc agent performs non-monotonic logical reasoning with prior commonsense domain knowledge and models of other agents’ behaviors learned and revised incrementally from limited examples, using heuristic methods to guide reasoning and learning. At each step, all agents receive observations of the domain state,

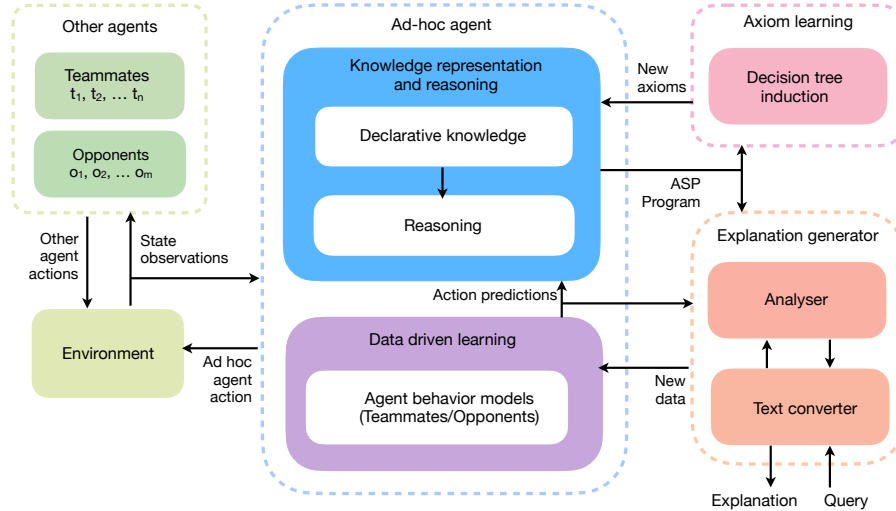


Fig. 2: Architecture combines complementary strengths of knowledge-based and data-driven reasoning and learning.

and they independently determine and execute their individual actions in the environment. The *axiom learning* component enables the ad hoc agent to acquire and revise its knowledge governing changes in the domain, and the *explanation generator* component enables it to provide relational descriptions as explanations of its decisions in response to questions. Please note that the basic reasoning components of our architecture were introduced in our conference papers [11, 9]. This paper primarily contributes the components for axiom learning and explanation generation; for completeness, we describe all components of our architecture in the following two example domains.

Example Domain 1 [Fort Attack (FA) Domain]

Consider three guards defending a fort from three attackers in the FA domain—Figure 1a [8]. An episode of the game ends if: (a) guards protect the fort for a given period of time; (b) all members of a team are terminated; or (c) an attacker reaches the fort. Each agent can move in one of the four cardinal directions with a specific velocity, rotate clockwise or anticlockwise, do nothing, or shoot an opponent within a range. The environment has four kinds of built-in policies for guards and attackers as described in Section 4.1. In our work, one of the guards is the ad hoc agent that can adapt to changes in the team and domain.

Example Domain 2 [Half Field Offense (HFO) Domain]

Consider a simulated 2D soccer domain where a team of offense agents are trying to score a goal against a team of defense agents (including a goalkeeper)—

Figure 1b [16]. An episode of the game ends if: (a) offense team scores a goal; (b) ball leaves the field; (c) defense team captures the ball; or (d) maximum episode length (500) is reached. There are two state space abstractions in HFO: low and high; we use the high-level features. There are three action abstractions: primitive, mid-level, and high-level; we use a combination of mid-level and high-level actions. In our work, our ad hoc agent is one of the offense agents. Similar to prior AHT methods, agents other than the ad hoc agent are selected from teams in the 2013 Robocup 2D simulation league competitions; offense agents are from *helios*, *gliders*, *cyrus*, *axiom*, *aut* and defense agents are from *agent2D*. The strategies of these agent types were trained using data-driven (probabilistic, deep, reinforcement) learning methods.

Prior commonsense knowledge in these domains includes relational descriptions of some domain/agent attributes, *e.g.*, location and shooting range, default statements, and axioms governing change, *e.g.*, an agent can only move to a location nearby, only shoot others within its shooting range (FA), and only score a goal from a certain angle (HFO). This knowledge may need to be revised over time.

3.1 Knowledge Representation and Reasoning

In KAT, the transition diagram of any domain is described using an extension of the action language \mathcal{AL}_d [13]. KAT’s domain representation comprises a system description \mathcal{D} , a collection of statements of \mathcal{AL}_d , and a history \mathcal{H} . \mathcal{D} has a sorted signature Σ with basic sorts, *e.g.*, *x_value*, *y_value*, *ad_hoc_agent*, *external_agent*, *step* (for temporal reasoning); actions, *e.g.*, *pass(ad_hoc_agent, external_agent)*, *dribble(ad_hoc_agent, x_value, y_value)*; statics, *i.e.*, domain attributes whose values cannot be changed by actions, *e.g.*, *next_to(x_value, y_value, x_value, y_value)* which describe the relative arrangement of places; and fluents, *i.e.*, attributes whose values can be changed by actions. Fluents can be *inertial*, *i.e.*, obey laws of inertia and changed by actions, *e.g.* *ball_loc(x_value, y_value)* encodes the ball’s location in the field; and *defined*, *i.e.*, not obey inertia laws and not directly changed by ad hoc agent’s actions, *e.g.*, an external agent’s location: *agent_loc(external_agent, x_value, y_value)*.

The domain dynamics are described in \mathcal{D} using three types of axioms: *causal law*, *state constraint*, and *executability condition*. Examples in HFO include:

$$\text{dribble}(R, X, Y) \text{ causes } \text{loc}(R, X, Y) \quad (1a)$$

$$\neg \text{has_ball}(A1) \text{ if } \text{has_ball}(A2), A1 \neq A2 \quad (1b)$$

$$\text{impossible } \text{pass}(R, T) \text{ if } \neg \text{has_ball}(R) \quad (1c)$$

Statement 5(a) is a causal law that states that dribbling to a place changes the location of the agent to that place. Statement 5(b) is a state constraint that implies only one agent can control the ball at any time. Statement 5(c) is an executability condition that prevents the ad hoc agent from considering the action of passing the ball when it does not control the ball. History \mathcal{H} is a record of observations and action executions, *i.e.*, relations of the form *obs(fluent, boolean, step)* and

$hpd(action, step)$ respectively, at specific time steps. It also includes initial state defaults, *i.e.*, statements initially believed to be true in most circumstances.

To reason with knowledge, the domain description is automatically translated to a program $\Pi(\mathcal{D}, \mathcal{H})$ in CR-Prolog [3], an extension to Answer Set Programming (ASP) that supports consistency restoring (CR) rules. ASP encodes *default negation* and *epistemic disjunction*, and supports non-monotonic reasoning; the ability to revise previously held conclusions. This ability is essential in practical multiagent collaboration domains in which agents often have to reason with incomplete knowledge and noisy observations. The CR rules allow the agent to recover from situations when the program $\Pi(\mathcal{D}, \mathcal{H})$ is inconsistent. For example, consider the situation in which an encoded default states that *attackers usually spread and attack* but the ad hoc agent observes an attacker displaying different behavior, e.g., mounting a frontal attack or staying back away from the shooting range of the guards. To address such exceptions to defaults, a CR rule is added to $\Pi(\mathcal{D}, \mathcal{H})$ for every default, allowing the ad hoc agent to assume that the default’s conclusion is false under exceptional circumstances, as a means of restoring consistency. For example:

$$\neg spread_attack(X) \leftarrow^{\pm} attacker(X)$$

allows the ad hoc agent to consider the rare situation of attackers mounting a frontal attack or display any behavior that does not agree with the default behavior.

$\Pi(\mathcal{D}, \mathcal{H})$ includes statements from \mathcal{D} and \mathcal{H} , inertia axioms, reality check axioms, closed world assumptions for defined fluents and actions, helper relations, e.g., $holds(fluent, step)$ and $occurs(action, step)$ to imply that a fluent is true and an action is part of a plan at a time step, and helper axioms that define goals and drive planning and diagnosis. Once the program Π is constructed, all reasoning tasks (e.g., planning, diagnostics, and inference) are reduced to computing *answer sets* of Π . We use the SPARC system [2] to write and solve CR-Prolog programs. Example programs for FA and HFO domains are in our open source repository [10].

3.2 Agent Behavior Models

The ad hoc agent’s decisions must take into account the state of the domain which is also governed by the actions of other agents. KAT thus also reasons with models that predict the behavior of the other agents. Our prior work introduced the use of the *Ecological Rationality* (ER) principle, which is based on Herb Simon’s definition of *Bounded Rationality* [14], and the algorithmic theory of heuristics [14, 15] to rapidly learn and revise these predictive models. The ER-based approach enables the ad hoc agent to choose relevant attributes and learn models of the behavior of other agents from limited training data (*e.g.*, 10000) while supporting rapid, incremental updates. Specifically, KAT enables the ad hoc agent to learn an ensemble of “fast and frugal” (FF) decision trees that predict the behavior of each type of other agents; each FF tree provides a

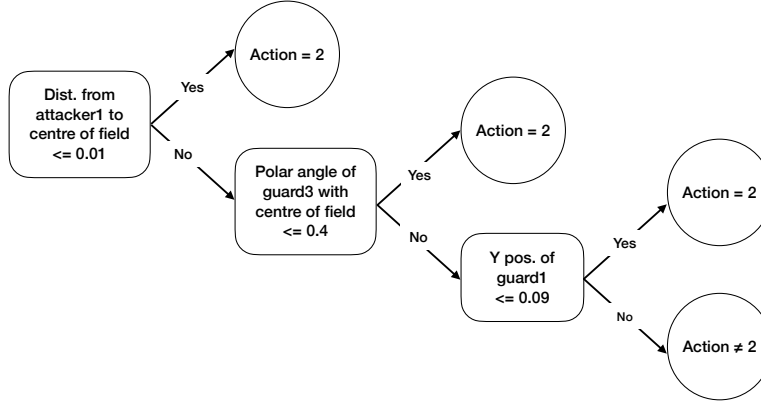


Fig. 3: FF tree in the ensemble for an attacker in the FA domain.

binary class label and the number of leaves in the tree is limited by number of attributes [17]. An individual FF tree learned for an attacker agent in the FA domain is shown in Figure 3.

Note that unlike state of the art AHT methods, KAT enables the predictive models of the other agents’ behavior to be learned and revised rapidly. Also, consistent agreement (disagreement) with predictions of an existing model triggers model choice (revision); the ad hoc agent is thus able to quickly adapt to changes in the domain or in the team composition.

3.3 Axiom Induction

Incrementally learning previously unknown domain knowledge is essential in complex domains such as ad hoc teamwork. One contribution of this paper is an approach to learn previously unknown causal laws and executability conditions governing the domain’s dynamics. Acquiring this knowledge will reduce ambiguity and enable the ad hoc agent to make more informed (also more reliable and efficient) decisions. We do so by adapting work that combined *decision tree induction* with knowledge-based reasoning [21], as described below.

For each candidate action, we collect possible state transitions involving this action and incrementally build a decision tree model by splitting nodes at each time step based on unused attributes. Every split must meet a minimum requirement of samples at each child. Then we build the candidate axioms by exploring different paths from the root to the leaves of the tree. These axioms are then validated for identifying further inconsistencies in the system. Specifically we:

1. Select an action and simulate the execution in different initial states to collect state transition information (e.g., start and end states, executed action,

and presence or absence of inconsistency). If the expected outcome is not observed after an action execution, that indicates the potential absence of an executability condition. If additional effects are observed (than what is expected), that indicates a missing causal law.

2. All fluent literals that exist in the answer set or initial state, and have an object constant that occurs in the executed action, are then stored as training data. The ground terms in the literals are further replaced by variables.
3. After data from a few execution traces are available, the decision tree models are created with relevant fluent literals as leaf nodes and the executed action as the root node. For executability conditions, the output or class label for each example is the presence or absence of an inconsistency. For causal laws, the output or class label for each example is the presence of unexpected fluent literals observed in the resultant state.
4. The candidate axioms are then generated along the path from the root node to the leaves in the decision trees. Examples of such path (*i.e.*, part of the decision tree) are illustrated in Figure 4 for the executability condition and the causal law below.

$$\neg occurs(shoot(R), I) \leftarrow \neg holds(has_ball(R), I). \quad (2)$$

$$occurs(rotate(R, D), I + 1) \leftarrow holds(face(R, D), I). \quad (3)$$

The candidate axioms then go through a series of validation checks. In particular, we only retain axioms that have sufficient support from the samples collected during the construction of the decision tree model of prior state transition experiences. In addition, we test the candidate axioms in some simulated situations similar to those used to collect the training examples and only retain axioms that do not cause any new inconsistencies. Furthermore, candidate axioms that pass the validation checks are merged with existing axioms to ensure that the more general version of the axioms (e.g., replace individual ground arguments of literals with their sorts) are included in the ad hoc agent’s knowledge base.

3.4 Transparency

Unlike methods in the existing literature that seek to make an entire learned model interpretable, or to explain (or justify) all the choices made by a reasoning system, KAT focuses on quickly identifying the relevant information to construct relational descriptions as *explanations* in response to causal, contrastive, or counterfactual questions about its decisions and beliefs. An automated decision-making system’s ability to reliably answer such questions about its decisions and beliefs promotes acceptability [1, 12]; this ability has been shown to play an important role in human reasoning and learning as well. KAT’s use of knowledge-based reasoning and simple predictive models provides the foundation of the approach introduced in this paper to support the desired transparency in the ad hoc agent’s operation. In addition, KAT’s approach for generating these

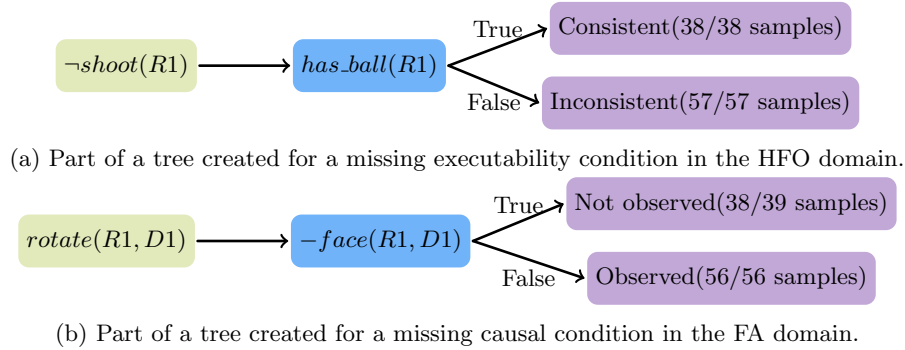


Fig. 4: Examples of part of trees created for missing axioms.

descriptions on-demand promotes computational efficiency. We build on prior work that demonstrated the ability to provide on-demand answers by iteratively and selectively identifying the axioms and literals that influence the desired action and belief, and have their antecedents satisfied in the relevant answer sets [21]. Specifically, KAT’s “*Explanation Generator*” in Figure 2 generates relational descriptions in response to four types of questions identified as being important in work on explainable planning [12]:

1. **(Action justification questions)** *Why did you do action A at step I?* When asked to justify an executed action, the ad hoc agent will:
 - extract actions A_{af} that occurred after A .
 - identify the axioms with a ($\in A_{af}$) in its head.
 - extract literals that would have prevented such an action from happening, e.g., from the body of its executability conditions.
 - identify that any such literal that exists in answer set at step I but not in $I+1$ was caused by the execution of A and thus enabled subsequent steps.
2. **(Contrastive questions)** *Why did you not do action A at step I?* When asked why an action was not included in the plan, the ad hoc agent will:
 - find executability conditions with A as its head to identify preconditions.
 - extract corresponding literals and check if they are satisfied by the answer set; each such literal prevented consideration of A .
 - if no preconditions of A are identified, compute cost of adding A to the computed plan. This will identify reasons for not selecting A .
3. **(Justify beliefs)** *Why did you believe L at step I?* To justify a belief at a specific step, ad hoc agent will:
 - find axioms that have given belief in its head.
 - extract related literals and check whether they are satisfied by the answer set. These will be the supporting statements for the belief.
 - if there are multiple supporting statements explaining a target belief, select one to provide the explanation. We leave the ranking of explanations and multi-step tracing of beliefs to future work that can build on ideas from [21].

4. **(Counterfactual Questions)** *What action do you think agent R will take at step I and why? What would be the outcome if you/agent R did action A instead?* When answering questions about the action choices of agents or the potential future state of the world, the agent will:
- make sure that it starts with the current best estimate of the state of the world, e.g., answer set at step I.
 - perform mental simulation of future steps from the current state using existing knowledge, including the action choices of other agents based on their behavior prediction models, and any specific ad hoc agent actions to be explored. Specifically, create potential next states of the world and explore the effects of specific actions or observations of interest.
 - use the newly acquired information to generate the explanation tailored to the question posed.

The newly acquired information/experience may also be used to further train the KAT system, specifically the behaviour prediction models of the other agents, for specific situations of interest. For all types of questions, the identified literals are processed with existing software tools and templates to generate textual descriptions provided as responses (i.e., explanations). We provide an execution trace in Section 4.3 of this approach, which can be used to answer questions during both planning and execution.

4 Experimental Setup and Results

Since KAT’s ability to reason with domain knowledge and learn behavior prediction models was already evaluated in our prior work, the work described in this paper evaluated three hypotheses:

- **H1:** KAT enables an ad hoc agent to accurately learn previously unknown causal laws and executability conditions;
- **H2:** Reasoning with incrementally learned axioms improves the performance of the ad hoc agent and the team; and
- **H3:** KAT supports the generation of relational descriptions as explanations of the ad hoc agent’s decisions and beliefs.

H1 was evaluated in both FA domain and HFO domain, with accuracy determined by the fraction of literals identified correctly in the learned axioms. As performance measures for H2, we used the team of guards’ win percentage in the FA domain, and the fraction of the goals scored by offense team in the HFO domain. H3 was evaluated in the FA domain, including qualitative evaluation. Further details of experiments and baselines are provided below.

4.1 Experimental Setup

FA domain provides four types of built-in policies for the attacker and guard agents as described below.

- **Policy220:** guards place themselves in front of the fort and shoot continuously; attackers try to approach the fort.
- **Policy650:** guards try to block the fort; attackers sneak in from all sides.
- **Policy1240:** guards spread and shoot the attackers; attackers spread and sneak in from all sides.
- **Policy1600:** guards are willing to move from the fort; some attackers approach the fort and shoot to distract guards while others try to sneak in.

For experiments in FA domain our ad hoc agent replaced one of the guard team members and each other agent in the team(guard/attacker) was assigned a policy from these built-in policies.

For the experiments in the **HFO domain**, we used six external agent teams from 2013 RoboCup simulation competition. The ad hoc agent represented one of the offense team members and the teammates were selected from *helios*, *gliders*, *cyrus*, *axiom* and *aut*, and the defense agents were based on *agent2d* team.

To evaluate **H1**, we used KAT in two sets of experiments; **Exp1**, in which we removed three axioms (two causal laws and one executability condition) from the agent’s knowledge base in the FA domain; **Exp2**, in which we removed four axioms (one causal law and three executability condition) from the agent’s knowledge base in the HFO domain.

FA domain:

$$\text{holds}(\text{face}(R, D), I + 1) \leftarrow \text{occurs}(\text{rotate}(R, D), I). \quad (4a)$$

$$\text{holds}(\text{in}(R, X, Y), I + 1) \leftarrow \text{occurs}(\text{move}(R, X, Y), I). \quad (4b)$$

$$\neg \text{occurs}(\text{shoot}(R, A), I) \leftarrow \neg \text{holds}(\text{in_range}(R, A), I). \quad (4c)$$

HFO domain:

$$\text{holds}(\text{in}(R, X, Y), I + 1) \leftarrow \text{occurs}(\text{dribble}(R, X, Y), I). \quad (5a)$$

$$\neg \text{occurs}(\text{dribble}(R, X, Y), I) \leftarrow \neg \text{holds}(\text{has_ball}(R), I). \quad (5b)$$

$$\neg \text{occurs}(\text{shoot}(R), I) \leftarrow \neg \text{holds}(\text{has_ball}(R), I). \quad (5c)$$

$$\neg \text{occurs}(\text{pass}(R, T1), I) \leftarrow \neg \text{holds}(\text{has_ball}(R), I). \quad (5d)$$

We ran the axiom learning algorithm 40 times; 20 times for each domain. Each time the ad hoc agent selected a different action and simulated its effects in each domain to incrementally learn all the missing axioms (see Section 3.3). As the performance measure we recorded the precision and recall of learning the missing axioms in terms of accurately identifying the literals in the axioms.

Next, in **Exp3** and **Exp4**, we allowed the agent to use the axioms it learned in **Exp1** and **Exp2** in the FA domain and HFO domain respectively. We conducted 100 game episodes in each domain with each built-in policy. As the baselines for these experiments (**Base1**) we used an ad hoc agent that does not use the learned axioms in its planning. These experiments were used to evaluate **H2**.

Finally to evaluate **H3**, in **Exp5** we randomly selected 10 sets of state observation and agent available domain knowledge from different game episodes in the FA domain. Then we posed 27 different questions (belonging to the four types

Table 1: Precision and recall of the learned axioms in FA (Exp1) and HFO (Exp2) domains.

<i>Domain</i>	<i>Precision</i>	<i>Recall</i>
FA	1.00	0.43
HFO	0.89	0.47

of questions described in Section 3.4) to the agent from each plan and recorded the precision and recall of retrieving relevant literals for constructing answers to these questions as explanations. Next, in **Exp6** we removed the axioms the agent learned in **Exp1** from its knowledge base and repeated the experiment. Additionally, we also provide execution traces as qualitative examples of the architecture’s performance in the FA domain to support **H3**.

4.2 Experiment Results

We first discuss the results of evaluating **H1**. As stated earlier, in **Exp1**, three axioms (two causal laws and one executability condition) were removed from the ad hoc agent’s knowledge in the FA domain, and the axiom learning algorithm was executed 20 times. In **Exp2**, four axioms (one causal law and three executability conditions) were removed from the ad hoc agent’s knowledge in the HFO domain, and the axiom learning algorithm was executed 20 times. We then measured the precision and recall of the agent learning these missing axioms in each run and summarized the results in Table 1. We observe high precision and average recall values in Table 1. The lower recall can be explained by the fact that only axioms with a high degree of support from the observed transitions were added to the ad hoc agent’s knowledge, i.e., a correct candidate axiom could be filtered and not added to the ad hoc agent’s knowledge if its addition is not justified by sufficient experiences of the ad hoc agent. We observed that the recall values improved with additional experiences. These results support **H1**.

Tables 2-3 summarize the results of **Exp3** and **Exp4** respectively. Compared with **Base1**, which corresponds to not using the learned axioms from **Exp1** and **Exp2**, i.e., reasoning without some key axioms, there was a significant improvement in the number of games/episodes in which the guards won in the FA domain, particularly with policies 220, 650 and 1240. Similar performance was observed in the HFO domain, with a substantial improvement in the fraction of goals scored by the offense team when the ad hoc agent included the learned axioms during reasoning and the other agents used the policies *helios*, *gliders*, *cyrus* and *axiom*. With both the built-in policy 1600 in the FA domain and the *aut* policy in the HFO domain the results are comparable with and without the learned axioms; note that the learned axioms are not necessarily used in each episode. This is a good outcome and supports **H2**.

Next, we discuss the results of evaluating **H3**. Specifically, Table 4 summarizes the precision and recall values corresponding to **Exp5** that evaluated the

Table 2: Win(%) of the team of guards in the FA domain with and without the learned axioms (Exp3).

<i>Policy</i>	<i>Without(%)</i>	<i>With(%)</i>
220	81	86
650	38	46
1240	47	56
1600	21	21

Table 3: Fraction of goals scored by the offense team in HFO with and without the learned axioms (Exp4).

<i>Policy</i>	<i>Without(%)</i>	<i>With(%)</i>
Helios	30	35
Gliders	25	29
Cyrus	32	38
Axiom	14	24
Aut	19	19

Table 4: Precision and recall of retrieving relevant literals for constructing explanations in the FA domain with the learned axioms (Exp5).

<i>Question type</i>	<i>Precision</i>	<i>Recall</i>
Action justification	1.00	1.00
Contrastive	0.96	0.94
Belief justification	0.95	0.85
Counterfactual	1.00	1.00

Table 5: Precision and recall of retrieving relevant literals for constructing explanations in the FA domain without the learned axioms (Exp6).

<i>Question type</i>	<i>Precision</i>	<i>Recall</i>
Action justification	1.00	0.34
Contrastive	0.90	0.90
Belief justification	0.88	0.63
Counterfactual	1.00	1.00

ability to generate relational descriptions as explanations of the agents decisions and beliefs. In these experiments, the learned axioms were included in the knowledge used by the ad hoc agent for reasoning. Table 5 summarizes the precision and recall values from **Exp6**, where the ad hoc agent generated the explanations for different queries while the learned axioms were removed from the knowledge base. The observed higher values of precision and recall in Table 4 compared with the values in Table 5, i.e., the observed improvement in retrieving the relevant literals needed to construct the (correct) responses to the questions, provides strong support for hypothesis **H3**.

4.3 Execution Trace

This section provides some execution traces illustrating some of the capabilities of our architecture. Consider, for example, the scenario in Figure 5, in which an ad hoc agent in grid (3,13) at time step 0, has the goal of shooting an attacker in grid (9,8). The plan generated by the ad hoc agent was:

occurs(move(learner, 4, 13), 1),
occurs(move(learner, 5, 13), 2),
occurs(move(learner, 6, 13), 3),
occurs(move(learner, 7, 13), 4),
occurs(shoot(learner, attacker1), 5)

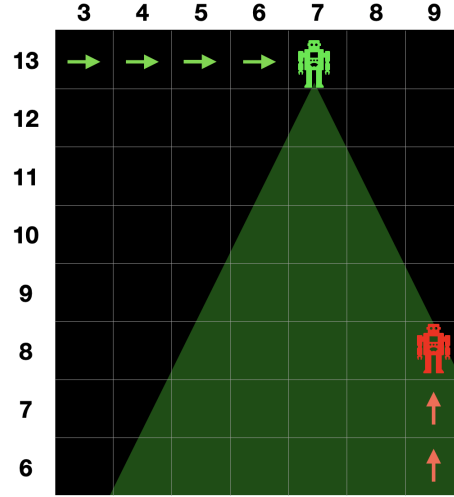


Fig. 5: Part of the domain showing the ad hoc guard agent (green) moving to track and shoot an attacker (red).

As an example of providing relational descriptions of decisions and beliefs, consider an exchange with the ad hoc agent after it executed this plan successfully;

- **Human:** “Why did you move to (7,13) in step 4 ?”
- **Ad hoc Agent:** “Because attacker1 was not in range for shooting and my goal was to shoot the attacker1 in step 5”.

This response was generated using the approach in Section 3.4. For example, since the *shoot* action occurred immediately after *move*, the relevant axiom identified included:

$$\neg occurs(shoot(R, A), I) \leftarrow \neg holds(in_range(R, A), I) \quad (7)$$

Next, the ad hoc agent explored the related answer sets and identified that ground literal $in_range(learner, attacker1)$ was present in step 5 but not in step 4. This literal was then selected to construct the answer (above) justifying the action execution.

- **Human:** “Why did you not shoot attacker1 in step 2? ”
- **Ad hoc Agent:** “Because attacker1 was not in the shooting range at step 2 since it had moved towards fort in step 0 and step 1.”

The agent first tried to identify axioms with action *shoot* in the head, e.g., Statement 7 in this scenario. Next the agent ground the body of the axiom and verified whether it was included in the answer set. Since the $\neg in_range(learner, attacker1)$ was valid in step 2, the agent identified it as a literal that prevented it from considering the shooting action at step 2. Since

the location of *attacker1* is also a key factor here, the agent also included the immediate actions of the attacker1 when generating its response (above).

In a scenario where the agent does not find any preconditions that prevents it from executing an action mentioned in a contrastive question, recall that it will still compare the cost of its plans including and excluding that particular action. An example for such a scenario is shown below.

- **Human:** “Why did you not move to (3,13) in step 2? ”
- **Ad hoc Agent:** “Because that action increases the cost of the plan; new plan cost = 8, old plan cost = 7.”

In this scenario, the ad hoc agent tried to create a plan with the suggested action. Then it computed the cost for the new plan and compared it with the cost of its original plan. Since the new plan’s cost was higher than that of the original plan, this information was used to justify its action choice and to generate the answer (above) to the question posed.

- **Human:** “Why did you believe attacker1 was in shooting range at step 5? ”
- **Ad hoc Agent:** “Because I observed that the attacker1 was located in 9,9 in step 5 and I was in 7,13 facing south.”

When probed to expand on this answer, the ad hoc agent also mentioned that ”This further satisfied the conditions $Y2 - Y1 \leq 5, Y2 - Y1 \geq 4, X2 - 2 \leq X1, X1 \leq X2 + 2$.”

When posed with a question about its beliefs, the ad hoc agent first identified the valid axioms (e.g., state constraints) that could influence the belief:

$$\begin{aligned}
 \text{holds}(\text{in_range}(L, A), I) \leftarrow \\
 & \text{holds}(\text{in}(L, X2, Y2), I), \\
 & \text{holds}(\text{agent_in}(A, X1, Y1), I), \\
 & \text{holds}(\text{face}(L, \text{south}), I), \\
 & Y2 - Y1 \leq 5, Y2 - Y1 \geq 4, \\
 & X2 - 2 \leq X1, X1 \leq X2 + 2.
 \end{aligned}$$

By grounding and verifying that the relevant literals are available in the answer set, the ad hoc agent generated the explanation described above. This answer demonstrates the agent’s ability to answer questions regarding its beliefs. A similar process can be used to trace the evolution of beliefs over multiple time steps. The ad hoc agent can also be asked about hypothetical situations, including those involving other agents, through counterfactual questions.

- **Human:** “What do you think the attacker will do in step 2? ”
- **Ad hoc Agent:** “Attacker will move up in step 2.”
- **Human:** “Why do you think the attacker will move up in step 2? ”
- **Ad hoc Agent:** “Attacker will decide to move up since all the attackers are alive and attacker is in front of the fort.”

In this scenario the agent retrieved the current state estimate stored and initialized a new FA environment. Then it simulated the environment for the desired future time steps while using the agent behaviour models to predict the actions of the other agents. Next, it went through the FF tree model predicting the attacker’s behavior, and identified rules that caused it to believe the attacker agent will ‘move up’ in step 2. This information was used to generate the responses (above) to the questions posed. A similar approach was followed to generate the explanations to queries such as:

- **Human:** “What will happened if guard1 shoots in step 2 ?”
- **Ad hoc Agent:** “If guard1 shoots in step 2, the number of alive attackers will be reduced to 1 in step 3. There will be 3 guards alive, and the nearest attacker to the fort will be attacker3.”

These results support hypothesis **H3**.

5 Conclusions

This paper described KAT, a knowledge-driven AHT architecture that supports non-monotonic logical reasoning with prior commonsense domain knowledge and predictive models of other agents’ behaviors that are learned and revised rapidly using heuristic methods. In this paper, we focused on describing KAT’s ability to incrementally learn and revise previously unknown axioms that govern action and change in the domain, enabling an ad hoc agent to better adapt to the environment. Moreover, we described KAT’s ability to provide transparency by generating on-demand relational descriptions of the ad hoc agent’s (and other agents) decisions in response to different types of questions. In the future, we will explore scenarios with multiple ad hoc agents, investigate scalability of our architecture to more complex domains, and use our architecture on physical robots in AHT settings.

References

1. Anjomshoae, S., Najjar, A., Calvaresi, D., Framling, K.: Explainable agents and robots: Results from a systematic literature review. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Montreal, Canada (2019)
2. Balai, E., Gelfond, M., Zhang, Y.: Towards Answer Set Programming with Sorts. In: International Conference on Logic Programming and Nonmonotonic Reasoning (2013)
3. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning (2003)
4. Barrett, S., Rosenfeld, A., Kraus, S., Stone, P.: Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence* **242**, 132–171 (2017)
5. Barrett, S., Stone, P., Kraus, S., Rosenfeld, A.: Teamwork with limited knowledge of teammates. In: AAAI Conference on Artificial Intelligence. vol. 27, pp. 102–108 (2013)

6. Bowling, M., McCracken, P.: Coordination and adaptation in impromptu teams. In: National Conference on Artificial Intelligence. p. 53–58 (2005)
7. Chen, S., Andrejczuk, E., Cao, Z., Zhang, J.: AATEAM: Achieving the ad hoc teamwork by employing the attention mechanism. In: AAAI Conference on Artificial Intelligence. pp. 7095–7102 (Apr 2020)
8. Deka, A., Sycara, K.: Natural emergence of heterogeneous strategies in artificially intelligent competitive teams. In: Tan, Y., Shi, Y. (eds.) Advances in Swarm Intelligence. pp. 13–25. Springer International Publishing, Cham (2021)
9. Dodamegama, H., Sridharan, M.: Back to the Future: Toward a Hybrid Architecture for Ad Hoc Teamwork. In: AAAI Conference on Artificial Intelligence (February 7-14, 2023)
10. Dodamegama, H., Sridharan, M.: Code (2023), <https://github.com/hharithaki/KAT>
11. Dodamegama, H., Sridharan, M.: Knowledge-based reasoning and learning under partial observability in ad hoc teamwork. *Theory and Practice of Logic Programming* **23**(4), 696–714 (2023). <https://doi.org/10.1017/S1471068423000091>
12. Fox, M., Long, D., Magazzeni, D.: Explainable Planning. In: IJCAI Workshop on Explainable AI (2017)
13. Gelfond, M., Incelezan, D.: Some Properties of System Descriptions of *AL_d*. *Applied Non-Classical Logics, Special Issue on Equilibrium Logic and ASP* **23**(1-2), 105–120 (2013)
14. Gigerenzer, G.: What is Bounded Rationality? In: *Routledge Handbook of Bounded Rationality*. Routledge (2020)
15. Gigerenzer, G., Gaissmaier, W.: Heuristic Decision Making. *Annual Review of Psychology* **62** (2011)
16. Hausknecht, M., Mupparaju, P., Subramanian, S., Kalyanakrishnan, S., Stone, P.: Half field offense: An environment for multiagent learning and ad hoc teamwork. In: AAMAS Adaptive Learning Agents Workshop (2016)
17. Katsikopoulos, K., Simsek, O., Buckmann, M., Gigerenzer, G.: *Classification in the Wild: The Science and Art of Transparent Decision Making*. MIT Press (February 2021)
18. Law, M., Russo, A., Broda, K.: The ILASP system for inductive learning of answer set programs. *CoRR* **abs/2005.00904** (2020), <https://arxiv.org/abs/2005.00904>
19. Miller, T.: Explanations in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence* **267**, 1–38 (February 2019)
20. Mirsky, R., Carlucho, I., Rahman, A., Fosong, E., Macke, W., Sridharan, M., Stone, P., Albrecht, S.: A Survey of Ad Hoc Teamwork: Definitions, Methods, and Open Problems. In: *European Conference on Multiagent Systems* (2022)
21. Mota, T., Sridharan, M., Leonardis, A.: Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics. *Springer Nature CS* **2**(242) (2021)
22. Rahman, M.A., Hopner, N., Christianos, F., Albrecht, S.V.: Towards open ad hoc teamwork using graph-based policy learning. In: *International Conference on Machine Learning*. pp. 8776–8786 (18–24 Jul 2021)
23. Ravula, M., Alkoby, S., Stone, P.: Ad hoc teamwork with behavior switching agents. In: *International Joint Conference on Artificial Intelligence* (7 2019)
24. Santos, P.M., Ribeiro, J.G., Sardinha, A., Melo, F.S.: Ad hoc teamwork in the presence of non-stationary teammates. In: Marreiros, G., Melo, F.S., Lau, N., Lopes Cardoso, H., Reis, L.P. (eds.) *Progress in Artificial Intelligence*. Springer International (2021)

25. Sridharan, M., Meadows, B.: Towards a theory of explanations for human-robot collaboration. *KI - Künstliche Intelligenz* **33**, 1–12 (09 2019). <https://doi.org/10.1007/s13218-019-00616-y>
26. Sridharan, M., Meadows, B., Gomez, R.: What can i not do? towards an architecture for reasoning about and learning affordances. *Proceedings of the International Conference on Automated Planning and Scheduling* **27**(1), 461–469 (Jun 2017). <https://doi.org/10.1609/icaps.v27i1.13852>, <https://ojs.aaai.org/index.php/ICAPS/article/view/13852>
27. Stone, P., Kaminka, G., Kraus, S., Rosenschein, J.: Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In: *AAAI Conference on Artificial Intelligence*. pp. 1504–1509 (2010)
28. Zand, J., Parker-Holder, J., Roberts, S.J.: On-the-fly strategy adaptation for ad-hoc agent coordination. In: *International Conference on Autonomous Agents and Multiagent Systems*. p. 1771–1773 (2022)
29. Zintgraf, L., Devlin, S., Ciosek, K., Whiteson, S., Hofmann, K.: Deep interactive bayesian reinforcement learning via meta-learning. In: *International Conference on Autonomous Agents and Multiagent Systems* (May 2021)