Collaborate and Explain on the Fly: Nonmonotonic Logical Reasoning and Incremental Learning for Ad Hoc Teamwork

Hasra Dodampegama, Mohan Sridharan

Intelligent Robotics Lab, School of Computer Science, University of Birmingham, UK

Abstract

This paper describes an architecture for ad hoc teamwork, i.e., to enable an agent to collaborate with other agents "on the fly". State of the art frameworks for ad hoc teamwork often 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. These models are computationally expensive to learn, lack transparency, and make it difficult to recognize and adapt to previously unseen changes. In a departure from existing work, we introduce an architecture for ad hoc teamwork that performs non-monotonic logical reasoning with prior commonsense domain knowledge and models that are learned and revised rapidly from limited examples to predict the behavior of other agents. In addition, the architecture enables the agent to provide relational descriptions as on-demand explanations of its decisions and beliefs in response to different types of questions. We evaluate the architecture's capabilities in two benchmark multiagent collaboration domains: Fort Attack and Half field Offense, in comparison with data-driven and knowledge-driven baselines.

Keywords

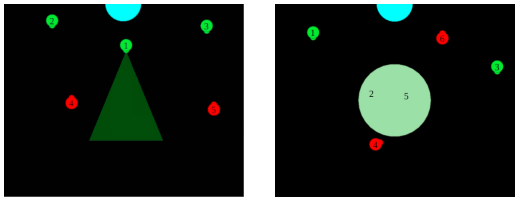
Knowledge representation and reasoning, Non-monotonic logical reasoning, Ad hoc teamwork, Multi-agent systems

1. Introduction

Ad hoc teamwork (AHT) refers to the problem of enabling an agent to collaborate with previously unknown teammates [3]. Consider a scenario from the simulated multiagent domain *Fort Attack* (FA, Figure 1a), with a team of guards defending a fort from attackers [4], or *Half Field Offense* (HFO, Figure 2), in which a team of offense agents has to score against a team of defenders [5]. Agents in these domains have limited information about each other and no prior experience of working as a team. They may also have to operate under partial observability (Figure 1b) and restricted communication. These conditions are representative of practical applications such as disaster rescue and surveillance that require the agents to reason with prior knowledge and noisy observations.

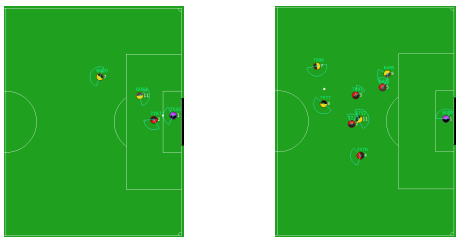
The state of the art in AHT has moved from using predetermined policies for selecting actions in specific states to methods based on a key "data-driven" component [6]. This component uses a long history of prior experiences to build probabilistic or deep network methods that model the behavior of other agents (or agent types) and optimize the behavior of the ad hoc agent. However, in practical domains, it is difficult to gather large training datasets of different situations. Also, these methods lack transparency, and make it difficult to adapt to unforeseen changes (e.g., in team composition) and to leverage commonsense domain knowledge. Unlike existing work, we follow a *cognitive systems* approach that formulates AHT as a joint reasoning and learning problem. Our knowledge-guided architecture for AHT (**KAT**) builds on the principles of *refinement* and *ecological rationality* such that the ad hoc agent:

1. Performs non-monotonic logical reasoning with commonsense domain knowledge and rapidly-learned predictive models of other agents' behav-



(a) Fully observable (b) Partially observable

Figure 1: Screenshots from the fort attack environment.



(a) Limited version (b) Full version

Figure 2: Screenshots from the half-field offense environment.

21st International Workshop on Nonmonotonic Reasoning, September 2–4, 2023, Rhodes, Greece

*Some parts of the architecture described in this paper have been described in papers published in AAAI'23 [1] and ICLP'23 [2].

hhd968@student.bham.ac.uk (H. Dodampegama); m.sridharan@bham.ac.uk (M. Sridharan)

0000-0003-2302-1501 (H. Dodampegama); 0000-0001-9922-8969 (M. Sridharan)

iors to determine its actions;
2. Uses reasoning to guide the selection of relevant behavior models and the learning of new models under partial observability; and
3. Provides on-demand relational descriptions of its decisions and beliefs as *explanations* in response to different types of questions.

Some of these contributions (e.g., #1 and #2) have been described in recent papers [1, 2]. We demonstrate that KAT supports reliable, efficient, and transparent reasoning, learning, and adaption in both FA and HFO domains while providing comparable or better performance than state of the art data-driven baselines.

2. Related Work

AHT has been researched under different names, as described in a recent survey [6]. Early work encoded specific protocols (or plays) for different scenarios, with an agent choosing specific protocols in specific states [7]. Subsequent work used sampling-based methods such as Upper Confidence bounds for Trees (UCT) to determine the ad hoc agent's action selection policy [8].

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 different *types* of teammates from a lengthy history or prior observations of similar agents or situations [9, 10]. 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 [9]. Also, attention-based deep neural networks have been used to jointly learn policies for different agent types [11], and to account for different team compositions [10]. Sequential and hierarchical variational auto-encoders have been used to model beliefs over other agents, and approximate belief inference has been meta-learned for a given prior [12]. Other work has combined learned policy methods with adversarial teammate prediction to account for changes in the agents' behavior [13] and used Convolutional Neural Networks to detect and adapt to changing teammate types [14]. Sampling strategies have also been combined with such learning methods to optimize performance [15].

Various studies have investigated communication under different AHT settings. This includes a multi-armed bandits formulation to broadcast messages to teammates while incurring a cost [16], and evaluating the cost and value of different queries in a heuristic algorithm [17].

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 strengths of knowledge-based and data-driven methods.

3. Architecture

Figure 3 provides an overview of KAT. The 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 (partial) observations of the domain state, and they independently determine and execute their individual actions in the environment. KAT's components are described using two example domains.

Example Domain 1. *[Fort Attack (FA) Domain]* Three guards are defending a fort from three attackers (Figure 1a). One guard agent is the ad hoc agent that can adapt to changes in the team and domain. 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 in its shooting range. The environment has four kinds of built-in policies for guards and attackers (see Section 4.1). The original FA domain is fully observable, i.e., each agent can observe the state of other agents at each step. We simulate partial observability by creating a forest region (Figure 1b); any agent in this region is hidden from others except the ad hoc agent.

Example Domain 2. *[Half Field Offense (HFO) Domain]* An offense team is trying to score a goal against a defense team in this simulated 2D soccer domain [5]. There are two versions: (i) *limited*: two offense agents against two defense agents (including goalkeeper); and (ii) *full*: four offense agents against five defense agents (including goalkeeper). Our ad hoc agent is one of the offense agents. 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. Similar to prior AHT methods, agents other than the ad hoc agent are selected for each episode 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. 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.

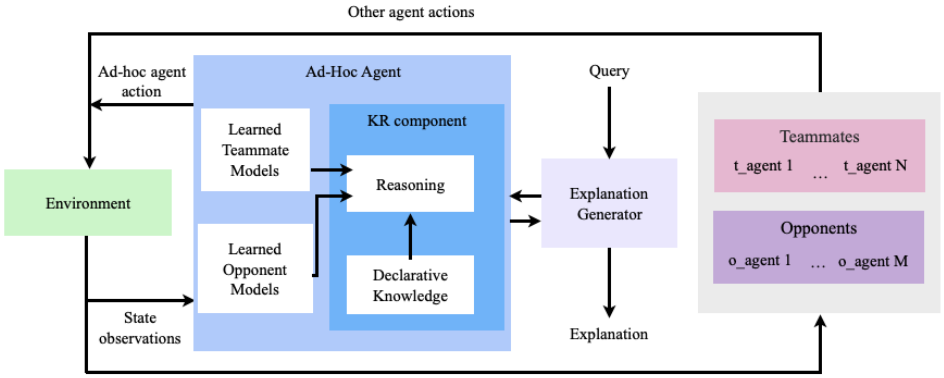


Figure 3: Architecture combines complementary strengths of knowledge-based and data-driven reasoning and learning.

Prior commonsense knowledge in above domains includes relational descriptions of some domain attributes (*e.g.,* location information), agent attributes (*e.g.,* 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. Representation and Reasoning

Any domain's transition diagram in KAT is described using an extension of the action language $\mathcal{AL}_d$ [18]. 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, *actions*, *statics*, i.e., domain attributes whose values cannot be changed by actions, and *fluents*, i.e., attributes whose values can be changed by actions. Basic sorts of the HFO domain includes ad_hoc_agent, $external\_agent$, $agent$, $offense\_agent$, $defense\_agent$, x_val, y_val, and $step$ for temporal reasoning. Sorts such as $offense\_agent$, $defense\_agent$ can be subsorts of others (*e.g.,*$external\_agent$). Statics in HFO domain include relations such as $next\_to(x\_val, y\_val, x\_val, y\_val)$ which describe the relative arrangement of places. Fluents include *inertial* fluents that obey the laws of inertia and can be changed by actions, and *defined* fluents that do not obey inertia laws and cannot be directly changed by actions, e.g., the following inertial fluents describe the location of the ad hoc agent and the ball, and which agent has the control of the ball at the current step.

$$loc(ad\_hoc\_agent, x\_val, y\_val)$$
$$ball\_loc(x\_val, y\_val),\ \ has\_ball(agent)$$

Defined fluents of the HFO domain describe the location of other agents, whether an opponent is close to any of the offense team members, and whether the ad hoc agent is too far from the goal. Actions of the domain include:

$$move(ad\_hoc\_agent, x\_val, y\_val)$$
$$dribble(ad\_hoc\_agent, x\_val, y\_val)$$
$$pass(ad\_hoc\_agent, offense\_agent)$$

which encodes the ad hoc agent's ability to move to a location, dribble the ball, and pass the ball to a teammate. The domain dynamics are described in $\mathcal{D}$ using three types of axioms: *causal law*, *state constraint* and *executablility condition*. Examples in HFO include:

$$move(R, X, Y) \textbf{ causes } loc(R, X, Y) \quad (1a)$$
$$dribble(R, X, Y) \textbf{ causes } ball\_loc(X, Y) \quad (1b)$$
$$\neg has\_ball(A1) \textbf{ if } has\_ball(A2),\ A1 \neq A2 \quad (1c)$$
$$\textbf{impossible } kick\_goal(R) \textbf{ if } far\_goal(R) \quad (1d)$$

Statements 1(a-b) are causal laws that state that moving (dribbling) to a place changes the location of the agent (ball) to that place. Statement 1(c) is a state constraint that implies only one agent can control the ball at any time. Statement 1(d) is an executability condition that prevents the consideration of an action kicking toward the goal if the ad hoc agent is too far from the goal.

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 all but a few exceptional circumstances.

To reason with knowledge, the domain description is automatically translated to a program $\Pi(\mathcal{D}, \mathcal{H})$ in CR-Prolog [19], 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; this ability to revise previously held conclusions is essential in domains such

as AHT. $\Pi(\mathcal{D}, \mathcal{H})$ includes statements from $\mathcal{D}$ and $\mathcal{H}$, relations $holds(fluent, step)$ and $occurs(action, step)$ to imply that a fluent is true and an action is part of a plan at a time step, inertia axioms, reality check axioms, closed world assumptions for defined fluents and actions, and helper axioms that define goals and drive planning and diagnosis. Each default statement is also matched with a CR rule that allows the agent associated with Π to assume that the default statement does not hold true; this CR rule is only triggered under exceptional circumstances, e.g., to restore consistency. More broadly, this ability to revise previously held conclusions is essential in practical multiagent collaboration domains in which agents often have to reason with incomplete knowledge and noisy observations.

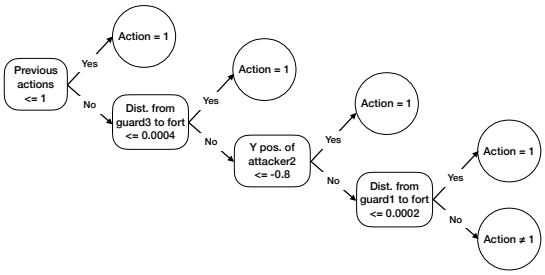
Once the program Π is constructed, all reasoning tasks (e.g., planning, diagnostics, and inference) are reduced to computing *answer sets* of Π. The ad hoc agent can prioritize different goals at different times,*e.g.,* score a goal when it has control of the ball, or position itself efficiently to receive the ball. It automatically selects the goal based on current state and considers the cost of different actions to compute a plan that minimizes the cost for achieving the goal.

We use the SPARC system [20] to write and solve CR-Prolog programs. Example programs for FA and HFO domains are in our repository [21]. For computational efficiency, our programs build on prior work in our group to represent and reason at two tightly-coupled resolutions—see paper [22] for details.

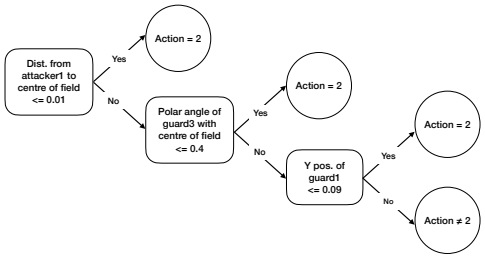
3.2. Learning Agent Behavior Models

State of the art AHT methods try to optimize performance by training models offline with large amounts of (e.g., few hundred-thousand or million) training data from different situations. However, it is not feasible to collect such labeled data for various situations in complex domains. KAT focuses on choosing relevant attributes and learning models of the behavior of other agents from limited training data (*e.g.,* 5-10K) while supporting rapid, incremental updates and accurate predictions.

To learn the predictive models, we use the *Ecological Rationality* (ER) approach, which is based on Herb Simon's definition of *Bounded Rationality* [23], and the algorithmic theory of heuristics [23, 24]. ER studies decision making under true uncertainty (*i.e.,* in open worlds), characterizes behavior as a function of the internal (cognitive) processes and environment, and focuses on *adaptive satisficing*. Also, heuristic methods (*e.g.,* one-reason, lexicographic) are viewed as a strategy to ignore part of the information in order to make decisions more quickly, frugally, and/or accurately than complex methods, experimentally choosing the method that best leverages domain structure [24]. 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 binary class label and the number of leaves is limited by number of attributes [25]. Individual FF trees learned for an attacker and a guard in the FA domain are shown in Figures 4b- 4a.



(a) FF tree in the ensemble for a guard in the FA domain.



(b) FF tree in the ensemble for an attacker in the FA domain.

Figure 4: Examples of FF trees in the FA domain.

Unlike state of the art AHT methods, predictive models in KAT can be learned and revised rapidly. Specifically, KAT enables the ad hoc agent to automatically revise existing models, switch between existing models, or learn new models. The ad hoc agent uses Algorithm 1 to select the appropriate models for the other agents by periodically comparing the existing models' predictions with the observed action choices of each agent over a sliding window of steps; in Algorithm 1, we limit this window to size of 1 (Lines 4-5). It then uses a graded strategy to penalize the error in orientation less than the error in location (Lines 6-7), selecting the models whose predictions best matches the observations for subsequent use (Line 10). If none of the models provide a good match over multiple steps, the learning of a new model is triggered.

3.3. Control Loop

The overall control loop of KAT is described in Algorithm 2. The initial setup takes as input the ad hoc agent's prior knowledge and the current set of learned behavior models of agent types ($\mathcal{M}$), and the policies $\mathcal{P}$ that gov-

Algorithm 1: model_selection

Input: $\mathcal{A}$: other agents; $\mathcal{M}$: subset of behavior models; $\{a_{act}\}$, $\{a_{pred}\}$: actual and predicted actions of each agent in current step of game; $scores$: initial values (100) for each agent-model combination.
Output: model: selected model for each agent.

1 **for** $i = 0$ **to** $\mathcal{A}$ **do**
2 **for** $m = 0$ **to** $\mathcal{M}$ **do**
3 **if** $a_{pred}[i, m] \neq a_{act}[i]$ **then**
4 $l_{act}, o_{act} \leftarrow$ actual_pose(a_{act})
5 $l_{pred}, o_{pred} \leftarrow$ predicted_pose(a_{pred})
6 $penalty \leftarrow abs(l_{act} - l_{pred}) + abs(o_{act} - o_{pred})/10$
7 $scores[i, m] = scores[i, m] - penalty$
8 **end**
9 **end**
10 model[i] = select_model($\mathcal{M}$, scores[i, *])
11 **end**

Algorithm 2: Control Loop of Architecture

Input: N: number of games; $\Pi(\mathcal{D}, \mathcal{H})$: core ASP program, $\mathcal{M}$: behavior models of other agents; $\mathcal{P}$: other agents' policies; $\mathcal{A}$: other agents
Output: game_stats: statistics of games

1 Create environment, load $\mathcal{P}$, initialize environment
2 **for** $i = 0$ **to** $N - 1$ **do**
3 $\mathbf{s} \leftarrow$ state of environment
4 **while** $\neg\, game\_over(\mathbf{s})$ **do**
5 $\mathbf{a}_o \leftarrow$ other_agents_action($\mathbf{s}, \mathcal{P}$)
6 $a_{ah}, a_{pred} \leftarrow$ adhoc_agent_action($\mathbf{s}, \Pi, \mathcal{M}$)
7 $\mathbf{a} = \mathbf{a}_o \cup a_{ah}$
8 $\mathbf{s}'$ = execute($\mathbf{s}, \mathbf{a}$)
9 model_selection($\mathcal{A}$,$\mathcal{M}$, $\mathbf{a}_o$, a_{pred})
10 **if** $game\_over(\mathbf{s}')$ **then**
11 update(game_stats)
12 initialize environment
13 **else**
14 $\mathbf{s} = \mathbf{s}'$
15 **end**
16 **end**
17 **end**
18 **return** game_stats

Algorithm 3: adhoc_agent_action

Input: $\mathbf{s}$, $\Pi(\mathcal{D}, \mathcal{H})$, $\mathcal{M}$
Output: a

1 $a_{pred} \leftarrow$ action_predictions($\mathcal{M}$)
2 $\mathbf{s}' \leftarrow$ simulate_effects(a_{pred})
3 zones $\leftarrow$ compute_relevance($\mathbf{s}, \mathbf{s}'$)
4 ASP_program $\leftarrow$ construct_program($\mathbf{s}, \Pi$, zones)
5 answer_set $\leftarrow$ SPARC(ASP_program)
6 $a \leftarrow$ next_action(answer_set)
7 **return** a, a_{pred}

ern the other agents' action choices (Line 1, unknown to the ad hoc agent). In each episode of a game, other agents' actions are based on $\mathcal{P}$ (Line 5), and the ad hoc agent computes an action by reasoning with domain knowledge and $\mathcal{M}$ (Line 6). These actions are executed in the simulated environment to compute the updated state (Line 8). Then the actual and predicted actions of the other agents are used by the model selection Algorithm 1 to incrementally revise $\mathcal{M}$ (Line 9). Finally, the updated state is used for the next step (Lines 13-15). This process continues until the game ends; the related statistics are stored before moving to next game (Lines 10-12).

Algorithm 3 describes how the ad hoc agent determines its action choice in Line 6 of Algorithm 2. It uses $\mathcal{M}$ to predict and simulate the effects of the next action (or few actions) of other agents in the current state (Lines 1-2). These effects are used to automatically identify the relevant domain regions (i.e., *zones*), axioms, level of abstraction, and goal(s) that need to be considered in the ASP program (Lines 3-4) whose answer set determines the ad hoc agent's next action (Line 6) that is returned with the predicted actions of other agents to Algorithm 2.

3.4. Partial Observability

Practical AHT domains are often partially observable, with restricted communication. To simulate such partial observability to the FA domain, we implemented a "forest" region where attackers can hide from guards other than ad hoc agent and secretly approach the fort—Figure 1b. The ad hoc agent determines when to communicate using statements such as:

$$\textbf{impossible } comm(AHA, G, AA) \textbf{ if } not\ in\_range(G, AA) \quad (2a)$$

$$in\_forest(AA) \textbf{ if } agent\_loc(AA, X, Y),\ forest(X, Y),\ not\ shot(AA) \quad (2b)$$

$$comm(AHA, G, AA) \textbf{ causes } shoots(G, AA) \quad (2c)$$

Statement 2a encodes that communication is to be only used when a hidden attacker is within the shooting range of a teammate; Statement 2b defines when an attacker is hidden; and Statement 2c describes the ad hoc agent's belief that a teammate receiving information about a

hidden attacker will shoot it, although the teammate may ignore this information. Communication actions are thus triggered only if (a) one or more attackers are hidden; and (b) one or more teammates are closer to the hidden attacker(s) than the ad hoc agent.

HFO domain provides built-in functionality to enable partial observability by limiting each agent's perception of objects (e.g., other agents, ball) to a specific viewing cone. We used this built-in ability, together with helper axioms, to simulate partial observability in the HFO domain without any communication actions.

3.5. 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; this ability has been shown to play an important role in human reasoning and learning as well [26, 27]. KAT's use of knowledge-based reasoning and simple predictive models provides the foundation to support the desired transparency in the ad hoc agent's decisions and beliefs. In addition, KAT's approach for generating the desired descriptions on-demand promotes computational efficiency. We build on prior work that demonstrated the ability to provide on-demand answers to such questions 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 [28]. Specifically, KAT's "*Explanation Generator*" (in Figure 3) generates relational descriptions in response to three types of questions identified as being important in work on explainable planning [27]:

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 a subsequent action from happening.
   - any such literal that exists in answer set at step I but not in I+1 triggered A.
2. **(Contrastive questions)** *Why did you not do action A at step I?* When asked to justify why an action was not included in the plan, the ad hoc agent will:
   - find axioms with A as head to obtain preconditions.
   - extract corresponding literals and check if they are satisfied by the answer set; each such literals 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 the axioms which has the given belief in its head(state constraints).
   - 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 [28].

The selected 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. This approach can also provide on-demand explanations of decisions and beliefs during planning and execution.

4. Experimental Setup and Results

We evaluated the following hypotheses about KAT:

- **H1:** Performance is comparable or better than state of the art baselines with much less training;
- **H2:** Enables adaptation to unforeseen changes in team composition under full and partial observability (with limited communication);
- **H3:** Supports generation of relational descriptions of the ad hoc agent's decisions and beliefs.

H1 was evaluated in both domains (FA, HFO) under full observability. For H2, we considered partial observability in both domains, and explored limited communication in the FA domain; H3 was evaluated in the FA domain. As performance measures, we used the team of guards' win percentage and shooting accuracy (i.e., fraction of times shooting eliminates an attacker) in the FA domain, and the fraction of the goals scored by offense team in the HFO domain. In both domains, we also measured the accuracy of the predictive models. Further details of experiments and baselines are provided below.

4.1. Experimental Setup

In **FA domain**, we used two kinds of policies for the ad hoc agent's teammates and attackers: hand-crafted and built-in. Hand-crafted policies were simple strategies that mimic an agent's basic expected behavior. We used two sets of hand-crafted policies: (**Policy1**) guards stay close to the fort and try to shoot attackers, while attackers

spread and approach fort; (**Policy2**) guards and attackers spread and shoot opponents. Four built-in policies were provided with the FA domain:

- **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 try to sneak in from all sides.
- **Policy1240:** guards spread and shoot the attackers; attackers sneak 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.

These are based on graph neural networks trained using several hundred-thousand training examples.

We evaluated KAT in two sets of experiments; **Exp1**, in which other agents followed the hand-crafted policies; and **Exp2**, in which other agents followed the built-in policies. To simulate training from limited examples, we collected state observations and action choices of other agents by running the hand-crafted policies, using only 10000 examples to train the ensemble of FF trees for different agent types. The learned behavior prediction models were used by the ad hoc agent to predict the actions of other agents, including when the other agents used the previously unseen built-in policies.

For experiments under full observability, each agent other than our ad hoc agent was assigned a policy selected randomly from the available policies (described above). The baselines for this experiment were: (i) **Base1** in **Exp1** with other agents following a random mix of hand-crafted policies; (ii) **Base2** in **Exp2** with other agents following a random mix of built-in policies; and (iii) **GPL**, a state of the art AHT method based on graph neural networks [10] in **Exp3** as an extension of **Exp2**. The baselines under partial observability conditions in the FA domain were: (i) **Base3** in **Exp1**, in which other agents followed hand-crafted policies and ad hoc agent did not use any communication; and (ii) **Base4** in **Exp2**, in which other agents followed built-in policies and the ad hoc agent did not use any communication. For each experiment, we used 150 episodes and results were tested for statistical significance. For GPL, we took the average results from their supplementary material.

In the **HFO domain**, we used six external agent teams from 2013 RoboCup simulation competition; ad hoc agent's teammates were selected from *helios, gliders, cyrus, axiom and aut*, and the opponents were based on *agent2d* team. We deployed these agent teams in the HFO domain and collected state observations from only 300 game episodes. Since the actions of the other agents were not directly observable, we computed them from the observed state transitions. We used this limited data to learn the models used by the ad hoc agent to predict behavior of other agents.

Table 1
Average % of episodes in which guards won with handcrafted policies and learned agent models (Exp1). Guards were more successful when one of them was our ad hoc agent.

With ad-hoc agent	Without ad-hoc agent (Base1)
73%	72%

Table 2
Average % of episodes in which guards won with previously unseen built-in policies (Exp2). Guards were more successful when one of them was our ad hoc agent.

With ad-hoc agent	Without ad-hoc agent (Base2)
55%	35%

We used two sets of experiments to measure KAT's performance under full observability: (i) **Exp4**: limited version with two offense players (including ad hoc agent) against two defense agents (including goalkeeper); and (ii) **Exp5**: full version with four offense players (including ad hoc agent) against five defense agents (including goalkeeper). In **Exp6** and **Exp7**, we evaluated KAT's performance under partial observability in the limited and full versions (respectively). In each of these experiments our ad hoc agent replaced one of the offense team agents and all other offense agents were based on one external team at a time. As baselines for **Exp4-Exp5** we used **Base5**, an ad hoc agent architecture that only uses non-monotonic logical reasoning with prior knowledge without any behavior prediction models, and recent state of the art AHT methods PPAS [13] and PLASTIC [9]. For **Exp6-Exp7**, we used the external agent teams from RoboCup as baselines. For each experiment, we used 1000 episodes and tested results for statistical significance.

4.2. Experiment Results

Table 1 summarizes the results of **Exp1** in the FA domain. The number of episodes in which the guards won was higher when one of the guards was our ad hoc agent in comparison with **Base1** in which all agents used hand-crafted policies. Table 2 summarizes the results of **Exp2** in which all agents other than the ad hoc agent used the built-in policies. When the team of guards included our ad hoc agent, they won a higher fraction of the episodes compared with **Base2**. These results demonstrated our ad hoc agent's ability to adapt to unforeseen other agents using the built-in policies, based on online revision of the behavior models learned from the hand-crafted policies. These results support hypotheses **H1** and **H2**.

Table 3 shows the prediction accuracy of the (ensemble of FF trees) models learned from hand-crafted poli-

Table 3
Accuracy of learned behavior prediction models in FA domain.

Agent Model	Accuracy
Guard type 1	85.5%
Guard type 2	60.0%
Attacker type 1	86.9%
Attacker type 2	85.2%

Table 4
Average shooting accuracy % of the ad hoc agent with previously unseen built-in policies (Exp3).

Ad hoc agent	Base2	GPL
34%	10%	30%

Table 5
Wins (%) for team of guards with hand-crafted policies (**Exp1**). Communication addresses partial observability.

Policy	With Comm. (%)	Without Comm. (%, **Base3**)
Policy1	73	58
Policy2	19	8

cies. The prediction accuracy ranges from $60 - 87\%$, i.e., these models were not perfect. However, when the ad hoc agent reasons with prior knowledge and these models, the performance of the team of guards was comparable or better than baselines without an ad hoc agent. These results demonstrate the benefits of reasoning and learning guiding each other and further support **H2**.

Table 4 shows that our ad hoc agent's shooting accuracy was higher than that of a Base2 guard agent and the GPL agent, both of whom used policies based on orders of magnitude larger number of training samples.

Next, the results of **Exp1** under partial observability (in FA domain) are summarized in Table 5. We grouped the results according to the policy used. When the ad hoc agent used the communication actions, the fraction of games won by the guards was substantially higher than when the ad hoc agent did not use communication actions (**Base3**). **Policy2** is a particularly challenging scenario (e.g., guards and attackers shoot), which justifies the lower number of games won by the guards. Table 6 summarizes the results from **Exp2** where other agents used built-in policies. We observed that for policies 650, 1240 or 1600, the fraction of games won by the guards was comparable or higher than that of **Base4** that did not use any communication actions. For policy 220, the observed performance was slightly lower than the baseline. Partial observability and communication strategies did not contribute significantly to the outcome with policy 220 because the guards place themselves in front of the fort and shoot continuously. These results support **H2**.

The results of experiments **Exp4-Exp5** in the HFO

Table 6
Wins (%) for team of guards with built-in policies (**Exp2**). Communication addresses partial observability.

Policy	With Comm. (%)	Without Comm. (%, **Base4**)
Policy220	79	85
Policy650	42	41
Policy1240	46	43
Policy1600	18	17

domain are described in Table 7. In the limited version of the game, the fraction of goals scored by the offense team was higher when the ad hoc agent used KAT than when the ad hoc agent used the logical reasoning baseline (i.e., without any learned behavior prediction models), and comparable with offense team's performance using state of the art data-driven baselines that required orders of magnitude more training examples and did not support reasoning with prior domain knowledge. In the full version of the game, the team of offense agents was able to score a much higher fraction of goals when the team included an hoc agent (using KAT) than when the agents used the baselines. This indicates that by leveraging the interplay between representation, reasoning, and learning, KAT was able to improve performance substantially, strongly supporting hypotheses **H1** and **H2**.

The prediction accuracy of the learned behavior models in the HFO domain for the limited and full versions of the game are summarized in Tables 8 and 9 respectively. Recall that the models were learned using data from only 300 episodes for each external agent type, i.e., orders of magnitude fewer examples than the several hundred-thousand used by the data-driven baselines. The prediction accuracy varied for different agent types but these models could be learned and revised rapidly and resulted in good performance when the ad hoc agent also reasoned with prior knowledge.

Table 10 summarizes the results of evaluating KAT under partial observability in **Exp6-Exp7** in the HFO domain. The goals scored by the offense team with our ad hoc agent is slightly lower than that of the external agent teams. However this difference was not significant and mainly due to noise, e.g., in the perceived angle to the goal. The ability to provide performance comparable with teams whose training datasets were orders of magnitude larger strongly supports **H2**.

Additional results and videos, particularly those involving unexpected changes in the number and type of agents, are in our open-source repository [21].

4.3. Execution Trace

In the scenario in Figure 5, an ad hoc agent in grid (2,14) at time step 0, has the goal of shooting an attacker in grid

Table 7
Fraction of goals scored by the offense team in HFO domain in the limited version(2v2) and full version(4v5).

Version	KAT (%)	Logical Reasoner (%)	PPAS (%)	PLASTIC (%)
Limited (2v2)	79	67	80	80
Full (4v5)	30	26	20	20

Table 8
Prediction accuracy of the learned agent behavior models in limited (2v2) version of the HFO domain (**Exp4**).

Agent Type	Accuracy (%)
Helios	78.2
Gliders	83.2
Cyrus	69.5
Aut	72.4
Axiom	76.2
Agent2D	79.8

Table 9
Prediction accuracy of the learned agent behavior models in full (4v5) version of the HFO domain (**Exp5**).

Agent Type	Accuracy (%)
Helios	86.0
Gliders	66.4
Cyrus	77.6
Aut	67.7
Axiom	73.6
Agent2D	71.9

Table 10
Goals scored by offense team in HFO domain under partial observability. KAT's performance comparable with baseline that had no ad hoc agents in the team.

Version	KAT (%)	Original Team (%)
Limited (2v2)	71	76
Full (4v5)	18	20

(8,7). The plan generated by the ad hoc agent was:

$$occurs(move(learner, 3, 14), 0)$$
$$occurs(move(learner, 4, 14), 1)$$
$$occurs(move(learner, 5, 14), 2)$$
$$occurs(shoot(learner, attacker1), 3)$$

As an example of providing relational descriptions of decisions and beliefs, consider an exchange with the ad hoc agent after it shot the attacker;

- **Human:** "Why did you move to (5,14) in step 2?"
- **Ad hoc Agent:** "Because attacker1 was not in range for shooting".
  This response was generated using the approach in Section 3.5. For example, since the *shoot* action occurred immediately after *move*, the relevant axioms identified included:

$$-occurs(shoot(R, A), I) \leftarrow$$
$$-holds(in\_range(R, A), I).$$

  Next, the ad hoc agent explored its answer set to check whether the grounded literal $in\_range(learner, attacker1)$ was present in step 2 and 3. Since it existed at step 2 but not in step 3, this literal was selected as an explanation for justifying the action execution.
- **Human:** "Why did you not move to grid (3,14) in time step 3? "
- **Ad hoc Agent:** "Because it increases the cost of the plan; new plan cost = 6, old plan cost = 4."
  In this scenario, the ad hoc agent tried to create a plan with the suggested action. Such a plan to achieve the goal was found but the new plan's cost was higher than the original plan. This information was included in the explanation to the contrastive question.
- **Human:** "Why did you believe the attacker was in shooting range at time step 4?"
- **Ad hoc Agent:** "Because I observed attacker1 in (8,7) and I was in (5,14) facing south and the values satisfied the conditions $Y2 - Y1 <= 8, X2 - 3 <= X1, X1 <= X2 + 3$.".
  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:

$$holds(in\_range(L, A), I) \leftarrow$$

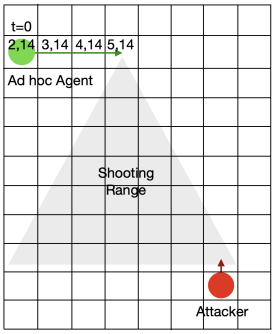


Figure 5: Part of the domain showing the ad hoc guard agent (green) moving to track and shoot an attacker (red).

$holds(in(L, X2, Y2), I),$
$holds(agent\_in(A, X1, Y1), I),$
$holds(face(L, south), I),$
$X2 - 3 <= X1,$
$X1 <= X2 + 3,$
$Y2 - Y1 <= 8.$

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 [28].

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. KAT automatically selects and uses the relevant behavior prediction models, and learns new ones when appropriate, enabling an ad hoc agent to adapt to previously unseen teammates and opponents on the fly. Moreover, KAT provides transparency by generating on-demand relational descriptions of its decisions and beliefs 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.

Acknowledgments

This work was supported in part by the US Office of Naval Research award N00014-20-1-2390. All conclusions are those of the authors alone.

References

[1] H. Dodampegama, M. Sridharan, Back to the Future: Toward a Hybrid Architecture for Ad Hoc Teamwork, in: AAAI Conference on Artificial Intelligence, 2023.

[2] H. Dodampegama, M. Sridharan, Knowledge-based Reasoning and Learning under Partial Observability in Ad Hoc Teamwork, in: International Conference on Logic Programming, 2023.

[3] P. Stone, G. Kaminka, S. Kraus, J. Rosenschein, Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination, in: AAAI Conference on Artificial Intelligence, 2010, pp. 1504–1509.

[4] A. Deka, K. Sycara, Natural emergence of heterogeneous strategies in artificially intelligent competitive teams, in: Y. Tan, Y. Shi (Eds.), Advances in Swarm Intelligence, Springer International Publishing, Cham, 2021, pp. 13–25.

[5] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, P. Stone, Half field offense: An environment for multiagent learning and ad hoc teamwork, in: AAMAS Adaptive Learning Agents Workshop, 2016.

[6] R. Mirsky, I. Carlucho, A. Rahman, E. Fosong, W. Macke, M. Sridharan, P. Stone, S. Albrecht, A Survey of Ad Hoc Teamwork: Definitions, Methods, and Open Problems, in: European Conference on Multiagent Systems, 2022.

[7] M. Bowling, P. McCracken, Coordination and adaptation in impromptu teams, in: National Conference on Artificial Intelligence, 2005, p. 53–58.

[8] S. Barrett, P. Stone, S. Kraus, A. Rosenfeld, Teamwork with limited knowledge of teammates, in: AAAI Conference on Artificial Intelligence, volume 27, 2013, pp. 102–108.

[9] S. Barrett, A. Rosenfeld, S. Kraus, P. Stone, Making friends on the fly: Cooperating with new teammates, Artificial Intelligence 242 (2017) 132–171.

[10] M. A. Rahman, N. Hopner, F. Christianos, S. V. Albrecht, Towards open ad hoc teamwork using graph-based policy learning, in: International Conference on Machine Learning, 2021, pp. 8776–8786.

[11] S. Chen, E. Andrejczuk, Z. Cao, J. Zhang, AATEAM: Achieving the ad hoc teamwork by employing the attention mechanism, in: AAAI Conference on Artificial Intelligence, 2020, pp. 7095–7102.

[12] L. Zintgraf, S. Devlin, K. Ciosek, S. Whiteson, K. Hofmann, Deep interactive bayesian reinforcement learning via meta-learning, in: International Conference on Autonomous Agents and Multiagent Systems, 2021.

[13] P. M. Santos, J. G. Ribeiro, A. Sardinha, F. S. Melo, Ad hoc teamwork in the presence of non-stationary teammates, in: G. Marreiros, F. S. Melo, N. Lau, H. Lopes Cardoso, L. P. Reis (Eds.), Progress in Artificial Intelligence, Springer International, 2021.

[14] M. Ravula, S. Alkoby, P. Stone, Ad hoc teamwork with behavior switching agents, in: International Joint Conference on Artificial Intelligence, 2019.

[15] J. Zand, J. Parker-Holder, S. J. Roberts, On-the-fly strategy adaptation for ad-hoc agent coordination, in: International Conference on Autonomous Agents and Multiagent Systems, 2022, p. 1771–1773.

[16] S. Barrett, N. Agmon, N. Hazon, S. Kraus, P. Stone,

Communicating with unknown teammates, in: European Conference on Artificial Intelligence, 2014.
[17] W. Macke, R. Mirsky, P. Stone, Expected value of communication for planning in ad hoc teamwork, in: AAAI Conference on Artificial Intelligence, 2021, pp. 11290–11298.
[18] M. Gelfond, D. Inclezan, Some Properties of System Descriptions of AL_d, Applied Non-Classical Logics, Special Issue on Equilibrium Logic and ASP 23 (2013) 105–120.
[19] M. Balduccini, M. Gelfond, Logic Programs with Consistency-Restoring Rules, in: AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning, 2003.
[20] E. Balai, M. Gelfond, Y. Zhang, Towards Answer Set Programming with Sorts, in: International Conference on Logic Programming and Nonmonotonic Reasoning, 2013.
[21] H. Dodampegama, M. Sridharan, Code, 2023. https://github.com/hharithaki/KAT.
[22] M. Sridharan, M. Gelfond, S. Zhang, J. Wyatt, REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics, Journal of Artificial Intelligence Research 65 (2019) 87–180.
[23] G. Gigerenzer, What is Bounded Rationality?, in: Routledge Handbook of Bounded Rationality, Routledge, 2020.
[24] G. Gigerenzer, W. Gaissmaier, Heuristic Decision Making, Annual Review of Psychology 62 (2011).
[25] K. Katsikopoulos, O. Simsek, M. Buckmann, G. Gigerenzer, Classification in the Wild: The Science and Art of Transparent Decision Making, MIT Press, 2021.
[26] S. Anjomshoae, A. Najjar, D. Calvaresi, K. Framling, Explainable agents and robots: Results from a systematic literature review, in: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Montreal, Canada, 2019.
[27] M. Fox, D. Long, D. Magazzeni, Explainable Planning, in: IJCAI Workshop on Explainable AI, 2017.
[28] T. Mota, M. Sridharan, A. Leonardis, Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics, Springer Nature CS 2 (2021).