Coordination in Ad Hoc Teams using Knowledge-based Reasoning and Learning (Full)

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

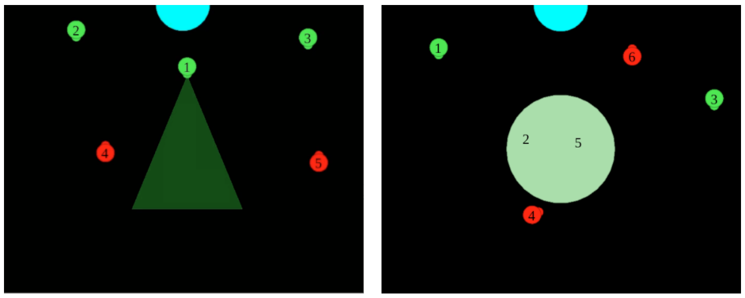
University of Birmingham, UK
hhd968@student.bham.ac.uk
m.sridharan@bham.ac.uk

Abstract. Ad hoc teamwork focuses on enabling an agent to collaborate with others without prior coordination. This is a central problem in the coordination of open multiagent systems in many practical applications. Data-driven learning methods represent the state of the art in ad hoc teamwork. They use a large labeled dataset of prior observations to model the behavior of other agent *types* and to determine the ad hoc agent's behavior. These methods are computationally expensive, lack transparency, and make it difficult to adapt to changes in team composition. Our recent work introduced an architecture that determined an ad hoc agent's behavior based on non-monotonic logical reasoning with prior commonsense domain knowledge and models that were learned from limited examples to predict other agents' behavior. In this paper, we expand the architecture's capabilities, supporting: (a) online adaptation and choice of learned models of other agents' behavior; and (b) collaboration in the presence of partial observability and limited communication. Experimental evaluation in two simulated benchmark domains for ad hoc teamwork demonstrates performance comparable or better than state of the art data-driven baselines in simple and complex scenarios, particularly in the presence of limited training data, partial observability, and changes in team composition.

Keywords: Non-monotonic logical reasoning · Ecological rationality · Ad hoc teamwork.

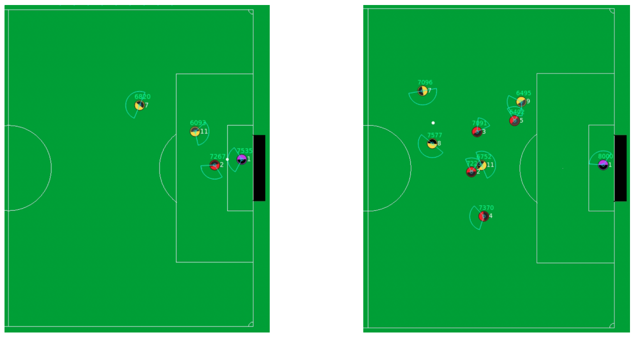
1 Introduction

Ad Hoc Teamwork (AHT) refers to the problem of enabling an agent to collaborate with previously unknown teammates toward a common goal [27]. This is a central problem in the coordination and governance of multiagent teams deployed in many practical applications such as disaster rescue and surveillance. As motivating examples, consider the simulated multiagent collaboration domain *Fort Attack* (FA, Figure 1a), where a team of guards has to protect a fort from a team of attackers [10], and the *Half Field Offense* domain (HFO, Figure 2), where a team of soccer-playing agents has to score a goal while playing



(a) Fully observable (b) Partial observable

Fig. 1: Scenarios in the *fort attack* environment: (a) guards protecting a fort from attackers; (b) domain has a region of reduced perception.



(a) Limited version (b) Full version

Fig. 2: Scenarios in the *half-field offense* environment: offense players trying to score a goal in the presence of a team of defense players in the (a) limited (2v2) setting; and (b) full (4v5) setting.

against defenders [17]. Although collaboration is beneficial and often necessary for success in such domains, agents in these domains have limited knowledge of each other and no prior experience of working as a team. Furthermore, team composition may change unexpectedly (e.g., one of the offense team players may be sent off the field for some time), each agent may only be able to observe part of the environment (Figure 1b), and the bandwidth available for communicating with other agents may be limited.

The state of the art in AHT has moved from the use of predetermined policies for selecting actions in specific states to focusing on the development of a "data-driven" component. This component uses probabilistic methods or deep network methods to model the behavior (i.e., the selection of specific actions in specific states) of other agents or agent types, and optimize the behavior of the ad hoc agent, based on a long history of prior experience. It is difficult to obtain such training examples in many practical domains and computationally expensive to build the necessary models. Also, the trained models lack transparency and make it difficult to adapt to changes, e.g., in team composition. In a departure from existing work, we pursue a *cognitive systems* approach based on the observation that AHT jointly poses representation, reasoning, and learning challenges, and that the focus on data-driven optimization makes it difficult to leverage the rich commonsense domain knowledge available in many domains. Specifically, our knowledge-driven AHT architecture (KAT) supports three capabilities:

1. Non-monotonic logical reasoning with prior commonsense domain knowledge and rapidly-learned predictive models of other agents' behaviors;
2. Use of reasoning and observations to trigger the selection of relevant agent behavior models and the learning of new models as needed; and
3. Use of reasoning to guide collaboration with teammates under partial observability and limited communication.

Our recent work provided a proof of concept demonstration of the first capability in the FA domain [11]. In this paper, we use Answer Set Prolog (ASP) for non-monotonic logical reasoning, and heuristic methods based on ecological rationality principles [15] for rapidly learning and revising the agent behavior models. We ground and evaluate our architecture's capabilities in the FA domain and the more complex HFO domain, demonstrating performance comparable or better than state of the art data-driven methods in simple and complex domains, even in the presence of partial observability and changes in team composition.

2 Related Work

There has been considerable prior work in AHT under different names for around 15 years, as described in a recent survey paper [20]. Early work used specific protocols ('plays') to define how an agent should behave in different scenarios (states) [8]. Subsequent work used sample-based methods such as Upper Confidence bounds for Trees (UCT) [7], or combined UCT with other methods to learn models from historical data and use them for online planning [28]. More recent methods have included a key data-driven component, using probabilistic, deep-network, and reinforcement learning (RL)-based methods to learn action (behavior) choice policies for different *types* of teammates from a lengthy history or prior observations of similar agents or situations [6, 21]. For example, RL methods have been used to identify and use the most useful policy (from a set of learned policies) for each situation [6], or to consider the predictions from all learned policies when selecting an ad hoc agent's actions for different types of agents [22]. Also, attention-based deep neural networks have been used to jointly learn policies for different agent types [9], and to account for different team compositions [21]. Other work has combined sampling strategies with such learning methods in an attempt to optimize performance [29]. There has also been work on using deep networks to learn sequential and hierarchical models, which are used with approximate belief inference methods [30]. In addition, researchers have explored different communication strategies for AHT, e.g., a multiagent, multi-armed bandit formulation to broadcast messages to teammates while incurring a cost [6], or assessing the cost and value of different queries in a heuristic algorithm [19]. All these methods require considerable resources (e.g., computation, memory, training examples), build opaque models, and make it difficult to adapt to unexpected changes, e.g., in team composition.

There has been considerable research in developing action languages and logics for single and multiagent domains. This includes work on developing an action language ($\mathcal{A}$) for an agent to compute cooperative actions in multiagent domains [25], and an exploration of using action language $\mathcal{C}$ for modeling benchmark multiagent domains with minimal extensions [5]. Action language $\mathcal{B}$ has also been combined with Prolog and ASP to implement a distributed multiagent planning system that supports communication in a team of collaborative agents [24]. More recent work has explored the use of $\mathcal{B}$ for planning in single agents and multiagent teams, including a distributed planning approach for non-cooperative or partially-collaborative agents using negotiations [23]. To model realistic interactions, researchers have introduced specific action types, e.g., world-altering, sensing, and communication actions, to manipulate the knowledge of agents in the domain [4]. Recent work has represented these action types in action language m$\mathcal{A}*$ while also supporting epistemic planning and dynamic awareness of action occurrences [3]. These studies have demonstrated that ASP can be used to represent and reason in multiagent domains. Our work draws on these findings to address the reasoning and learning challenges faced by an ad hoc agent that has to collaborate with teammates under conditions of partial observability and limited communication.

3 Architecture

Figure 3 is an overview of our KAT architecture. Our ad hoc agent (one member of a team) performs non-monotonic logical reasoning with prior commonsense domain knowledge, and with incrementally learned behavior models of teammate and opponent agents. At each step, valid observations of the domain state are available to all the agents, who then independently determine and execute their individual actions in the environment. The architecture's components are described using the following two example domains.

Example 1 *[Fort Attack (FA) Domain]*
There are three guards protecting a fort from three attackers (Figure 1). One guard is the ad hoc agent that can adapt to changes in the team and domain. An episode ends if: (a) guards manage to protect the fort for a period of time; (b) all attackers are eliminated; or (c) any attacker reaches the fort.

At each step, each agent can choose to move in any of the four cardinal direction with a particular velocity, turn clockwise or anticlockwise, do nothing, or shoot; any agent in the shooting range is killed. The environment provides four types of built-in policies for guards and attackers (see Section 4.1). The

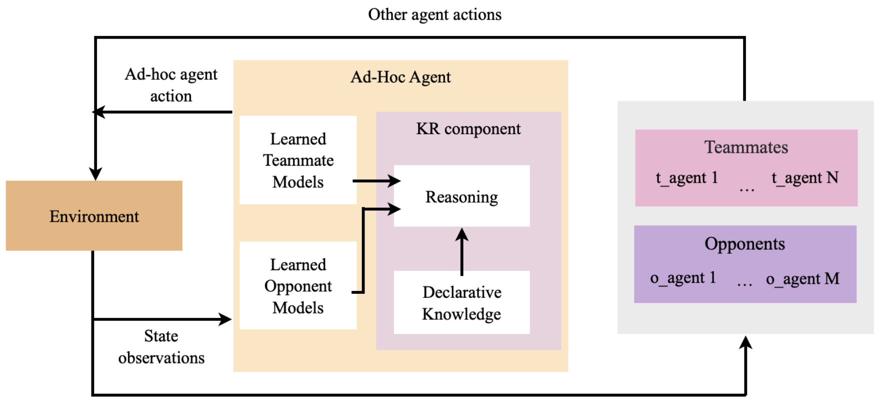


Fig. 3: Our KAT architecture combines complementary strengths of knowledge-based and data-driven heuristic reasoning and learning.

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*"; any agent in this region is hidden from others.

Example 2 *[Half Field Offense (HFO) Domain]*
This simulated 2D soccer domain is a complex benchmark for multiagent and AHT methods [17]. The ad hoc agent is a member of the offensive team that seeks to score a goal against a defensive team. An episode ends when: (a) offensive team scores a goal; (b) ball leaves field; (c) defensive team captures the ball; or (d) maximum episode length (500) is reached.

There are two version of the domain: (i) limited version: two offense agents and two defense agents (including goalkeeper); (ii) full version: four offense agents, five defense agents (including goalkeeper). Agents other than the ad hoc agent are selected from teams created in the RoboCup 2D simulation league competitions. Similar to existing AHT methods, our other offensive team agents were based on the binary files of five teams: *helios, gliders, cyrus, axiom, aut*. For defenders, we use *agent2D* agents, whose policy was derived from *helios*. The strategies of these agents were trained using data-driven (probabilistic, deep, reinforcement) learning methods. HFO supports two state space abstractions: low, high; we use the high-level feature set. Also, there are three abstractions of the action space: primitive, mid-level, and high-level; we use a combination of mid-level and high-level actions.

Prior commonsense domain knowledge in the FA domain and the HFO domain includes relational descriptions of some domain attributes (e.g., safe regions), agent attributes (e.g., location), 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 determines changes in state when any action is executed; it may also need to be revised over time.

3.1 Knowledge Representation and Reasoning

To describe the transition diagrams of our domains, we use an extension of the action language $\mathcal{AL}_d$ [13]. The domain representation consists of 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 which consists of *actions*, *statics*, i.e., domain attributes whose values cannot be changed, and *fluents*, i.e., attributes whose values can be changed by actions. For example, Σ in the HFO domain consists of basic sorts such as ad_hoc_agent, $external\_agent$, $agent$, $offense\_agent$, $defense\_agent$, x_val, y_val, and sort *step* for temporal reasoning. Sorts are arranged hierarchically, with some sorts (e.g., $offense\_agent$, $defense\_agent$,) being subsorts of other sorts ($external\_agent$). Statics in Σ are relations such as $next\_to(x\_val, y\_val, x\_val, y\_val)$ that encode the relative arrangement of locations. Σ includes *inertial* fluents that obey inertia laws and can be changed by actions, and *defined* fluents that do not obey inertia laws are not directly changed by actions. Inertial fluents in the HFO domain include:

$$\begin{aligned} &loc(ad\_hoc\_agent, x\_val, y\_val),\ ball\_loc(x\_val, y\_val), \\ &has\_ball(agent) \end{aligned} \tag{1}$$

which describes the location of the ad hoc agent, location of the ball, and the agent that has control of the ball. Defined fluents of the HFO domain include:

$$\begin{aligned} &agent\_loc(external\_agent, x\_val, y\_val), \\ &defense\_close(agent, defense\_agent),\ far\_from\_goal(ad\_hoc\_agent) \end{aligned} \tag{2}$$

which encode the location of the external (i.e., non ad hoc) agents, and describe whether a defense agent is too close to another agent, and whether the ad hoc agent is far from the goal. Next, actions in the HFO domain include:

$$\begin{aligned} &move(ad\_hoc\_agent, x\_val, y\_val),\ kick\_goal(ad\_hoc\_agent), \\ &dribble(ad\_hoc\_agent, x\_val, y\_val),\ pass(ad\_hoc\_agent, offense\_agent) \end{aligned} \tag{3}$$

which describe the ad hoc agent's ability to move to a location, kick the ball toward the goal, dribble the ball to a particular location, and and pass the ball to a teammate. Σ also has relation $holds(fluent, step)$, which indicates that a particular fluent is true at a given step, and $occurs(action, step)$, which states that a particular action occurs in a plan at a particular step.

Given Σ, axioms in $\mathcal{D}$ describe the dynamics of the domain using elements in Σ in causal laws, state constraints and executability conditions such as:

$$move(R, X, Y) \textbf{ causes } loc(R, X, Y) \tag{4a}$$

$$dribble(R, X, Y) \textbf{ causes } ball\_loc(X, Y) \tag{4b}$$

$$\neg has\_ball(A1) \textbf{ if } has\_ball(A2),\ A1 \neq A2 \tag{4c}$$

$$\textbf{impossible } shoot(R) \textbf{ if } far\_from\_goal(R) \tag{4d}$$

where Statements 4(a-b) are causal laws that imply that moving and dribbling change the ad hoc agent's and ball's location (respectively) to the desired location. Statement 4(c) is a state constraint that implies that only one agent can control the ball at any time. Statement 4(d) is an executability condition that prevents the consideration of a shooting action (during planning) if the ad hoc agent is far from the goal. Finally, the history $\mathcal{H}_c$ is a record of observations of fluents, i.e., $obs(fluent, boolean, step)$, and action executions, i.e., $hpd(action, step)$ at specific time steps. It also includes initial state defaults, i.e., statements that are initially believed to be true in all but a few exceptional circumstances.

To enable an ad hoc agent to reason with prior knowledge, the domain description in $\mathcal{AL}_d$ is automatically translated to program $\Pi(\mathcal{D}_c, \mathcal{H}_c)$ in CR-Prolog [2], an extension of ASP that supports consistency restoring (CR) rules. ASP is a declarative programming paradigm that is based on stable model semantics and represents constructs difficult to express in classical logic formalisms. It encodes concepts such as *default negation* and *epistemic disjunction*, i.e., unlike "$\neg a$" that states *a is believed to be false*, "*not a*" only implies *a is not believed to be true*, and unlike "$p \ \vee \ \neg p$" in propositional logic, "$p \ or \ \neg p$" is not tautologous. Each literal is true, false, or "unknown", and the agent only believes that which it is forced to believe. In particular, it supports *non-monotonic reasoning*, i.e., the ability to revise previously held conclusions, which is essential in AHT domains and not supported by classical first order logic. $\Pi(\mathcal{D}_c, \mathcal{H}_c)$ also includes inertia axioms, reality check axioms, closed world assumptions for defined fluents and actions, and helper axioms, e.g., to define goals and drive planning and diagnosis. All reasoning tasks can then be reduced to computing *answer sets* of Π. Each answer set is a collection of ground literals describing a possible model of the world.

The ad hoc agent may need to prioritize different goals at different times, e.g., score a goal when it has control of the ball, and position itself otherwise:

$$\begin{aligned} &goal(I) \leftarrow holds(scored\_goal, I). \\ &goal(I) \leftarrow holds(loc(ad\_hoc\_agent, X, Y), I). \end{aligned} \tag{5}$$

A suitable goal is selected and included at run-time based on current state, and the cost is minimized when computing a plan of actions for a given goal:

$$\begin{aligned} &total(S) \leftarrow S = sum\{C, A : occurs(A, I), cost(A, C)\}. \\ &\#minimize\{S@p, S : total(S)\}. \end{aligned}$$

We use the SPARC system [1] to write and solve CR-Prolog programs; examples are in our open source repository [12]. For computational efficiency, our programs build on prior work in our group to represent and reason at two formally-coupled resolutions—please see [26] for details.

3.2 Agent Models and Model Selection

In addition to prior domain knowledge, the ad hoc agent reasons with models that predict the action choices of teammates and opponents. Recall that these

Table 1: Attributes considered for models of other agents' behavior in FA domain.

Description of attribute	Number
x, y position of agent	12
distance from agent to center of field	6
agents' polar angle with center of field	6
orientation of the agent	6

Description of attribute	Number
distance from agent to fort	6
distance to nearest attacker from fort	1
number of attackers not alive	1
previous action of the agent	1

Table 2: Attributes for models of teammates' behavior in HFO domain.

Description of attribute	Number
x position of agent	4
y position of agent	4
goal opening angle	2
proximity to the nearest opponent	2
x position of the ball	1
y position of the ball	1

Table 3: Attributes for models of defense agents' behavior in HFO domain.

Description of attribute	Number
x position of agent	4
y position of agent	4
x position of the ball	1
y position of the ball	1

models need to be learned and revised rapidly from limited examples to provide sufficient accuracy. Instead of using many (e.g., several million) examples under different conditions, which is difficult to obtain in practical domains, we focused on the choice of attributes and used simple hand-crafted policies (e.g., spread and shoot in FA, pass when possible in HFO) to collect limited (e.g., 10K) training examples. Table 1 list the attributes used for predictive models in FA domain, and Tables 2 and 3 list the attributes for HFO domain.

Similar to our recent work [11], the predictive models were learned using the *Ecological Rationality* (ER) approach, which draws on insights from human cognition and builds on Herb Simon's definition of *Bounded Rationality* [15] and an algorithmic model of heuristics [16]. Unlike the focus on optimal search in state of the art AI methods, ER focuses on decision making under true uncertainty (e.g., open worlds), characterizes behavior as a function of internal (cognitive) processes and environment, and focuses on *satisficing* based on differences between observations and predictions of simple models. 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 identifying the method that best leverages domain structure. This approach has led to good performance in many practical applications [14]. In our case, behavior prediction models for teammates and opponents (in FA, HFO) were based on an ensemble of "fast and frugal" (FF) decision trees that can be learned and revised at run-time; each tree provides a binary class label and the number of leaves is limited by the number of attributes considered to build the model [18].

Algorithm 1: Model Selection

```
   Input: $\mathcal{A}$: other agents; $\mathcal{M}$: subset of behaviour models; $\{a_r\}$: actual
          action choices of agents, $\{a_p\}$: action predictions from behaviour
          models
 1 for $i = 0$ to $\mathcal{A}$ do
 2     for $m = 0$ to $\mathcal{M}$ do
 3         if $a_p \neq a_r$ then
 4             $l_r, o_r \leftarrow$ pred_location_orientation($a_p$)
 5             $l_p, o_p \leftarrow$ real_location_orientation($a_r$)
 6             $penalty \leftarrow abs(l_r - l_p) + abs(o_r - o_p)/10$
 7         end
 8         scores = scores - penalty
 9     end
10     update_model_scores($\mathcal{M}$, scores)
11 end
```

The ad hoc agent's teammates or opponents may include different types of agents, and their behavior may change over time. *Unlike our prior work, we enabled the ad hoc agent to automatically identify, reason about, and respond to such behavior changes by revising models, switching between models, or learning new models.* Existing models are revised by changing the parameters of FF trees. An instance of our model selection approach is summarized in Algorithm 1 in the context of models predicting the pose (i.e., position and orientation) of agents. Specifically, the ad hoc agent periodically compares the predictions of the existing models with the actual (i.e., observed) action choices of each agent (teammate, opponent) over a sliding window of information about the domain state and the agents' action choices. Also, the ad hoc agent has a graded strategy for computing the error, e.g. small differences in orientation are penalized less than large differences in position (Lines 4-6, Algorithm 1). An existing model is used (and revised) or a new one is learned based on the degree of match between observed and predicted action choices (Line 10, Algorithm 1).

3.3 Partial Observability and Communication

In practical multiagent AHT domains, any single agent cannot observe the entire domain. At the same time, communication is often a scarce resource. In order to explore the interplay between partial observability and communication between agents, we modified the original domains (FA, HFO). Specifically, in the FA domain, we introduced a “forest" region where attackers can hide from the view of the two guards (other than ad hoc agent), giving them an opportunity to secretly approach the fort—see Figure 1b. The ad hoc agent has visibility of the forest region; it can decide when to communicate with its teammates when needed, e.g., when: (a) one of more attackers are hidden in the forest; and (b) one of the other guards is closer to the hidden attacker(s) than the ad hoc agent. The associated reasoning can be encoded using statements such as:

$$holds(shoots(G, AA), I+1) \leftarrow occurs(communicate(AHA, G, AA), I) \tag{6a}$$

$$\begin{aligned} holds(in\_forest(AA), I) \leftarrow\ & holds(agent\_loc(AA, X, Y), I),\ forest(X, Y),\\ & not\ \ holds(shot(AA), I) \end{aligned} \tag{6b}$$

$$-\,occurs(comms(AHA, G, AA), I) \leftarrow\ \ not\ holds(in\_range(G, AA), I). \tag{6c}$$

where Statement 6(c) encodes that communication is used only when a hidden attacker is within range of a teammate; Statement 6(b) defines when an attacker is hidden in the forest; and Statement 6(a) describes the ad hoc agent's belief that a teammate receiving information about a hidden attacker will shoot it. Also, if there are multiple guards satisfying the encoded conditions, the ad hoc agent may only communicate with the guard closest to the hidden attacker(s). Note that any guard receiving the communication acts independently and may choose to ignore the information.

In the HFO domain, we represent partial observability using the builtin ability to modulate the noise in an agent's perception and actions. Specifically, each agent's perception is limited to a viewing cone, i.e., the agent is only able to sense objects (i.e., agents, ball) within this cone and objects outside this cone are not visible. Unlike the FA domain, given the use of builtin functions to define partial observability, no additional communication action was implemented in the HFO domain. We only added a few helper axioms to ensure that the ad hoc agent only considered the visible objects for reasoning.

4 Experimental setup and results

We experimentally evaluated the following hypotheses about our architecture's capabilities in the benchmark simulation domains:

H1: Our architecture's performance is comparable or better than state of the art baselines in simple and complex scenarios while using much fewer training samples;

H2: Our architecture enables adaptation to unforeseen changes in the number and type of agents (teammates and opponents); and

H3: Our architecture supports adaptation to partial observability with limited communication capabilities.

We evaluated the H1 and H2 in both FA domain and HFO domain under full observability. For H3, we considered partial observability in both FA and HFO domains, and explored limited communication in the FA domain. Each game of the FA domain consisted of three guards and three attackers, with *our ad hoc agent replacing one of the guards*. In HFO domain, each game consisted of two offense and two defense players (including one goalkeeper) in the limited version; and four offense players and five defensive agents (including one goalkeeper) in the full version. *Our ad hoc agent replaced one of the offense agents in each game* *of the HFO domain.* In the FA domain, the key performance measure was the win percentage of the guards team. In the HFO domain, the key performance measure was the fraction of games that ended with the offense team scoring a goal. In both domains, we also measured the accuracy of the predictive models. Further details of the experiments and baselines are provided below.

4.1 Experimental Setup

In the **FA domain**, we used two sets of policies for the agents other than our ad hoc agent: *hand-crafted policies* and *built-in policies*. Hand-crafted policies were constructed as simple strategies that produce basic behaviour of agents in the FA domain. Built-in policies were provided with the domain; they are based on graph neural networks trained using a large dataset of examples. We briefly describe these policies below.

Hand-Crafted Policies:

– **Policy1:** guards stay close to the fort and try to shoot the attackers, attackers spread and approach fort;
– **Policy2:** both guards and attackers spread and shoot their opponents.

Built-in Policies:

– **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 spread and move from the fort if needed; some attackers approach and shoot at the guards to divert their attention, while other attackers wait outside the guards' shooting range for the right moment to sneak in.

We evaluated the ad hoc agent in two experiments: **Exp1**, in which agents other than the ad hoc agent followed the hand-crafted policies; and **Exp2**, in which other agents followed the built-in policies. To explore the ability to learn from limited examples, we created basic behavior models in the form of FF trees from just 10000 state and action observations obtained by running the hand-crafted policies in the FA domain. Our ad hoc agent did not receive any prior experience or models of the built-in policies.

Our previous work documented the accuracy of static behavior models and the performance of a proof of concept AHT architecture in the FA domain [11]. So, in this paper, we focused on evaluating the ability to revise the predictive models and adapt to partial observability. For evaluating the model selection algorithm, we used a team of agents based on a mix of policies described earlier (under **Exp1** and **Exp2**); specifically, agents other than our ad hoc agent randomly chose a policy from the mix of available policies. The baselines for this experiment were:

- **Base1:** other agents followed a mix of hand-crafted policies; ad hoc agent did not revise behavior models or use the model selection algorithm.
- **Base2:** other agents followed a mix of hand-crafted policies; ad hoc agent used a model selection algorithm without a graded strategy to compare predicted and actual actions (binary comparison instead of Line 6 in Algorithm 1).
- **Base3:** other agents followed a mix of builtin policies; ad hoc agent did not revise models or use the model selection algorithm.
- **Base4:** other agents followed a mix of built-in policies; ad hoc agent used the model selection algorithm without a graded strategy to compare predicted and actual actions (binary comparison instead of Line 6 in Algorithm 1).

The baselines for evaluating partial observability and communication were:

- **Base5:** in **Exp1**, other agents followed hand-crafted policies and ad hoc agent did not use any communication actions.
- **Base6:** in **Exp2**, other agents followed a mix of built-in policies and the ad hoc agent did not use any communication actions.

Each experiment consisted of 150 episodes of the FA game.

In the **HFO domain**, we used six external agent teams from 2013 RoboCup simulation competition to create the ad hoc agent's teammates and opponents. Five teams were used to create offense agents: *helios, gliders, cyrus, axiom and aut*; agents of the defense team were based on *agent2d* team. Similar to initial phase in FA domain, we deployed the existing agent teams in the HFO domain and collected state observations. Since the actions of other agents are not directly observable, the actions in these training samples were computed from the observed state transitions. To evaluate the ability to learn from limited data, we only used data from 300 episodes for each type of agent to create the corresponding basic tree models that were then used by the ad hoc agent to predict the behavior of other agents in its reasoning.

We evaluated our architecture's performance in HFO domain with each builtin external team, i.e., all offense agents other than the ad hoc agent were based on one external team at a time. In **Exp3**, we measured performance in the limited version of the game, i.e., two offense players (including ad hoc agent) against two defense agents (including goalkeeper). In **Exp4**, we measured performance in the full version of the game, i.e., four offense players (including the ad hoc agent) played against five defense agents (including the goal keeper). In **Exp5** and **Exp6**, we conducted experiments to evaluate performance under partial observability in the limited and full versions of the HFO domain (respectively). We evaluate our ad hoc agent using 1000 episodes each of the limited version and full version with the external agent types selected randomly; the same setup was used for experiments with full observability and partial observability. As the baselines for our experiments, we used recent (state of the art) AHT methods: PPAS [22], and PLASTIC [6]. These methods used the same external agent types for comparison, allowing us to compare our results with those in their papers.

Together, these experiments were used to evaluate the three hypotheses (**H1, H2**, and **H3**). All claims are statistically significant unless stated otherwise.

Table 4: Wins (%) for guards with hand-crafted policies (**Exp1**). Model adaptation improves performance.

Experiment	Win %
Without model selection (**Base1**)	63
When using direct comparison (**Base2**)	68
With model selection algorithm (**KAT**)	73

Table 5: Wins (%) for guards with built-in policies (**Exp2**). Model adaptation improves performance.

Experiment	Win %
Without model selection (**Base3**)	47
When using direct comparison (**Base4**)	45
With model selection algorithm (**KAT**)	55

Table 6: Wins (%) for guards with hand-crafted policies (**Exp1**). Communication addresses partial observability.

Policy	With Comm. (%)	Without Comm. (%, **Base5**)
Policy1	73	58
Policy2	19	8

Table 7: Wins (%) for team of guards built-in policies (**Exp2**). Communication addresses partial observability.

Policy	With Comm. (%)	Without Comm. (%, **Base6**)
Policy220	79	85
Policy650	42	41
Policy1240	46	43
Policy1600	18	17

4.2 Experiment Results

We begin with the results of experiments in the **FA domain**. First, Table 4 summarizes the results of using our model selection algorithm in **Exp1**. When the other agents followed the hand-crafted policies and the model selection mechanism was not used by the ad hoc agent (**Base1**), the team of guards had the lowest winning percentage. When the ad hoc agent used the model selection algorithm without a graded strategy for comparing predicted and actual actions (**Base2**), the performance of the team of guards improves. When the ad hoc agent used our model selection method (Algorithm 1), the winning percentage of the team of guards is substantially higher. These results demonstrated that the adaptive selection of behavior models in our architecture (KAT) improved the performance of the team with the ad hoc agent.

Next, the results of **Exp2** are summarized in Table 5. We observed that the ad hoc agent was able to adapt to unforeseen agents (teammates and opponents) that were using the builtin policies (of FA domain) based on online revision of the behavior models (i.e., the FF trees) learned from the hand-crafted policies and the model selection algorithm. Our approach provided the best performance compared with not using any model adaptation or selection (**Base3**), and with using model selection without the graded strategy (**Base4**). These results and the results from the Table 4 support hypotheses **H1** and **H2**.

The results from **Exp1** under partial observability, with and without communication (**Base5**) strategies, are summarized in Table 6. Recall that other agents

Table 8: Prediction accuracy of the learned agent behaviour models in limited (2v2) version of the HFO domain (**Exp3**).

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 behaviour models in full (4v5) version of the HFO domain (**Exp4**).

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

used the hand-crafted policies of the FA domain in this experiment. When the communication actions were enabled for the ad hoc (guard) agent, the winning percentage of the team of guards was substantially higher than the winning percentage of the guards that did not have the opportunity to use the communication actions. **Policy2** is a particularly challenging scenario (both guards and attackers shoot), which justifies the lower winning percentage.

Next, the results from **Exp2** under partial observability, with and without communication (**Base6**) strategies, are summarized in Table 7. Recall that the other agents used the builtin policies of the FA domain. We observed that the winning percentage of the team of guards when following the policies 650, 1240, and 1600 was comparable or higher when the communication actions are enabled compared with the absence of these actions (**Base6**). With Policy 220, we observed that the performance was slightly worse with the communication actions. However, recall that unlike the other policies, Policy 220 results in the guards spreading themselves in-front of the fort and shooting continuously. As a result, partial observability and communication strategies did not contribute significantly to the outcome. These results support hypothesis **H3**.

We next describe the results from the **HFO domain**. Recall that the behavior models were learned for the agents other than the ad hoc agent using data from 300 episodes (for each external agent type). This is much smaller than the number of training samples (often a few million) used by state of the art data-driven methods that do not reason with domain knowledge. The prediction accuracy of the learned behavior models created for the limited version (**Exp3**) and full version (**Exp4**) of the HFO domain are summarized in Tables 8 and 9 respectively. We observed that the prediction accuracy varies over a range for the different external agent types. Although the accuracy values are not very high, the models can be learned and revised rapidly (during run-time).

The results of **Exp3** and **Exp4**, i.e., evaluating our architecture (KAT) compared with the state of the art baselines for the HFO domain (PPAS, PLASTIC), are summarized in Table 10. Recall that these data-driven baselines are based on millions of training examples and without any knowledge-based reasoning. The fraction of goals scored (and games won) by the team of offense agents includ-

Table 10: Fraction of goals scored by the offense team in HFO domain in the limited version(2v2) and full version(4v5).

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

Table 11: Fraction of goals scored by the offense team in HFO domain in the limited version (2v2) and full version (4v5) under partial observability.

Version	KAT (%)	Original Team (%)
Limited (2v2)	70.5	76.6
Full (4v5)	18	20.5

ing our ad hoc agent is comparable to the goals scored by the baselines for the limited version, and substantially better than the baselines for the full version. These results strongly support hypotheses **H1** and **H2**.

Finally, the results of evaluating KAT under partial observability (in HFO domain) are summarized in Table 11 compared with teams of external agent types without any ad hoc agent. Although the numbers indicate that performance with our architecture is slightly lower than without any ad hoc agents, the difference is not significant and due to noise (e.g., in perceived angle to the goal). Also note that this is a challenging multiagent collaboration domain, and the ability to provide performance comparable with teams whose training datasets are orders of magnitude larger strongly supports **H3**. Due to space constraints, additional video results, including that of different number of team members, are in our open-source repository [12].

5 Conclusions

Ad hoc teamwork (AHT) is a fundamental problem in the coordination of multi-agent teams. This paper described an AHT architecture (KAT) that embedded the principles of refinement and ecological rationality to leverage the complementary strengths of knowledge-based reasoning and data-driven learning methods. The architecture supports non-monotonic logical reasoning with commonsense domain knowledge and simple predictive models of other agents' behavior that are learned from limited training examples using heuristic methods. We experimentally evaluated our architecture in benchmark simulated domains and demonstrated that our architecture's performance is comparable or better than state of the art data-driven baselines. We also demonstrated the ability to adapt learned behavior models and to operate under partial observability and limited communication conditions.

Our architecture provides multiple directions for further research. In particular, we will further explore the interplay between representation, reasoning, and learning for AHT in other (more) complex domains and on physical robots. We will also investigate use of different communication strategies and the introduction of multiple ad hoc agents (equipped with our architecture) in a team. Furthermore, we will explore the use of the learned behavior models to augment or revise the relevant knowledge used for reasoning.

References

1. Balai, E., Gelfond, M., Zhang, Y.: Towards Answer Set Programming with Sorts. In: International Conference on Logic Programming and Nonmonotonic Reasoning (2013)
2. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning (2003)
3. Baral, C., Gelfond, G., Pontelli, E., Son, T.C.: An action language for multi-agent domains. Artificial Intelligence **302**, 103601 (2022)
4. Baral, C., Gelfond, G., Son, T.C., Pontelli, E.: Using answer set programming to model multi-agent scenarios involving agents' knowledge about other's knowledge. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS. vol. 1, p. 259–266 (05 2010)
5. Baral, C., Son, T.C., Pontelli, E.: Reasoning about multi-agent domains using action language $\mathcal{C}$: A preliminary study. In: Dix, J., Fisher, M., Novák, P. (eds.) Computational Logic in Multi-Agent Systems. pp. 46–63. Springer Berlin Heidelberg (2010)
6. Barrett, S., Rosenfeld, A., Kraus, S., Stone, P.: Making friends on the fly: Cooperating with new teammates. Artificial Intelligence **242**, 132–171 (2017). https://doi.org/https://doi.org/10.1016/j.artint.2016.10.005
7. 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). https://doi.org/10.1609/aaai.v27i1.8659
8. Bowling, M., McCracken, P.: Coordination and adaptation in impromptu teams. In: National Conference on Artificial Intelligence. p. 53–58 (2005)
9. 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). https://doi.org/10.1609/aaai.v34i05.6196
10. 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)
11. Dodampegama, 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)
12. Dodampegama, H., Sridharan, M.: Code and results for the ICLP 2023 paper (2023), https://github.com/hharithaki/KAT
13. Gelfond, M., Inclezan, 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.: Towards a Rational Theory of Heuristics, pp. 34–59. Palgrave Macmillan UK, London (2016)

15. Gigerenzer, G.: What is Bounded Rationality? In: Routledge Handbook of Bounded Rationality. Routledge (2020)
16. Gigerenzer, G., Gaissmaier, W.: Heuristic Decision Making. Annual Review of Psychology **62**, 451–482 (2011)
17. 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)
18. Katsikopoulos, K., Simsek, O., Buckmann, M., Gigerenzer, G.: Classification in the Wild: The Science and Art of Transparent Decision Making. MIT Press (February 2021)
19. Macke, W., Mirsky, R., Stone, P.: Expected value of communication for planning in ad hoc teamwork. In: AAAI Conference on Artificial Intelligence. pp. 11290–11298 (May 2021)
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. 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)
22. 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. pp. 648–660. Springer International (2021)
23. Son, T., Balduccini, M.: Answer set planning in single- and multi-agent environments. Künstliche Intelligenz **32** (06 2018). https://doi.org/10.1007/s13218-018-0546-8
24. Son, T.C., Pontelli, E., Nguyen, N.H.: Planning for multiagent using asp-prolog. In: Dix, J., Fisher, M., Novák, P. (eds.) Computational Logic in Multi-Agent Systems. pp. 1–21. Springer Berlin Heidelberg (2010)
25. Son, T.C., Sakama, C.: Reasoning and Planning with Cooperative Actions or Multiagents using Answer Set Programming. In: Declarative Agent Languages and Technologies VII, Lecture Notes in Computer Science, vol. 5948, pp. 208–227. Springer Berlin Heidelberg (2010)
26. Sridharan, M., Gelfond, M., Zhang, S., Wyatt, J.: REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Journal of Artificial Intelligence Research **65**, 87–180 (May 2019)
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. Wu, F., Zilberstein, S., Chen, X.: Online planning for ad hoc autonomous agent teams. In: International Joint Conference on Artificial Intelligence. p. 439–445 (2011)
29. 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)
30. 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), https://www.microsoft.com/en-us/research/publication/deep-interactive-bayesian-reinforcement-learning-via-meta-learning/