On the benefit of ontological reasoning in collaborative problem-solving*

Chukwuemeka David Emele[1], Timothy J. Norman[1], Murat Şensoy[1], and Simon Parsons[2]

[1] University of Aberdeen, Aberdeen, AB24 3UE, UK
[2] Brooklyn College, City University of New York, 11210 NY, USA
{c.emele,t.j.norman,m.sensoy}@abdn.ac.uk
parsons@sci.brooklyn.cuny.edu

Abstract. In multi-agent systems, agents often depend on others to perform different aspects of shared goals. In such settings, deciding to whom to delegate a task could be complicated because activities of collaborators may be regulated by policies. Especially when such policies are private, learning these policies may become crucial to estimate the outcome of proposed task delegation decisions. In this paper, we present an approach that utilises ontological reasoning to aid the learning of policies. Our approach combines ontological reasoning, machine learning and argumentation in a novel way to accomplish this. Using our approach, autonomous agents can reason about the policies that others are operating with, and make informed decisions about to whom to delegate a task (or approach for the provision of certain resources). In a series of experiments, we demonstrate the utility of this novel combination of techniques. Our empirical evaluation shows that models of others' policies can be refined through ontological reasoning, and such models can be used to guide future task delegations.

Keywords: Dialogue, Negotiation, Collaboration, Machine learning, Policies, Norms, Ontological reasoning, Domain knowledge, Delegation

1 Introduction

In many collaborative problem-solving settings, it is very important to ascertain whether or not other collaborators possess constraints that may hinder them from acting as requested or desired. In fact, collaborators cannot presuppose that the policies of their peers will always permit them to carry out tasks

* This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

delegated to them. Agents must, therefore, accept the fact that a number of constraints may prohibit a collaborator from committing to a task (or agreeing to provide a resource) under certain circumstances. An important question is, "how can agents identify what (social) constraints others are working within?" We envisage that one way to do this is to learn the policy of others by observing their behaviours. This approach is suitable for environments where there is a one-to-one mapping between policy and behaviour. Another way to do this is to use evidence derived from dialogue. Here, behaviour is correlated with both policy and other constraints, such as resource availability. Evidence acquired during dialogue is used to aid the construction of models of others' policies.

In Emele *et al.* [2], we show that intelligent agents can determine what policies others are operating within by mining the data gathered from past encounters with that agent (or similar agents - using some similarity metrics [5]) as they collaborate to solve problems. This prior research uses a novel combination of evidence derived from argumentation-based dialogue (which we refer to as ADE) and machine learning to build stable models of others' policies. In this paper, we explore the question that given agents may have access to some background (or ontological) domain knowledge, how can we exploit such knowledge to improve models of others' policies? To do this, we propose the use of ontological reasoning, argumentation and machine learning to aid in making effective predictions about who to approach if some other collaborator is to be delegated to provide a resource or perform a task on the behalf of another.

As a way of evaluating the contribution of our approach we hypothesize as follows: exploiting ontological reasoning in learning agents' resourcing policies mean that more accurate and stable models of others' policies can be derived more rapidly than without exploiting such reasoning.

The rest of this paper is organised as follows. Section 2 discusses plan resourcing in the face of policies. Section 3 presents our approach for learning agents' policies. Section 4 reports the results of our evaluations. Section 5 summarises our findings, discusses related work and future directions. Section 6 concludes.

2 Plan resourcing in the face of policies

Plan resourcing, in general, is concerned with identifying a suitable candidate (or candidates) to approach for the provision of resources required for the enactment of a plan (or plans). In collaborative settings where each agent is regulated by a set of rules, referred to as *policies* (or norms), policies could determine what actions an agent is (or is not) allowed to perform, including what resources an agent is (or is not) permitted to release to others when requested. Plan resourcing, in this case, is not just a matter of finding agents that possess the appropriate resource (or expertise); if an agent has the required resource but is operating under a policy that prohibits the provision of such resource, it may decline the request. In such settings, the aim is to identify agents who possess the required resource, and whose policies permit the provision of the required resource (or performance of the required action). It may also be useful

to distinguish between agents that are permitted to accede to a request and those that are obliged to do so [7, 6], but we do not make this distinction here.

We assume that there is a set $\mathcal{T} = \{t_1, \cdots, t_n\}$ of tasks which agents are assigned to fulfill. The fulfillment of each task requires a number of resources to be used. We focus on the resource acquisition necessary for the successful completion of assigned tasks. Resources generally refers to physical equipment, capabilities, expertise or information that are required to carry out a task. For example, *helicopters*, *jeeps*, etc. We denote the set of resources, $\mathcal{R}$, as a finite set such that $r_1, r_2, \ldots r_n \in \mathcal{R}$.

Definition 1 (Resource Requirements). *The resource requirements of task t is a function which maps task t to a set of resources required to carry out t, and is defined as follows:*

$$requirements : \mathcal{T} \rightarrow 2^{\mathcal{R}}$$

In our framework, an agent that has been assigned a task is solely responsible for its fulfillment. However, an agent can delegate an aspect of a task to another [10]. For example, if agent x is responsible for performing task t_x, which requires the use of some resource r_x (i.e. $r_x \in requirements(t_x)$) then x may choose to delegate the provision of r_x to another agent y_1. Provided agent y_1 has resource r_x, and does not have any policy that forbids the provision of r_x then we assume y_1 will make r_x available to x.

2.1 Policies

Agents have policies that govern how resources are deployed to others. In our model, policies are conditional entities (or rules) and so are relevant to an agent under specific circumstances only. These circumstances are characterised by a set of features, e.g., vehicle type, weather conditions, etc.

We define a feature as a characteristic of the prevailing circumstance under which an agent is operating (or carrying out an activity). However, we leave the determination of what characteristics should be captured and how best to represent them to the system designer (as it is largely system-specific). Let $\mathcal{F}$ be the set of all features such that $f_1, f_2, \ldots \in \mathcal{F}$. Our concept of policy maps a set of features into an appropriate policy decision. In our framework, an agent can make one of two policy decisions at a time, namely (1) *grant*, which means that the agent is allowed to provide the resource when requested; and (2) *decline*, which means that the policy prohibits the agent from providing the resource.

Definition 2 (Policies). *A policy is defined as a function with the signature $\Pi : 2^{\mathcal{F}} \rightarrow \{grant, decline\}$, which maps feature vectors of agents to appropriate policy decisions.*

In order to illustrate the way policies may be captured in this model, we present an example. Let us assume that f_1 is resource, f_2 is purpose, f_3 is weather report (with respect to a location), f_4 is the affiliation of the agent, and f_5 is the day the resource is required, then $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ in Table 1 will be interpreted as follows:

Table 1. An agent's policy profile.

Policy Id	f_1	f_2	f_3	f_4	f_5	Decision
$\mathbb{P}_1$	h	trm				grant
$\mathbb{P}_2$	av		vc			decline
$\mathbb{P}_3$	j					grant
$\mathbb{P}_4$	c		vc	xx		grant
...	...	...	...	...	...	...
$\mathbb{P}_n$	q	yy	w	xx	z	decline

$\mathbb{P}_1$**:** You are **permitted** to release a *helicopter* (h), to an agent if the *helicopter* is required for the purpose of transporting relief materials (trm);
$\mathbb{P}_2$**:** You are **prohibited** from releasing an *aerial vehicle* (av) to an agent in bad weather conditions - e.g. volcanic clouds (vc);
$\mathbb{P}_3$**:** You are **permitted** to release a *jeep* (j) to an agent.

If an agent intends to deploy a *helicopter* in an area with volcanic clouds then the provider is forbidden from providing the resource but might offer a ground vehicle (e.g. *jeep*) to the consumer if there is no policy prohibiting this and the resource is available. Furthermore, whenever a consumer's request is refused, the consumer may challenge the decision, and seek justifications for the refusal. This additional evidence is beneficial, and could be used to improve the model, hence, the quality of decisions made in future episodes.

2.2 Argumentation-based dialogue

The argumentation-based framework we propose here enables agents to negotiate and argue about plan resourcing, and use evidence derived from argumentation to build more accurate and stable models of others' policies. The arguments exchanged include the features that an agent requires in order to make a decision about whether or not to *grant* or *decline* a request to provide resources when requested. The agent attempts to predict the policies of other agents by reasoning over policy models (built from past experience). Such predictions are further refined by exploiting ontological reasoning (discussed later in Section 3.2).

To illustrate the sorts of interaction between agents, consider the example dialogue in Table 2. Let x and y be consumer and provider agents respectively. Suppose we have an argumentation framework that allows agents to ask for and receive explanations (as in Table 2, *lines* 11 *to* 14), offer alternatives (*line* 10 in Table 2), or ask and receive more information about the attributes of requests (*lines* 4 *to* 9 in Table 2), then x can gather additional information regarding the policy rules guiding y concerning the provision of resources.

Negotiation for resources takes place in a turn-taking fashion. Figure 1 illustrates the protocol employed in this framework, which guides dialogical moves. Our approach in this regard extends the dialogue for resource negotiation proposed by McBurney & Parsons [8]. The dialogue starts, and then agent x sends

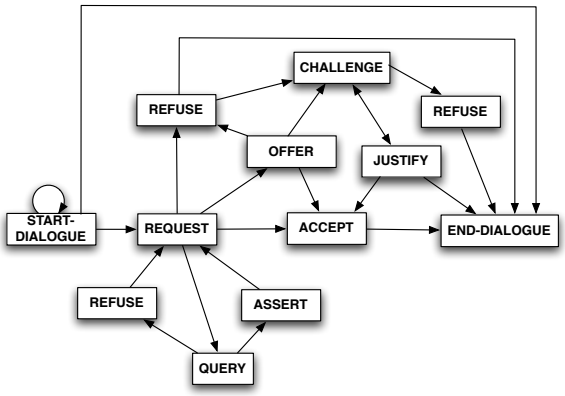

Fig. 1. The negotiation protocol.

a request to agent y (e.g., *line* 3, Table 2). The provider, y, may respond by conceding to the request (accept), refusing, offering an alternative resource, or asking for more information (query) such as in *line* 4 in Table 2. If the provider agrees to provide the resource then the negotiation ends. If, however, the provider declines the request then the consumer may challenge that decision, and so on. If the provider suggests an alternative (*line* 10 in Table 2) then the consumer evaluates it to see whether it is acceptable or not. Furthermore, if the provider needs more information from the consumer in order to decide, the provider would ask questions that will reveal the features it requires to make a decision (query, assert/refuse in Figure 1). The negotiation ends when agreement is reached or all possibilities explored have been rejected.

3 Learning agents' policies

Since policies are private, agents need to find effective ways of delegating the provision of resources if they are to successfully resource assigned tasks. To do so, our approach allows agents to find out agents whose policy constraints will most likely, according to a chosen metric, permit to execute the delegated task — in this case, resource provision. Whenever there is a task to be resourced, policy predictions are generated alongside their confidence values from the policy models that have been learned over time. Confidence values of favourable policy predictions are compared to determine which candidate to approach for the provision of a resource. These predictions are based on evidence derived from argumentation, and can be further refined by reasoning over domain knowledge.

Example 1 *Consider a situation where a Red Cross agent, x, is collaborating with a number of agents, y_1, y_2, y_3, and y_4, in a collaborative peace-keeping operation. Let us assume that agent x does not have a helicopter in its resource pool, and that each of agents y_1, y_2, y_3, and y_4 can provide helicopters, jeeps, vans, bikes, fire extinguishers, and unmanned aerial vehicles (UAVs).*

Table 2. Dialogue example.

#	**Dialogue Sequence**	**Locution Type**
1	x: Start dialogue.	START-DIALOGUE
2	y: Start dialogue.	START-DIALOGUE
3	x: Can I have a *helicopter* for \$0.1M reward?	REQUEST
4	y: What do you need it for?	QUERY
5	x: To transport relief materials.	ASSERT
6	y: To where?	QUERY
7	x: A refugee camp near region XYZ.	ASSERT
8	y: Which date?	QUERY
9	x: On Friday 16/4/2010.	ASSERT
10	y: I can provide you with a *jeep* for \$5,000.	OFFER
11	x: But I prefer a *helicopter*, why offer me a *jeep*?	CHALLENGE
12	y: I cannot release a *helicopter* in volcanic eruption.	JUSTIFY
13	x: There is no volcanic eruption near region XYZ.	CHALLENGE
14	y: I agree, but the ash cloud is spreading, and weather report advises that it is not safe to fly on that day.	JUSTIFY
15	x: Ok then, I accept your offer of a *jeep*.	ACCEPT
16	y: That's alright. Good-bye.	END-DIALOGUE

Agent x has to decide which agent (i.e. y_1, y_2, y_3, or y_4) to approach to provide the *helicopter*. Let us assume that the four agents have sim similar capabilities. Agent x, at this point, attempts to predict the policy of these agents with respect to resource provision. This prediction is based on policy models built from past encounters with these agents (or similar agents). Assuming the predictions are as follows: (i) y_1 *will accept to provide the helicopter with 0.6 confidence*; (ii) y_2 *will accept with 0.9 confidence*; (iii) y_3 *will accept with 0.7 confidence*; and (iv) y_4 *will decline with 0.8 confidence*. If the decision to choose a provider is based on policy predictions alone, then y_2 is the best candidate.

3.1 Learning from past experience

When an agent has a collection of experiences with other agents described by feature vectors (see Section 2.1), we can make use of existing machine learning techniques to learn associations between sets of features (e.g., $f_1, f_2, \ldots f_n \in \mathcal{F}$) and policy decisions (i.e., *grant* and *decline*). The training examples used in our approach are derived from plan resourcing episodes (or interactions), which involves resourcing a task t using agent y. The interaction can be represented as a feature vector ($\vec{F}_y, grant$ or $\vec{F}_y, decline$) where $\vec{F}_y \in 2^{\mathcal{F}}$ is the set of features for that characterise the task context. In this way, an agent may build a model of the relationship between observable features of agents and their policies. Subsequently, when faced with resourcing a new task, the policy model can be used to obtain a prediction of whether or not a particular provider has a policy that permits the provision of the resource.

Specifically, we investigate three classes of machine learning algorithms [9, 16], namely instance-based learning (using k-nearest neighbours), rule-based

learning (using sequential covering), decision tree learning (using C4.5). Owing to space limitation, we do not discuss instance-based and rule-based learning in this paper. In the following, we briefly describe a decision tree induction using C4.5.

C4.5 decision tree algorithm The well-known C4.5 decision tree algorithm [11] uses a method known as divide and conquer to construct a suitable tree from a training set S of cases. If all the cases in S belong to the same class C_i, the decision tree is a leaf labeled with C_i. Otherwise, let B be some test with outcomes $\{b_1, b_2, \ldots, b_n\}$ that produces a partition of S, and denote by S_i the set of cases in S that has outcome b_i of B. The decision tree rooted at B is shown in Figure 2, where T_i is the result of growing a sub-tree for the cases in S_i. The root node B is based on a feature that best classifies S. This feature is determined using information theory. That is, the feature having the highest *information gain* is selected.

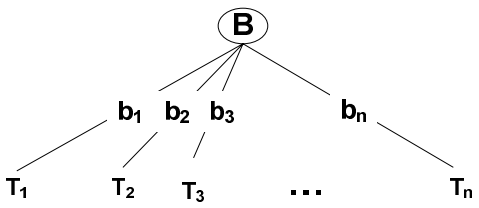

Fig. 2. Tree rooted at the test B and its branches based on its outcomes.

Information gain of a feature is computed based on *information content*. Assume that testing a feature A in the root of the tree will partition S into disjoint subsets $\{S_1, S_2, \ldots, S_t\}$. Let $RF(C_i, S)$ denote the relative frequency of cases in S that belong to class C_i. The information content of S is then computed using Equation 1. The *information gain* for A is computed using Equation 2.

$$I(S) = -\sum_{i=1}^{n} RF(C_i, S) \times log(RF(C_i, S)) \tag{1}$$

$$G(S, A) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \times I(S_i) \tag{2}$$

Once the feature representing the root node is selected based on its information gain, each value of the feature leads to a branch of the node. These branches divide the training set used to create the node into disjoint sets $\{S_1, S_2, \ldots, S_t\}$. Then, we recursively create new nodes of the tree using these subsets. If S_i contains training examples only from the class C_i, we create a leaf node labeled with the class C_i; otherwise, we recursively build a child node by selecting another feature based on S_i. This recursive process stops either when the tree perfectly classifies all training examples, or until no unused feature remains (we refer interested readers to [11] for a listing of the C4.5 algorithm).

Table 3. Training examples.

#	Vehicle ID	Type	Age	Price	Class
1	Van789	Van	10	10,000	*grant*
2	Van999	Van	5	20,000	*grant*
3	Car767	Car	8	5,000	*grant*
4	Car777	Car	15	1,000	*grant*
5	Coach09	Coach	2	200,000	*grant*
6	Yacht121	Yacht	20	300,000	*decline*
7	Yacht123	Yacht	2	500,000	*decline*
8	Speedboat1	Speedboat	4	8,000	*decline*
9	Speedboat2	Speedboat	15	2,000	*decline*
10	Cruiser31	Cruiser	10	100,000	*decline*

Consider the training examples listed in Table 3, where $Vehice\ ID$, $Type$, Age, and $Price$ are the only features. C4.5 decision tree algorithm makes induction only over numerical feature values. However, it is unable to make induction or generalisation over nominal feature values (i.e., terms). For instance, a decision node based on the *price* test in Figure 3 can be used to classify a new case with price $\$250,000$, even though there is no case in the training examples with this price value. However, a new case with an unseen type, for instance a *submarine*, cannot be classified using the decision node based on the feature $Type$. In the next section, we extend C4.5 decision tree using ontological reasoning (which we refer to as *semantically-enriched decision tree* — STree), and explore how much domain knowledge can improve learning.

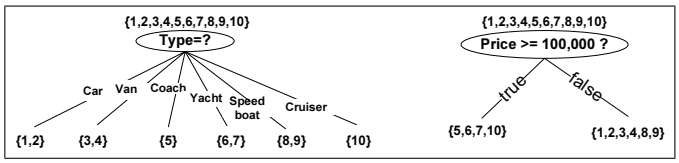

Fig. 3. Decision nodes created using the tests on $Type$ (on the left hand side) and $Price$ (on the right hand side).

3.2 Learning from domain knowledge

We argue that ontological reasoning can be used to improve the performance of machine learning approaches. Semantic Web technologies allow software agents to use ontologies to capture domain knowledge, and employ ontological reasoning to reason about it [4]. Figure 4 shows a part of a simple ontology about vehicles and weather conditions. The hierarchical relationships between terms in an ontology can be used to make generalisation over the values of features while learning policies as demonstrated in Example 2. Specifically, in this section, we describe how we exploit ontological reasoning to improve C4.5 decision trees.

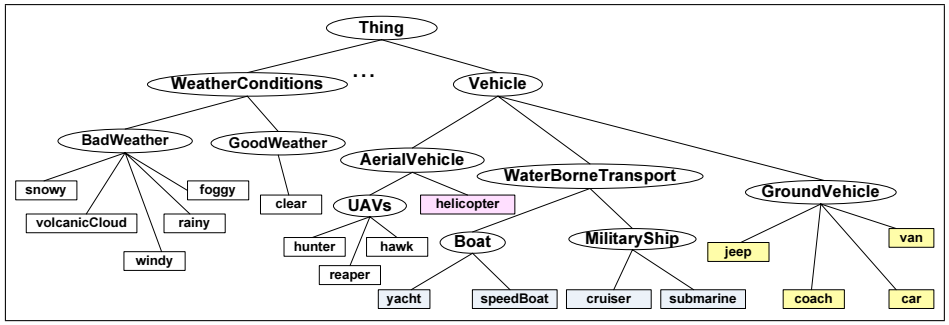

Fig. 4. A simple ontology for vehicles and weather conditions. Ellipsis and rectangles represent concepts and their instances respectively.

Example 2 *Suppose agent x in Example 1 has learned from previous interactions with agent y_1 that there is a policy that forbids y_1 from providing a helicopter when the weather is rainy, foggy, snowy, or windy. In addition, suppose agent x has learned from previous experience that agent y_1 is permitted to provide a jeep in these conditions. This information has little value for x if it needs a helicopter when the weather is not rainy, foggy, snowy, or windy but volcanic clouds are reported. On the other hand, with the help of the ontology in Figure 4, agent x can generalise over the already experienced weather conditions and expect that "agent y_1 is prohibited from providing helicopters in bad weather conditions". Such a generalisation allows x to reason about y_1's behavior for the cases that are not experienced yet. That is, with the help of the domain knowledge, agent x can deduce (even without having training examples involving volcanic clouds directly) that agent y_1 may be prohibited from providing a helicopter if there is an evidence of volcanic clouds in the region.*

Semantically-enriched decision tree algorithm Semantically-enriched decision trees (STrees) are built upon the subsumptions relationships between terms in the ontology. These relationships can be derived automatically using an off-the-shelf ontology reasoner [4]. The main idea of STree is to replace the values of nominal features with more general terms iteratively during tree induction, unless this replacement results in any decrease in the classification performance.

Algorithm 1 summarises how the values of a feature, A, are generalised for S. First, we compute the original gain $G(S, A)$ (line 3). Second, we create a set called *banned*, which contains the terms that cannot be generalised further (line 4). Initially, this set contains only the root concept $Thing$. Third, we create the set T that contains A's values in S (line 5). While there is a generalisable term $t \in T$ (lines 6-18), we compute its generalisation t' using ontological reasoning (line 8) and create the set T' by replacing more specific terms in T with t' (line 9). If this term is an instance of a concept, then the generalisation of the term is the concept, e.g., $Yacht$ is generalisation of $Yacht121$. If the term is a concept, its generalisation is its parent concept, e.g., $WaterBorneTransport$ is generalisation of $Yacht$. For instance, let S be the data in Table 3, then T would

Algorithm 1 Generalising values of nominal feature A in training set S.

1: **Input :** S, A
2: **Output:** T
3: $g = G(S, A)$
4: $banned = \{Thing\}$
5: $T = getFeatureValues(S, A)$
6: **while** *true* **do**
7: **if** $\exists t$ **such that** $t \in T \wedge t \notin banned$ **then**
8: $t' = generalise(t)$
9: $T' = replaceWithMoreSpecificTerms(T, t')$
10: $s = replaceFeatureValues(S, A, T')$
11: **if** $G(s, A) = g$ **then**
12: $S = s$ **and** $T = T'$
13: **else**
14: $banned = banned \cup \{t\}$
15: **end if**
16: **else**
17: **break**
18: **end if**
19: **end while**

contain $Yacht$, $Speedboat$, $Cruiser$, Van, Car, $Coach$, and $Cruiser$. If Car is selected as t, t' would be $GroundVehicle$. In this case, T' would contain $Yacht$, $Speedboat$, $Cruiser$, and $GroundVehicle$. Next, we check if the generalisation leads to any decrease in the information gain. This is done by creating a temporary training set s from S by replacing A's values in S with the more general terms in T' (line 10) and then comparing $G(s, A)$ with the original gain g (line 11). If there is no decrease in the information gain, S and T are replaced with s and T' respectively; otherwise t is added to *banned*. We iterate through until we cannot find any term in T to generalise without a decrease in information gain.

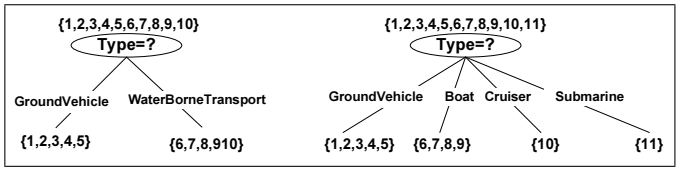

Fig. 5. Decision nodes using the generalisation of cases in Table 3 (left hand) and after the addition of a new case (11, *Submarine*, 40, 800,000, *grant*) (right hand).

For the examples in Table 3, $\{WaterBorneTransport, GroundVehicle\}$ would be the output of the algorithm because any further generalisation results in a decrease in information gain. Hence, a decision node based on $Type$ feature would be as shown in Figure 5 (left hand side). A new test case (11, $Submarine$, $40years$, \$800,000) would be classified as *decline* using this decision node, because a submarine is a $WaterBorneTransport$ and all known

WaterBorneTransports are labeled as *decline*. If the actual classification of the case is *grant* instead of *decline*, the decision node would be updated as seen in Figure 5 (right hand side), because generalisation of *Submarine* or *Cruiser* now results in a decrease in the information gain.

4 Evaluation

In evaluating our approach, we employed a simulated agent society where a set of consumer agents interact with a set of provider agents with regard to resourcing their plans. Each provider is assigned a set of resources. Providers also operate under a set of policy constraints that determine under what circumstances they are permitted to provide resources to consumers. In the evaluation reported in this section, we demonstrate that it is possible to use domain knowledge to improve models of others' policies, hence increase their predictive accuracy, and performance. We hypothesize as follows:

- *Hypothesis 1:* In relatively small domains or in situations where sufficiently large training data is available, agents that incorporate domain knowledge will perform no worse than those without domain knowledge.
- *Hypothesis 2:* In complex domains or in situations where limited training data is available, exploiting appropriate domain knowledge in learning policies mean that more accurate and stable models of others' policies can be derived more rapidly than without exploiting such knowledge.

4.1 Experimental Setup

We evaluate the performance of background domain knowledge in refining models of others' policies in situations where agents' policies are relatively small or large training data is available (which we refer to as the *closed* scenario) and in situations where agents' policies are relatively large or limited training data is available (which we refer to as the *open* scenario). In each scenario, five agent configurations (SM, C4.5, kNN, SC, and STree) are investigated. In SM, simple memorisation of outcomes is used. In C4.5, C4.5 decision tree classifier [16] is used. In kNN, k-nearest neighbour algorithm [1] is used. In SC, sequential covering rule learning algorithm [3] is used. Lastly, in STree, agents use our novel semantically-enriched decision trees to build models of others' policies.

There are five features that are used to capture agents' policies, namely *resource type* (*Resource*), *affiliation*, *purpose* (*Purpose*), *location* (*Location*), and *day* (*Day*). The experimental parameters are summarised in Table 4. These features provide the possible task context for each agent in the system. In the closed scenario, there are 375 (that is, $3 \times 5 \times 5 \times 5$) possible task configurations. Thus, given 4 providers, the consumer is faced with a problem domain in which there are (potentially) 1,500 individual policies for different task configurations (that is, 375×4). In the open scenario, however, there are $3 \times 20 \times 20 \times 20 = 24,000$ possible configurations, and the problem domain can (potentially) have 96,000

Table 4. Experimental Parameters

Parameter	Closed	Open	Description
Consumer	1	1	No. of consumer agents
Provider	4	4	No. of provider agents
ϕ	100	100	Learning interval
Task	800	800	No. of tasks per experiment
Resources	5	20	No. of resource types
Locations	5	20	No. of locations
Purposes	5	20	No. of purposes
Days	3	3	No. of simulated days

individual policies for different task configurations (considering the 4 providers). Consumer agents were initialised with random models of the policies of providers.

100 experimental runs were conducted. Each run consists of 8 rounds and at each round, 100 tasks were randomly generated (from the possible configurations) and assigned to a consumer, and the prediction rate of that consumer is recorded. The model built in each round is used to bootstrap the next round until the 8 rounds are completed. In the control condition (SM configuration), the consumer simply memorises outcomes from past interactions. Since there is no generalisation in SM, the *confidence* (or prediction accuracy) is 1.0 if there is an exact match in memory, else the probability is 0.5.

4.2 Results

Figure 6 illustrates the performance of the five configurations we considered in predicting agents' policies in the closed scenario. The results show that STree, SC, kNN and C4.5 consistently outperform SM. Furthermore, STree, SC and kNN consistently outperform C4.5. It is interesting to see that, with relatively small training set, SM performed better than C4.5. This is, we believe, because the model built by C4.5 overfit the data.

The decision trees (i.e. STree and C4.5) were pruned after each set of 100 tasks and after 300 tasks the accuracy of the C4.5 model rose to about 83% to tie with SM and from then C4.5 performed better than SM. Similarly, STree performed much better than SC with relatively small training set. We believe, this is because STree takes advantage of domain knowledge and so can make informed inference (or guess) with respect to feature values that do not exist in the training set. After 400 tasks the accuracy of STree, SC, and kNN remained unchanged at about 97% from 400 to 800 tasks. We believe, at this point, almost all the test instances have been encountered and so have been learned (and now exist in the training set for future episodes). This confirms hypothesis 1.

Figure 7 illustrates the effectiveness of exploiting domain knowledge in learning policies in the open scenario. The results show that the technique that exploits domain knowledge (STree) significantly outperforms the other techniques that did not. After 300 tasks the accuracy of the STree model had exceeded 82%

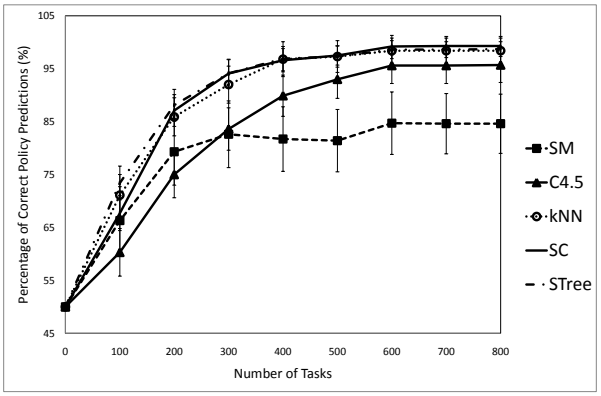

Fig. 6. The effectiveness of exploiting ontological reasoning in learning policies (closed scenario).

while that of SM, C4.5, kNN, and SC were just approaching 53%, 64%, 65% and 67% respectively. However, tests of statistical significance were applied to all results presented in this evaluation and they have been found to be statistically significant by t-test with $p < 0.05$. This confirms hypothesis 2.

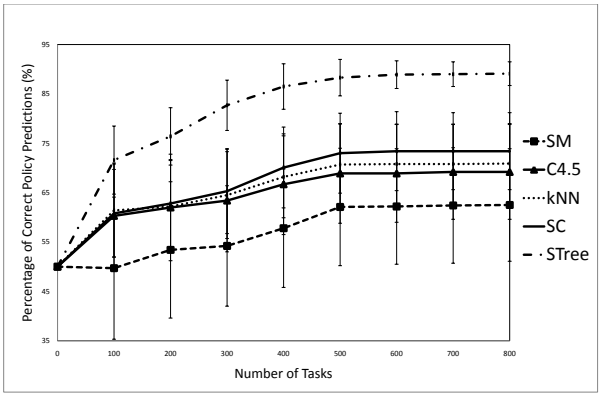

Fig. 7. The effectiveness of exploiting domain knowledge in learning policies (open scenario).

Overall, these results confirm that exploiting ontological reasoning over appropriate domain knowledge will mean that more accurate and stable models of others' policies can be derived more rapidly than without such reasoning.

5 Discussion

Results of our evaluation show that ontological reasoning can be exploited to refine and further improve models of others' policies. Of all the approaches examined, the semantically-enriched decision tree learner (STree) built consistently more accurate models of others. This is, we believe, as a result of the use of background domain knowledge in making appropriate generalisations/specialisations. When an agent encounters a new term, without domain knowledge, there is no way for the agent to make any useful inference regarding that term/concept on the basis of other terms/concepts. However, had the agent been aware of certain background domain information that relates to that term/concept then it could utilise ontological reasoning to see if there is a semantic link between the new term and any of the other terms that it had encountered before.

In a related work, [17] attempts to learn from, and classify partially specified datasets. In their work, the authors extended C4.5 decision trees [11] with attribute-value taxonomies (AVT). In their approach, classification is done using taxonomies defined over attribute values, which may be specified at different levels of precision. However, our approach is different from theirs because we directly incorporate existing domain ontologies and reason over domain knowledge during policy learning.

In recent research, Thomas *et al.* [14] propose the use of lightweight reasoners for reasoning about data on the Web. In their work, the authors introduced four possible language transformations which make reasoning on the Semantic Web more tractable. For example, for dealing with extremely large data sets, the authors argue that an approximation to RDF-DL allows the data to be loaded very quickly into a single machine, and balances this with good query performance for simple conjunctive queries.

Overall, there is a rich body of work that is focused on developing reasoners for the Semantic Web (e.g., TrOWL [13], Fact++ [15], Pellet [12]). However, it is worth noting that the focus of this paper is not to develop a reasoner nor investigate the performance of existing reasoners, rather our aim is to show that ontological reasoning can be employed in learning others' policies and that such reasoning could benefit human decision makers in plan resourcing.

In future, we plan to extend other machine learning methods with domain knowledge and explore how much this extension improves policy learning and enhances agents' support for human decision-making.

6 Conclusions

We have presented an agent decision-making mechanism where models of other agents are refined through argumentation-derived evidence from past dialogues, and these models are used to guide future plan resourcing decisions. We have also empirically evaluated our approach and shown that accurate models of others' policies could be learned by exploiting ontological reasoning. We believe that this research contributes both to the understanding of how policies may affect

plan resourcing in collaborative activities, and applications of evidence derived from argumentation and background domain knowledge (through ontological reasoning) to support human decision-making.

References

1. Duda, R., Hart, P.: Pattern classification and scene analysis. John Wiley & Sons (1973)
2. Emele, C.D., Norman, T.J., Parsons, S.: Argumentation strategies for plan resourcing. In: Yolum, Tumer, Stone, Sonenberg (eds.) Proc. of 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011). pp. 913–920. Taipei, Taiwan (2011)
3. Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. Morgan Kaufmann, 2nd edn. (2006)
4. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall (2009)
5. Huynh, T., Jennings, N.R., Shadbolt, N.: An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems 13(2), 119–154 (2006), http://eprints.ecs.soton.ac.uk/12593/
6. Kagal, L., Finin, T.: Modeling conversation policies using permissions and obligations. Autonomous Agents and Multi-Agent Systems 14, 187–206 (April 2007), http://dl.acm.org/citation.cfm?id=1210460.1210468
7. Kollingbaum, M., Norman, T.: Norm adoption and consistency in the noa agent architecture. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) Programming Multi-Agent Systems, Lecture Notes in Computer Science, vol. 3067, pp. 169–186. Springer Berlin / Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-25936-7-9, 10.1007/978-3-540-25936-7-9
8. McBurney, P., Parsons, S.: Games that agents play: A formal framework for dialogues between autonomous agents. Journal of Logic, Language and Information 12(2), 315 – 334 (2002)
9. Mitchell, T.M.: Machine Learning. McGraw Hill (1997)
10. Norman, T.J., Reed, C.: A logic of delegation. Artificial Intelligence 174(1), 51–71 (2010)
11. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann (1993)
12. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semant. 5(2), 51–53 (2007)
13. Thomas, E., Pan, J.Z., Ren, Y.: TrOWL: Tractable OWL 2 Reasoning Infrastructure. In: the Proc. of the Extended Semantic Web Conference (ESWC2010) (2010)
14. Thomas, E., Pan, J.Z., Taylor, S., Ren, Y.: Lightweight Reasoning and the Web of Data for Web Science. In: Proc. of the 2nd Int'l Conf. on Web Science (WebSci 2010) (2010)
15. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006. pp. 292–297. Springer (2006)
16. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2nd edn. (2005)
17. Zhang, J., Honavar, V.: Learning decision tree classifiers from attribute value taxonomies and partially specified data. In: Proceedings of the International Conference on Machine Learning (2003)