Learning policy constraints through dialogue

Chukwuemeka David Emele,
Timothy J. Norman, Frank Guerin
Department of Computing Science,
University of Aberdeen, Aberdeen. AB24 3UE. UK
{c.emele, t.j.norman, f.guerin}@abdn.ac.uk

Simon Parsons
Dept. of Computer & Information Science,
Brooklyn College, City University of New York,
2900 Bedford Avenue, Brooklyn,11210 NY, USA.
parsons@sci.brooklyn.cuny.edu

Abstract

An understanding of the policy and resource availability constraints under which others operate is important for effectively developing and resourcing plans in a multi-agent context. Such constraints (or norms) are not necessarily public knowledge, even within a team of collaborating agents. What is required are mechanisms to enable agents to keep track of who might have and be willing to provide the resources required for enacting a plan by modeling the policies of others regarding resource use, information provision, etc. We propose a technique that combines machine learning and argumentation for identifying and modeling the policies of others. Furthermore, we demonstrate the utility of this novel combination of techniques through empirical evaluation.

1. Introduction

Many problem solving scenarios usually require the formation of a team of collaborating agents. Members of the team agree to collaborate and perform joint activities in a mutually acceptable fashion. Often, agents in the team represent different organisations, and so there are different organisational constraints imposed on them. Even within a single organisation, team members often represent sub-organisations with different procedures and constraints. Examples of such constraints are constraints due to policies that guide the behaviour of representatives of organisations (or sub-organisations). Furthermore, team members may possess individual interests and goals that they seek to satisfy, which are not necessarily shared with other members of the team. These individual motivations largely determine the way in which members carry-out tasks assigned to them while engaging in joint activities.

In this paper, we focus on policy and resource availability constraints of coalition members, and define policy constraints as explicit permissions and prohibitions that members of the coalition are required to adhere to (often referred to as norms (Vasconcelos, Kollingbaum, and Norman 2007)). These policy constraints may be coalition-wide or individual. Coalition-wide policies are norms guiding the operations of the team as a whole and are expected to be public knowledge within the team. On the other hand, individual policies are often private to that individual member or subset of the coalition. In order to develop effective plans, an understanding of the policy and resource availability constraints of other members in the coalition is beneficial. However, tracking and reasoning about such information is non-trivial.

We envisage a system of agent support for human teams in which software agents aid the decision making of team members during collaborative planning (Sycara et al. 2009). One area of support that has been identified as important in this context is guidance in making policy-compliant decisions. This prior research focuses on giving guidance to humans regarding their own policies. An important and open question, however, is how can agents support human decision makers in developing models of others' policies and using these in guiding the decision maker?

We advocate a system where agents learn from practical dialogue by automatically extracting useful information from the dialogue and using these to model the policies, preferences and priorities of others in order to adapt their behaviour in the future. We, therefore, propose a technique that combines machine learning and argumentation for identifying and modeling the policies of others. We describe an experimental framework and present initial results of our evaluation which shows that an argumentation-based mechanism combined with a standard machine learning technique out-performs the machine learning technique on its own.

The remainder of this paper is organised as follows: In section 2 we introduce the problem domain. Learning with argumentation is discussed in section 3 and section 4 briefly describes our simulation environment. Experimental results are reported in section 5 and section 6 discusses related work and future direction. The paper is concluded in section 7.

2. Problem Domain

Planning for joint action in team problem solving activities is a complex problem on its own, and is further complicated by the various constraints that members of the team may have. Our conjecture is that machine learning techniques may be employed to aid decision making in this regard. Although this is not a new claim (Kelemen, Liang, and Franklin 2002), it is novel to combine it with argumentation analysis. Employing machine learning alone would enable the development of a model of the other agent but in a domain where there are underlying constraints that could yield similar results, standard machine learning techniques will have limited efficacy (See Examples 2 and 3). We present a technique for obtaining additional evidence (through dialogue) to indicate what constraints others may be operating with. This additional evidence serves to improve the quality of the models of the other agents that can be inferred from their observable actions. It is worth noting that we do not attempt to discuss planning but assume that the plans have been generated and need to be resourced.

For the sake of this paper, we define argumentation as the process whereby arguments are exchanged and evaluated in the light of their interactions with other arguments. By arguments, we refer to explanations/justifications offered in support of an action. Consider the following snippet of dialogue that may occur between two agents i and j:

Example 1:
i: Can I have R1? j: No.

What can be inferred from the interaction? Why did agent j say no to agent i's request?

1. Could it be that there exists some policy X that forbids agent j from providing R1 to agent i?
2. Could it be that R1 is not available at the moment?

There is very little that we can learn from the dialogue. On the other hand, suppose we have an argumentation framework that allows agents to ask for and receive explanations as in examples 2 and 3 below then agent i can gather more evidence regarding why agent j did not provide R1.

Example 2:	**Example 3:**
i: Can I have R1? j: No. i: Why? j: I'm not permitted to release R1.	i: Can I have R1? j: No. i: Why? j: R1 is not available.

From the above snippets, we see that example 1 is not very helpful in terms of learning the underlying constraints of agent j. However, employing argumentation, agent i could disambiguate the reason for the refusal, that is, whether it is due to policy constraints or the resource is not available.

Integrating learning techniques into the agent support framework will provide a level of support to human decision makers. However, can the use of argumentation improve the effectiveness and accuracy of the information learned about the policy constraints of others? We claim that significant improvements can be achieved because argumentation can help clarify reasons behind decisions made by the provider.

Hypothesis Allowing agents to exchange arguments during practical dialogue will mean that the proportion of correct policies learned during interaction will increase faster than when there is no exchange of arguments.

3. Learning with Argumentation

Our multi-agent learning framework is based on the decision tree learner, which applies the C4.5 algorithm (Quinlan 1993) on a given set of examples and generates a decision tree model. C4.5 builds decision trees from a set of training data, using the concept of information entropy (Mitchell 1997) (beyond the scope of this paper). The training data is a set $S = s_1, s_2, ..., s_n$ of already classified samples. Each sample $s_i = x_1, x_2, ..., x_m$ is a vector where $x_1, x_2, ..., x_m$ represent attributes of the sample. The training data is augmented with a vector $C = c_1, c_2, ..., c_n$ where $c_1, c_2, ..., c_n$ represent the class to which each sample belongs. Agent policies are represented as a vector of attributes (e.g. resource, purpose, location, etc.) and the C4.5 algorithm is used to classify each policy instance into a class.

The C4.5 algorithm has three base cases.

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

In pseudocode the C4.5 algorithm is outlined in Figure 1.

```
Step 1.  Check for base cases
Step 2.  For each attribute D,
           Find the normalized information gain from
           splitting on D
Step 3.  Let D_best be the attribute with the highest
         normalized information gain
Step 4.  Create a decision node that splits on D_best
Step 5.  Recurse on the sublists obtained by splitting on
         D_best , and add those nodes as children of node
```

Figure 1: The C4.5 algorithm (Kotsiantis 2007).

Once a model is built, an agent attempts to predict the policies of the other agent from the observed actions of that agent (assuming it abides by its policies). Over a number of encounters if there is significant variance between the predicted and the observed actions then the agent seeks additional evidence (through dialogue) to disambiguate whether the deviation was as a result of policy or resource availability constraints. This additional evidence serves to improve the quality of the models of the other agents that can be inferred from their observable actions.

To achieve this, we have developed a simple dialogue game[1] involving two players. The players take turns (Levin and Moore 1980; Walton and Krabbe 1995) and a move in the game involves issuing an utterance and may refer to a commitment to perform an action. The game starts with an agent, i, sending a request to another agent, j, for the use of some resources needed to fulfill a plan. The other agent

[1]Dialogue games have proven extremely useful for modeling various forms of reasoning in many domains, including law (Bench-Capon et al. 2003), and medicine (Perrussel et al. 2007).

Let $\mathcal{A}$ be the set of agents in the domain such that $i, j \in \mathcal{A}$. **A resource allocation** , denoted as Λ_i^t is a collection of resources that an agent, i has at its disposal at time t, where t denotes the time step in the dialogue.

Assume agent i has a plan (a subset of the joint plan) that requires the use of a set of resources $\mathcal{R}$ to achieve a goal G.

```
1. At time $t = 0$, agent $i$ starts with initial allocation $\Lambda_i^0$,
    request($i$, $j$, $r$): agent $i$ requests $j$ to provide resource $r$,
    where $r \in \mathcal{R}$ and has not been requested of $j$ before.
2. At the next time step, agent $j$ either:
   (a). agree($j$, $i$, $r$): agrees, and resource $r$ is allocated to $i$.
   (b). refuse($j$, $i$, $r$): refuses, and $r$ is not allocated to $i$.
3. At the next time step,
    if last received locution was agree($j$, $i$, $r$) then agent $i$
        records the evidence and moves to step 6.
    otherwise (switches to argumentation-based dialogue.)
      why($i$, $j$, refuse($r$)): $i$ asks $j$ for underlying interests
        or reasons why it has refused to provide resource $r$.
4. At the next time step,
    if last received locution was why($i$, $j$, $r$) then $j$ either:
        (a). inform($j$, $i$, $r$, reason($x$)): gives the reason for
            refusing to provide $r$ to $i$; or
        (b). inform($j$, $i$, $r$, wont-tell): gives no reason for
            refusing to provide $r$ to $i$.
    otherwise inform($j$, $i$, $r$, invalid-message): informs $i$
            that the message is invalid in this context.
5. At the next time step,
    if last received locution was inform($j$, $i$, $r$, reason($x$))
        then $i$ records the evidence and moves to step 6.
    otherwise move to step 6.
6. At the next time step,
   if there are resources in $\mathcal{R}$ that are yet to be requested
     then move to step 1 with the current allocation.
   otherwise
     close-dialogue($i$, $j$): terminate the dialogue.
```

Figure 2: Dialogue Game Protocol

(j) responds with an agree or refuse based on the prevailing context, e.g. policy constraints. The requesting agent could ask for explanations and reasons for an action, and so on until the game ends.

Figure 2 outlines the protocol for the dialogue game developed in this work and Figure 3 shows two examples of the kind of dialogue that may occur between two agents, i and j using the protocol. Note that although this is presented as a dialogue between two agents, in reality the initiator (agent i, the agent that wishes to resource its plan) may engage in multiple instances of this dialogue with other agents.

	Example A
i:	**request**(i, j, Jeep)
j:	**refuse**(j, i, Jeep)
i:	**why**(i, j, refuse (Jeep))
j:	**inform**(j, i, Jeep, reason(resource-unavailable))
i:	**close-dialogue**(i, j)

	Example B
i:	**request**(i, j, Helicopter)
j:	**refuse**(j, i, Helicopter)
i:	**why**(i, j, refuse (Helicopter))
j:	**inform**(j, i, Helicopter, wont-tell)
i:	**request**(i, j, Van)
j:	**agree**(j, i, Van)
i:	**close-dialogue**(i, j)

Figure 3: Two simple dialogues between agents i and j

4. Simulation Environment

Each agent has two main layers, the communication layer and the planning and reasoning layer (See Figure 4). The communication layer embodies the dialogue controller, which handles the communication with other agents in the domain. The planning and reasoning layer consists of three modules: the planner, the policy modeler, and the learner. The planning module of the agent uses some heuristics (beyond the scope of this paper) to generate a plan that is consistent with the individual policies of the agent. For the sake of brevity, the term plan will be used to mean a complete plan, partial plan, and/or a plan step. Once a plan of action is constructed, the agent is ready to communicate with other agents in the domain to identify and gain commitments for the resources required to execute the plan. The dialogue controller module sends (and receives) messages to (and from) other agents and reasons over the dialogue. The policy modeller looks up policy constraints from the knowledge-base and generates the appropriate utterance (or action) for the agent. Policy constraints are stored in the policy constraint knowledge-base while other (non-policy) constraints (e.g. resource constraints) are captured in the "other constraints" knowledge-base. The learner uses decision trees to learn policies based on the perceived actions of other agents.

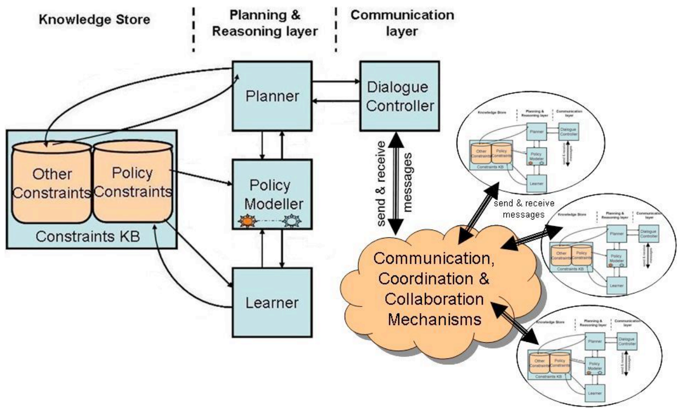

Figure 4: Architecture of the framework for learning policy constraints in team-based activities using dialogue.

Implementation

To test our hypothesis, we have developed a simulation environment for agent support in coalition missions and integrated our learning and argumentation mechanisms into the framework. The policies are captured as rules and implemented in a post production engine (Friedman-Hill 2003). The application programming interface in Weka (Witten and Frank 2005) was used to integrate a machine learning algorithm into the framework. We note that, although decision trees were used, the framework is configured such that other machine learning algorithms can be plugged in.

Agents in this domain play the role of a seeker, provider, or both in different interactions. For simplicity, we consider a setup with one seeker and one provider. The seeker simulates an agent that has a plan (the planning mechanism of the agent is beyond the scope of this paper) and needs to resource this plan. The plan is resourced by convincing a provider to commit resources from its resource pool. The seeker identifies the resources required for a task and sends a request message to the provider. The provider evaluates the request and responds accordingly. If the resources are available for use and releasing them to the seeker does not conflict with the policies of the provider then the provider agrees, otherwise it refuses. By availability we mean the resource is not committed to another task (or agent) at the time requested and the resource is in a usable state.

Suppose a seeker A sends a request for resource R to provider P then we can represent the decision function of the provider generally as follows:

```
IF     ( is_available(R) ∧ NOT (forbidden(release(R, A)) )
THEN   agree( release(R, A))
ELSE   refuse( release(R, A))
```

Figure 5: A simple decision function

Policies

The providers operate under a set of policies which govern how resources are deployed to others. In other words, the provider can make resources available to other agents if the resources are available and there is no policy forbidding that course of action. The policies in this framework are based on the following factors:

- Organisation - refers to the organisation of the requesting agent. An agent represents an organisation, therefore, the policies associated with that organisation are enacted whenever that agent makes a request for resources.
- Resources - generally denote physical equipment, capabilities or information that are required to carry out a task.
- Purpose - indicates the purpose for which the resource is being requested. For example, the provider may be obliged to release any resource to a member of the coalition if the resource is required for reconnaissance.
- Location - denotes the particular location or zone where the resource is to be deployed.
- Day - refers to the day the resource is to be deployed.

You are permitted to release resource R to team member X
if his affiliation is O and the resource is to be deployed
at location L for purpose P on day D.

Figure 6: An example of a policy

5. Experiments and Results

A variety of experiments were conducted to test the performance and behavior of our framework. In this section, we describe our experimental scenario and present the results.

Experimental scenario

We present an illustrative scenario that will serve as a vehicle for testing our hypothesis. The scenario involves two software agents collaborating to complete a joint activity in a region over a period of three days. The region is divided into five zones/locations. There are five resource types and five purposes that a resource could be used to fulfill. A task involves the seeker identifying resource needs for a plan and collaborating with the provider to see how that plan can be resourced.

For the purpose of the experiment, the seeker simulates an agent that has a plan and needs to collaborate with the provider to resource it. The seeker predicts (based on the model of the provider) whether the provider has a policy that forbids/permits the provision of such resource in that context. The seeker requests for the resource from the provider and the provider uses a simple decision function (described earlier) to decide whether to grant or deny the request. The dialogue follows the protocol specified in Figure 2 and at the end of the interaction the outcome is learned by the seeker and the model of the provider is updated accordingly.

Three agent support configurations were investigated and the performance of the seeker was evaluated. The configurations include:

i. Random Selection (RS): Here, the seeker does not employ any machine learning nor argumentation technique, rather it randomizes its choice of attributing the refusal to policy or resource availability constraints.

ii. Learning without Argumentation (LOA): In this setup, the seeker applies the C4.5 decision tree learner to learn the provider's policy. This setup does not use argumentation in any way.

iii. Learning with Argumentation (LWA): Here, argumentation is employed as a mechanism to augment the C4.5 learner in learning the policy of the provider. In other words, dialogue is used to gather additional evidence that serves to improve the quality of the models learned by disambiguating between underlying constraints that may have similar observable actions.

Results

This section presents the results of the experiments carried out to evaluate this work. Table 1 shows the average percentage of policies classified correctly and the standard deviations at the end of 6000 tasks.

Table 1: Average percentage of policies classified correctly at the end of 6000 tasks

Tasks	RS	LOA	LWA
1000	50.0 ± 1.8	57.3 ± 11.7	57.3 ± 11.7
2000	50.0 ± 1.4	68.0 ± 9.4	70.5 ± 10.3
3000	49.9 ± 0.8	73.7 ± 4.8	77.4 ± 6.1
4000	50.1 ± 1.7	74.5 ± 4.9	80.6 ± 5.0
5000	50.0 ± 1.6	73.9 ± 7.0	84.3 ± 3.5
6000	49.9 ± 2.0	67.9 ± 5.0	84.3 ± 4.7

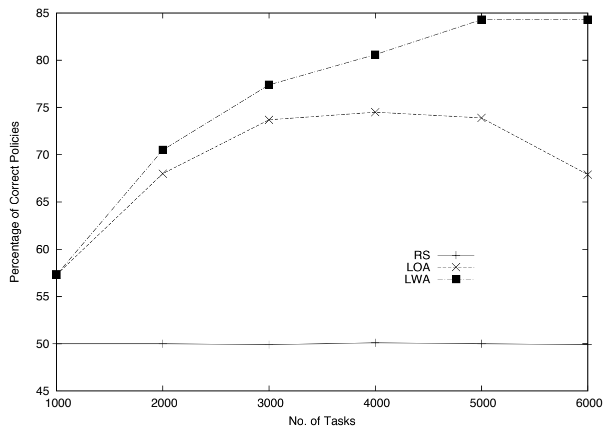

Figure 7: Graph showing the effectiveness of learning policies using the three configurations (RS, LOA & LWA).

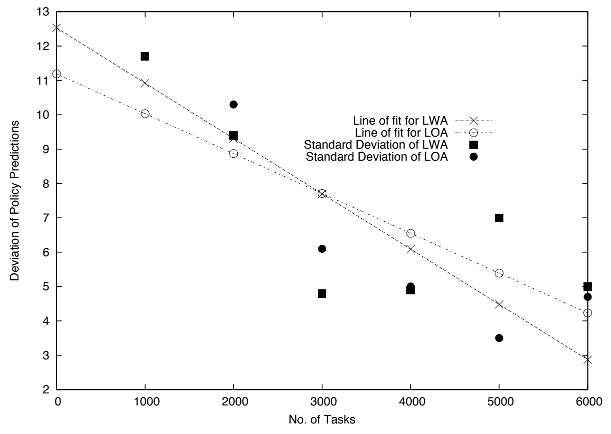

Figure 8: Graph showing the convergence of the policy predictions using LOA & LWA respectively.

Figure 7 illustrates the effectiveness of learning policies in the three configurations. These include: (1) random selection (RS), (2) standard machine learning approach only (LOA) and (3) combining machine learning with argumentation (LWA). It shows the percentage of the policies that the seeker predicted correctly in each configuration. The graph also shows that the argumentation-based approach enabled the agent to learn and build a more accurate model of the other agent's policies and thereby increased the accuracy of predictions. It is easy to see that the argumentation-based approach constantly out-performed the standard learning approach.

The standard deviations of the results were plotted and the trend line (using linear regression) shows that as the number of tasks increases, the argumentation-based approach ($y = 12.5333 - 0.0016x$) consistently converges at 95% confidence interval, with a F value of 18.9133 and significance $p = 0.0122$. (See Figure 8). On the other hand, with a significance $p = 0.0808$, there is no statistical significance as to whether the standard machine learning approach ($y = 11.1933 - 0.0012x$) converges or not.

6. Discussion and Related Work

To the best of our knowledge no other work has attempted to combine machine learning and argumentation in the way described in this paper. We have demonstrated the effectiveness of using argumentation enriched approaches to learn underlying social characteristics (e.g. policies) without assuming that those underlying features are public knowledge. Having said that, there are several related works that inspired this work. We discuss some of them in this section.

Rahwan et al. (2007) present a formal framework for analysing the outcomes of interest-based negotiation (IBN) dialogues and established that providing further information (especially about underlying interests) improves the likelihood and quality of an outcome. Policy constraints can be captured as underlying goals that agents are hoping to achieve (by adhering to them) and so argumentation can be used to tease out information regarding those constraints. In circumstances where knowledge is incomplete or imperfect, argumentation has proven to be effective in reaching some goals that would have otherwise been unreachable. It is worth noting that our work differs from Rahwan et al. (2007) in that while the authors are interested in gathering meta-information and using it to support interest-based negotiation, we are interested in learning the policies that other agents are operating with and using this knowledge to guide how a plan is resourced. Our framework neatly combines machine learning and argumentation in predicting what the other's policies are. Furthermore, our work is aimed at supporting human decision making in team-based activities.

Možina et al. (2007) combined machine learning with concepts of argumentation to produce a new machine learning technique called Argumentation-Based Machine Learning (ABML). With this framework, an expert can provide arguments for some learning examples and thereby enhance the predicting power of the learner. The work implemented an argument-based extension of CN2 rule learning (ABCN2) and was able to show that ABCN2 out-performed CN2 in most tasks. However, the framework is another kind of learning algorithm and will struggle to disambiguate between constraints that may produce similar outcome/effect and that is the main issue we are addressing in our work.

Atkinson and Bench-Capon (2007) treated reasoning about what action an agent should select as presumptive argumentation. The framework captured situations where the effect of an action is partially dependent upon the choices of another agent. In other words, an agent chooses a move, proposes presumptive reasons for the action and subjects it to critiquing in order to establish suitability or otherwise. This kind of framework is useful in our work as we argue that policy constraints impact on the behaviour (or action) of an agent and that, in turn, could be learned and used to infer what the policies of that agent are.

In our future work, we plan to develop strategies for advising human decision makers on how a plan may be resourced and who to talk to on the basis of policy and resource availability constraints learned (Oren, Norman, and Preece 2006). Parsons et al. (2003) investigated the properties of argumentation-based dialogues and examined how different classes of protocols can have different outcomes. We plan to explore ideas from this work to see which class of protocol will yield the "best" result in this kind of task. We are hoping that some of these ideas will drive the work on developing strategies for choosing who to talk to (and also which class of protocol to present first, and so on). Furthermore, we plan to incorporate the ability to suggest alternative resources based on the preferences of team members and to see what effect this will have on the learning of policies.

7. Conclusions

In this paper, we have presented a technique that combines machine learning and argumentation for learning policies in a team of collaborating agents engaging in joint activities. Individual policies are private but through the use of argumentation, we have been able to tease out certain information that can improve the performance in learning the policies of other agents. In our approach, an argumentation layer was built over a standard learning mechanism such that dialogue interactions enabled our agents to disambiguate between resource and policy constraints, thereby fine-tuning the policies learned. We have also shown that integrating argumentation into systems empowers agents with incomplete or imperfect knowledge to (potentially) perform better than they would have without argumentation in learning.

Acknowledgements

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.

References

Atkinson, K., and Bench-Capon, T. 2007. Action-based alternating transition systems for arguments about action. In *Proc. of the 22nd Conference on Artificial Intelligence (AAAI 2007)*, 24–29. Vancouver, Canada: AAAI Press.

Bench-Capon, T. J. M.; Freeman, J. B.; Hohmann, H.; and Prakken, H. 2003. Computational models, argumentation theories and legal practice. In Reed, C., and Norman, T. J., eds., *Argumentation Machines. New Frontiers in Argument and Computation*, 85–120. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Friedman-Hill, E. 2003. *Jess in Action*. Manning.

Kelemen, A.; Liang, Y.; and Franklin, S. 2002. A comparative study of different machine learning approaches for decision making. In Mastorakis, N. E., ed., *Recent Advances in Simulation, Computational Methods and Soft Computing*, 181–186. Piraeus, Greece: WSEAS Press.

Kotsiantis, S. B. 2007. Supervised machine learning: A review of classification techniques. *Informatica* 31(3):249–268.

Levin, J., and Moore, J. 1980. Dialogue-games: meta communication structure for natural language interaction. *Cognitive Science* 1(4):395–420.

Mitchell, T. M. 1997. *Machine Learning*. McGraw Hill.

Možina, M.; Žabkar, J.; and Bratko, I. 2007. Argument based machine learning. *Artif. Intell.* 171(10-15):922–937.

Oren, N.; Norman, T. J.; and Preece, A. 2006. Loose lips sink ships: A heuristic for argumentation. In *In Proceedings of the Third International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2006*, 121–134.

Parsons, S.; Wooldridge, M.; and Amgoud, L. 2003. Properties and complexities of some formal inter-agent dialogues. *Journal of Logic and Computation* 13(3):347–376.

Perrussel, L.; Doutre, S.; Thevenin, J.; and McBurney, P. 2007. A persuasion dialog for gaining access to information. In *Proc. of the AAMAS International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2007).*

Quinlan, J. R. 1993. *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rahwan, I.; Pasquier, P.; Sonenberg, L.; and Dignum, F. 2007. On the benefits of exploiting underlying goals in argument-based negotiation. In *Proc. of the 22nd International Conference on Artificial Intelligence (AAAI).* California, USA: AAAI Press.

Sycara, K.; Norman, T. J.; Giampapa, J. A.; Kollingbaum, M. J.; Burnett, C.; Masato, D.; McCallum, M.; and Strub, M. H. 2009. Agent support for policy-driven collaborative mission planning. *The Computer Journal* bxp061.

Vasconcelos, W.; Kollingbaum, M. J.; and Norman, T. J. 2007. Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proc. of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007).*

Walton, D. N., and Krabbe, E. C. W. 1995. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. Albany, NY, USA: SUNY Press.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann, 2nd edition.