Agents that learn how to generate arguments from other agents

Ariel MONTESERIN and Analía AMANDI

ISISTAN Research Institute, CONICET - UNCPBA, Campus Universitario, Tandil, Bs. As. Argentina

ariel.monteserin, analia.amandi@isistan.unicen.edu.ar

Received 11 October 2012 - Revised 29 April 2013

Abstract Learning how to argue is a key ability for a negotiator agent. In this paper, we propose an approach that allows agents to learn how to build arguments by observing how other agents argue in a negotiation context. Particularly, our approach enables the agent to infer the rules for argument generation that other agents apply to build their arguments. To carry out this goal, the agent stores the arguments uttered by other agents and the facts of the negotiation context where each argument is uttered. Then, an algorithm for fuzzy generalized association rules is applied to discover the desired rules. This kind of algorithm allows us (a) to obtain general rules that can be applied to different negotiation contexts; and (b) to deal with the uncertainty about the knowledge of what facts of the context are taken into account by the agents. The experimental results showed that it is possible to infer argument generation rules from a reduced number of observed arguments.

Keywords
Intelligent Agents,
Argumentation-based Negotiation,
Fuzzy Generalized Association Rules,
Learning.

§1 Introduction

In multi-agent environments, autonomous agents need to interact to achieve their goals because reciprocal dependencies exist among them. In this context, negotiation is a fundamental tool to reach agreements among agents with conflicting goals. The essence of a negotiation process is the exchange of proposals. Agents make and respond to proposals in order to converge towards a mutually acceptable agreement. However, not all approaches are restricted to that exchange. Several approaches to automated negotiation have been developed. One of them is the argumentation-based approach [15, 26, 24, 22, 5, 9]. In argumentation-based approaches, agents are allowed to exchange some additional information as arguments, besides the information uttered on the proposal [24]. Thus, in the negotiation context, an argument is seen as a piece of information that supports a proposal and allows an agent either (a) to justify its position in the negotiation, or (b) to influence other agents' position in the negotiation [12].

When a conflict arises during the negotiation, an agent must observe the negotiation context and determine what arguments can be uttered in order to reach an agreement. At this point, argument generation can be carried out in several ways. One of them involves using explicit rules [15, 24]. A rule for argument generation establishes a set of conditions that the negotiation context must fulfil to generate a given argument. For instance, if we want to generate a reward, we need to know what the target of the argument would like to receive in exchange for its proposal acceptance. In formal terms, to generate a reward, an agent a_i, which needs to persuade an opponent a_j, has to observe in the negotiation context a goal g_{j1} that must be achieved by a_j and an action t_A that produces the fulfilment of such goal g_{j1}. Thus, if the agent finds these facts in the negotiation context, it can generate a reward by saying: "if you (a_j) accept my proposal, I promise you to perform action t_A".

In most argumentation-based negotiation frameworks, the rules for argument generation are defined in design time. However, no techniques are defined to learn from other agents how the agent must build arguments. In fact, people learn to build arguments from experience. That is, based on the arguments that a person receives from others and the context in which these arguments are uttered, he/she is able to infer how the arguments are built and what context facts are included in the conditions needed to generate those arguments.

In this paper, we propose an approach that allows an agent to learn how to build arguments by observing how other agents argue in a negotiation context.

Specifically, our approach focuses on how to learn rules to generate rhetorical arguments. To infer these rules, we utilise an algorithm for mining fuzzy generalized association rules [11]. Performing this algorithm, we obtain rules that associate a set of conditions (antecedents) with an argument (consequent), such is the format of the argument generation rules that we intend to find.

Mining association rules allows us to obtain rules from a set of transactions stored in a database. As introduced by Agrawal et al. [2], given a set of transactions, where each transaction is a set of items, an association rule is an expression $X \rightarrow Y$, where X (antecedent) and Y (consequent) are also sets of items. That is, the transactions in the database which contain the items of X will also contain the items of Y. So, if an agent observes and stores in a database the arguments generated by another agent and the context facts that can be part of the conditions to generate these arguments, the agent will be able to infer the rules for argument generation that the other agent applies.

However, the question arises: is it sufficient to utilise a traditional (crisp) algorithm for mining association rules (e.g. Apriori algorithm [2]) to infer argument generation rules? We have concluded that it is not. There are two factors that determine this answer: generality and uncertainty.

First, we want the agent to apply the learned rules for argument generation in different negotiation contexts, that is, we need to obtain general rules. Nevertheless, arguments and facts observed by the agent are expressed in constant terms, because they were uttered in a particular negotiation context. So, if we utilise a traditional algorithm for mining association rules, the learned rules will also be expressed in constant terms. For this reason, we opt for an algorithm for mining generalized association rules. These algorithms use the existent hierarchical taxonomy of the data to generate different association rules at different levels in the taxonomy [28]. Our aim is to build a hierarchical taxonomy of conditions and arguments in which the leaves are the facts of the negotiation context (conditions) and arguments observed by the agent, and the upper levels are the same propositions but expressed in variable terms with different degrees of generality. Then, the generalised association rules algorithm will especially be able to generate rules at upper levels in the taxonomy of conditions and arguments. Therefore, the rules will be variable.

The second problematic factor is uncertainty. An agent observing other agents during the negotiation can only be certain of the arguments uttered, but cannot be sure of the conditions that these agents check to generate such

arguments. The reason behind this fact is that usually it is not necessary to include all the information used to generate an argument in its premises and that agents maintain their information private. Despite this uncertainty, the agent can access to information in the negotiation context that could be part of the conditions to generate an argument. For instance, following the previous example, the contextual information around the reward could be: agent a_j has goals g_{j1}, g_{j2} and g_{j3}; agent a_k has the goal g_{k1}; agent a_i knows that performing action t_A enables it to fulfil goal g_{j1}, that performing action t_B enables it to attain goal g_{j2}, and that performing action t_A enables it to fulfil goal g_{k1}. These facts are present in the negotiation context, but not all this information is necessary to generate the reward. For this reason, the agent has to differentiate relevant from irrelevant information. Thus, taking into account the information that can be extracted from the argument (e.g. a_j is the target, action t_A is the "reward"), we can determine if a piece of information is semantically related to the argument. For example, by observing the previous reward, we can see that the fact that agent a_i knows that performing action t_A enables it to fulfil goal g_{j1} is more related to the argument than the fact that agent a_i knows that performing action t_A enables it to fulfil goal g_{k1}, since action t_A is the action promised and goal g_{j1} is a goal of agent a_j (target of the argument); in contrast, goal g_{k1} is a goal of another agent not mentioned in the argument.

Therefore, we propose the use of a fuzzy approach for generalised association rules mining to handle this uncertainty. Mining fuzzy association rules is the discovery of association rules using fuzzy set concepts [17]. The fuzzy set theory [31] has been used more and more frequently in intelligent systems because of its simplicity and similarity to human reasoning [13]. Fuzzy sets are sets whose elements have degrees of membership. In the context of our work, we see the facts observed in the negotiation context when an argument is generated as a fuzzy set, where each fact has a degree of membership as regards the semantic relation between the fact and the argument. Thus, these sets of observed facts and arguments constitute fuzzy transactions [8]. Consequently, uncertainty is taken into account during the mining process.

The experimental results showed a high precision of the proposed approach. To determine the efficiency of our approach, we carried out three experiments. First, we compared the rules learned by using our approach with the original rules used by the observed agents. Second, we compared the rules learned by using our fuzzy approach with the rules learned by using a crisp

one, in order to assess the contribution of fuzziness to this problem. Finally, we compared the set of arguments that can be generated by using the original rules with the set of arguments that can be generated by using the rules learned by the fuzzy approach as well as by the crisp one.

The article is organised as follows. Section 2 introduces basic concepts about argumentation based negotiation. In Section 3, we present the approach for learning argument generation rules by observing how other agents argue. In Section 4, the results extracted from the experiments are presented. Section 5 places this work in the context of previous ones. Finally, in Section 6, we state our conclusions and suggest future work.

§2 Argumentation-based negotiation

In accordance with the work of Rahwan et al. [23], there are two major strands in the literature on argumentation-based negotiation: (a) attempts to adapt dialectical logics for defeasible argumentation by embedding negotiation concepts within them [21, 4]; and (b) attempts to extend bargaining-based frameworks by allowing agents to exchange rhetorical arguments, such as promises and threats [15, 24]. Our work belongs to the second strand.

There are several types of rhetorical arguments that an agent can exchange during the negotiation. Such types have been commonly studied in the field of persuasion in human negotiation [14, 20]. Based on these studies, the current literature identifies at least six types of arguments that an agent can use during the negotiation [15, 24, 3]. These types are: rewards, used to promise a future reward; threats, used to warn about negative consequences in case the counterpart does not accept a proposal; and appeals, used to justify a proposal. Particularly, these appeals can be: appeal to a past reward, to remind an opponent about a past reward; counterexample, to convey the persuadee a contradiction between what it says and past actions; appeals to prevailing practice, to persuade the opponent that a proposal will further its goals since it has furthered others' goals in the past; and appeal to self-interest, to convince a persuadee that accepting a proposal will enable achievement of a goal. In general terms, a rhetorical argument is composed of four elements: a sender, a receiver, a conclusion that normally represents the proposal that the argument supports, and a set of premises that support the conclusion [26].

In an argumentation-based negotiation approach, agents can exchange arguments in order to justify their proposals, to persuade their opponent, and

to reach an expected agreement. In addition to evaluating and generating proposals, agents with the ability for argumentation must be able to (a) evaluate incoming arguments and update its mental state as a result; (b) generate candidate outgoing arguments; and (c) select an argument from the set of candidate arguments [6]. As mentioned above, we will focus on argument generation.

Argument generation is related to the generation of candidate arguments to present to a counterpart. To this end, rules for argument creation are defined (e.g. [15, 24]). Such rules specify conditions for argument generation. Thus, if the condition is satisfied in the negotiation context, the argument may be generated and it becomes a candidate argument. Normally, these rules are defined explicitly. However, we claim that it is possible to learn them by observing how other agents argue in a negotiation context.

Since several frameworks for argumentation-based negotiation that use rules for argument generation have been defined, the aim of our proposal is to define a general approach to infer such rules that can be used by any framework. However, each framework maintains a specific formal model. For instance, Kraus et al. [15] developed a formal logic that forms a basis for the development of a formal axiomatization system for argumentation. They proposed a logical model of the mental state of an agent based on a representation of its beliefs, desires, intentions, and goals by using a modal BDI logic [25]. In contrast, Ramchurn et al. [24] defined a simple logical language to define agents' mental states, actions and illocutionary acts (e.g. rhetorical arguments) based on the framework proposed by Sierra et al. [26]. In this context, we specify an approach which is neutral with respect to the argumentation-based negotiation framework. For this reason, we define a simple negotiation language that is neutral with respect to the underlying semantic of the argumentation model. The following sections describe the language of negotiation used during the experiments and the standard argument generation rules defined by Kraus et al. [15] that we use as example.

2.1 Negotiation language

The negotiation language L used by the agents to exchange proposal and arguments during the negotiation is composed of the following predicates[*1]:

- *goal(G)*: G is a goal.
- *hasgoal(A, goal(G))*: A pursues goal G. Agent A has G in its goals.

[*1] Predicates are expressed using *Prolog* syntax.

- *believe(A, B)*: A believes B. Agent A has B in its beliefs.
- *prefer(A, goal(G_1), goal(G_2))*: A prefers to fulfil G_1 over G_2.
- *accept(P)*: acceptation of proposal P.
- *reject(P)*: rejection of proposal P.
- *imply(Q, R)*: Q implies R (It represents the classical inference).
- *pastpromise(A_1, A_2, do(A_1, Action))*: A_1 promised to do *Action* to A_2, but has not fulfilled it yet.
- *do(A, Action)*: A executes *Action*. *Action* can be instantiated with *accept(P)* or *reject(P)*.
- *wasgoal(A, goal(G))*: A pursued goal G in the past.
- *did(A, Action)*: A performed *Action* in the past.
- *appeal(A_1, A_2, do(A_2, Action), [Just])*: A_1 uses an appeal to persuade A_2. The goal of the argument is to support a proposal *do(A_2, Action)* by using a set of justifications *Just*.
- *reward(A_1, A_2, do(A_2, $Action_1$), [do(A_1, $Action_2$)])*: A_1 uses a promise of a future reward to persuade A_2. A_1 promises to execute $Action_2$ if A_2 executes $Action_1$.
- *threat(A_1, A_2, do(A_2, $Action_1$), [do(A_1, $Action_2$)])*: A_1 uses a threat to persuade A_2. A_1 warn about negative consequences (the execution of $Action_2$) in case the counterpart does not accept to execute $Action_1$.

Moreover, other propositions strongly related to the domain exist, especially those related to goals, proposals and actions the agent can execute. For instance, in the domain of meeting scheduling, the extra propositions are:

- *discusstopic(T)*: T is a topic that can be discussed in the meeting.
- *inplace(P)*: the meeting can take place in P.
- *date(D)*: the meeting can be on date D.
- *time(S)*: the meeting can be at time S.

In this way, *hasgoal(a_1, goal(discusstopic($topic_1$)))* represents the a_1*'s* goal of discussing $topic_1$ in a meeting. It is worth noticing that some notions related with the argument evaluation and argument selection processes are out of the scope of this language. This is because we want to keep the focus on the predicates used by the argument generation process. For example, the trust in the opponent is a key concept used by the argument selection process [19]: if the trust in the opponent is high then the agent will prefer to utter weak arguments (appeals instead of threats) as long as arguments of these types had been previously generated. Moreover, the trust values are updated by the evaluation process

when a promise is fulfilled or a request is accepted, among other situations. In such cases, new predicates that represent this information can be incorporated to L without affecting our approach.

2.2 Argument generation rules

As mentioned above, it is possible to obtain different kinds of appeal by modifying how they are justified. The rules are:

- Appeal to past promise:
  - Conditions: *pastpromise(Y, X, do(Y, Action))*
  - Argument: *appeal(X, Y, do(Y, Action), [pastpromise(Y, X, do(Y, Action))])*
- Appeal to self interest:
  - Conditions: *hasgoal(Y, goal(Goal)), believe(X, imply(do(Y, Action), Goal))*
  - Argument: *appeal(X, Y, do(Y, Action), [imply(do(Y, Action), Goal), hasgoal(Y, goal(Goal))])*
- Appeal to prevailing practice:
  - Conditions: *hasgoal(Y, goal(Goal)), believe(Y, imply(do(Y, Action), not(Goal))), wasgoal(Z, goal(Goal)), did(Z, Action)*
  - Argument: *appeal(X, Y, do(Y, Action), [wasgoal(Z, goal(Goal)), did(Z, Action)])*
- Counterexample:
  - Conditions: *hasgoal(Y, goal(Goal)), believe(Y, imply(do(Y, Action), not(Goal))), believe(Y, imply(do(Y, ActionB), not(Goal))), wasgoal(Y, goal(Goal)), did(Y, ActionB)*
  - Argument: *appeal(X, Y, do(Y, Action), [did(Y, ActionB), imply(do(Y, ActionB), not(Goal))])*

In addition, we define two rules to generate rewards and threats.

- Reward (observing the goals of the sender agent):
  - Conditions: *hasgoal(Y, goal(Goal)), believe(Y, imply(do(X, ActionR), Goal)), hasgoal(X, goal(Goal2)), believe(X, imply(do(Y, ActionP), Goal2))*
  - Argument: *reward(X, Y, do(Y, ActionP), [do(X, ActionR)])*
- Threat:
  - Conditions: *hasgoal(Y, goal(GoalA)), hasgoal(Y, goal(GoalB)),*

prefer(Y, goal(GoalA), goal(GoalB)), believe(X, imply(do(X, ActionT), not(GoalA))), believe(X, imply(do(Y, ActionP), not(GoalB)))

– Argument: *threat(X, Y, do(Y, ActionP), [do(X, ActionT)])*

In order to clarify our proposal, we have only defined actions for the main arguments. Nevertheless, other rules may be defined by changing the conditions. For example, we can define an additional appeal to self interest by changing the agent that has the belief, or an additional counterexample by changing the goals.

- Additional appeal to self interest:
  – Conditions: *hasgoal(Y, goal(Goal)), believe(Y, imply(do(Y, Action), Goal))*
  – Argument: *appeal(X, Y, do(Y, Action), [imply(do(Y, Action), Goal), hasgoal(Y, goal(Goal))])*
- Additional counterexample:
  – Conditions: *hasgoal(Y, goal(not(Goal))), believe(Y, imply(do(Y, Action), Goal)), believe(Y, imply(do(Y, ActionB), Goal)), wasgoal(Y, goal(not(Goal))), did(Y, ActionB)*
  – Argument: *appeal(X, Y, do(Y, Action), [did(Y, ActionB), imply(do(Y, ActionB), Goal)])*

§3 Learning argument generation rules

As mentioned above, we propose to use an algorithm for mining fuzzy generalized association rules to learn argument generation rules. This kind of algorithms allows us (a) to obtain rules with different degrees of generality that can be applied by the agent in any negotiation context; and (b) to take into account the uncertainty about the fact that should be part of the conditions of such rules.

We decide to use an algorithm for mining fuzzy generalized association rules rather than other soft-computing approaches (for example, probabilistic graphical models) for two reasons. First, as we stated above, this approach allows us to deal with the problems of generality and uncertainty. Second, the association rule format matches perfectly with the format required by an argumentation-based negotiation framework using rules for argument generation. In contrast, probabilistic graphical models can also deal with uncertainty, but cannot deal with generality, at least in an intuitive way. In addition, if we use a probabilistic graphical model, we should define a *middleware* to translate the information of the model into rules for argument generation.

In this section, we will first present a brief description of the algorithm for mining fuzzy generalized association rules, and then we will detail how to learn argument generation rules from a set of observed arguments.

3.1 Mining fuzzy generalized association rules

Mining fuzzy association rules is the discovery of association rules by using the concept of fuzzy set [17]. The fuzzy set theory [31] has been more and more used in intelligent systems because of its simplicity and similarity to human reasoning [13]. Fuzzy sets are sets whose elements have degrees of membership. In the context of our work, we assume that the set of observed facts is a fuzzy set, where each fact has a membership value that takes into account a presumed semantic relationship between the observed fact and the argument. Thus, the set of observed facts and the argument constitute a fuzzy transaction [8]. We will briefly describe the basic concepts about mining fuzzy generalized association rules below.

[1] Mining traditional (crisp) association rules

As introduced in [2], let $D = \{t_1, ..., t_n\}$ be a transactional database and t_i represent the i^{th} transaction in D. Moreover, $I = \{i_1, ..., i_m\}$ represents all attributes or items appearing in D and i_j represents the j^{th} item. Then, each transaction t_i is a set of items belonging to I, and an association rule is an expression $X \rightarrow Y$, where X and Y are also sets of items (itemset). That is, the transactions in the database which contain the items of X will also contain the items of Y. That is assured by computing the support and the confidence of the rule. Support and confidence are the main measures in association rule mining algorithm. The support of a rule $X \rightarrow Y$ is the ratio (in percent) of the transactions (T) that contain $X \cup Y$ to the total number of transactions in the database ($|D|$):

$$Support(X \rightarrow Y) = \frac{|\{T \in D \,|\, X \cup Y \subseteq T\}|}{|\, D \,|}$$

The confidence is the ratio (in percent) of the number of records that contain $X \cup Y$ to the number of records that contain X.

$$Confidence(X \rightarrow Y) = \frac{|\{T \in D \,|\, X \cup Y \subseteq T\}|}{|\{T \in D \,|\, X \subseteq T\}|}$$

[2] Mining generalized association rules

As we need argument generation rules with a variable format, it is not enough to use a traditional algorithm to mining association rules, since the arguments and the facts observed are expressed in constant terms [18].

In order to deal with this problem, we employ an algorithm of generalised association rules. These algorithms use the existence of a hierarchical taxonomy of the data to generate different association rules at different levels in the taxonomy [28]. A generalised association rule $X \rightarrow Y$ is defined identically to that of regular association rules, except that no item in Y can be an ancestor of any in X. An ancestor of an item is one above in the taxonomy. Consequently, we build a hierarchical taxonomy of conditions and arguments, in which its leaves are the possible conditions and arguments observed by the agent, and the upper levels are the same propositions but more general and expressed in variable terms. Then, the algorithm for mining generalised association rules will especially be able to generate rules at upper levels in the taxonomy of conditions and arguments. Therefore, the rules will be variable.

To obtain generalised association rules, we must generate association rules for all the levels in the taxonomy. A simple approach to do this would be to take each transaction and expand each item to include all items above in the hierarchy [28]. That is, to add all the ancestors of each item in a transaction t_i to t_i. As expected, when rules are generated from items at a higher level in the taxonomy, both the support and the confidence increase. This is desirable since the algorithm for mining association rules seeks rules with values of support and confidence higher than the minimum ones. For further details about implementation and comparison of performance of generalised association rules algorithm, we recommend to see [28, 16].

[3] Mining fuzzy generalized association rules

The concept of mining fuzzy association rules originates with the need to reduce the effect of sharp boundary when we have to deal with quantitative attributes divided into discrete intervals [17]. In the traditional approach, a quantitative attribute is divided in intervals. The discrete interval method [27] divides the attribute domain into discrete intervals. So, each element will contribute weight to its own interval. Thus, we can use the weights to estimate the importance of an interval. However, we may miss some interesting intervals with the exclusion of some potential elements near the sharp boundaries. To

tackle this problem, the discrete intervals are replaced with fuzzy sets. In the fuzzy set theory, an element can belong to a set with a membership value in [0, 1]. This value is assigned by a membership function associated with each fuzzy set. For each attribute x and its domain D_x the mapping of the membership function is $m_{f_x}(x) : D_x \rightarrow [0, 1]$.

Given a transactional database $D = \{t_1, ..., t_n\}$ and a set of attributes or items $I = \{i_1, ..., i_m\}$ present in the transactions stored in D, each item i_j will associate with several fuzzy sets $F_{i_j} = \{fi_j^1, ..., fi_j^l\}$. Then, an algorithm for mining fuzzy association rules will be able to find rules of the following format: $\langle X, A\rangle \rightarrow \langle Y, B\rangle$, where $X = \{x_1, x_2, ..., x_p\}$ and $Y = \{y_1, y_2, ..., y_q\}$ are subsets of I, and $A = \{f_{x_1}, f_{x_2}, ..., f_{x_p}\}$ and $B = \{f_{y_1}, f_{y_2}, ..., f_{y_q}\}$ contain the fuzzy sets associated with the corresponding attributes in X and Y [17].

In our work, we assume a more general definition of fuzzy transactions and fuzzy association rules that was presented by Delgado et al. [8]. A fuzzy transaction is a nonempty fuzzy subset $t \subseteq I$. For every $i_j \subseteq I$, $m_{t_i}(i_j)$ defines the membership degree of i_j in a fuzzy transaction t_j, and $m_{t_i}(I_0)$ is the degree of inclusion of an itemset $I_0 \subseteq I$ in a fuzzy transaction t_j, defined as $m_{t_i}(I_0) = \min_{i_j \in I_0} m_{t_i}(i_j)$. Then, let I be a set of items, D a set fuzzy transactions, $A, C \subseteq I$ with $A, C \neq \emptyset$, and $A \cap C = \emptyset$, a fuzzy association rule $A \rightarrow B$ holds in T iff $m_{t_i}(A) \leq m_{t_i}(C)\ \forall t_i \in D$. This definition preserves the meaning of association rules, due to the fact that if we assumed $A \subseteq t_i$ in some sense, we must assume $C \subseteq t_i$ given that $m_{t_i}(A) \leq m_{t_i}(C)$ [8].

There are several algorithms to mining fuzzy association rules [17, 7]; however, we utilize the algorithm presented by Hong et al. [11], because this algorithm integrates mining fuzzy and generalized association rules.

3.2 Mining argument generation rules: a fuzzy approach

Since agents interact in a multiagent system, they can observe the arguments that other agents generate during a negotiation. Additionally, the agents can also observe the negotiation context in which each argument is generated and store the facts that made up the context in that moment. Following this idea, the observations made by an agent will be stored in a knowledge base $O = \{o^1, o^2, ..., o^n\}$, where o^i represents the i^{th} observation in O . Each observation o^i is a tuple with the format: (H^i, a^i), where $H^i = \{h_1^i, h_2^i \ldots, h_s^i\}$ is the set of facts in the context where the argument a^i was generated and h_j^i represents

the j^{th} fact of the i^{th} observation in O. Moreover, the argument a^i is defined as $a^i = (ae^i, ar^i, conc^i, prem^i)$, where ae^i is the agent that uttered the argument; ar^i is the agent that received it; $conc^i$ is the conclusion of the argument; and $prem^i$ is its list of premises. Given these observations, we want to find the relations between the facts observed and the arguments uttered, since these relationships are the argument generation rules we want to learn. The steps to fulfil this goal are:

1. Definition of fuzzy transactions from the observations.
2. Taxonomy building.
3. Execution of the algorithm for mining fuzzy generalised association rules.
4. Post-processing of rules.

We detail these steps below.

[1] Defining fuzzy transactions from observations

Based on the observations stored in O, we should define the fuzzy transactions for mining association rules. As introduced above, the observations are tuples with the format *(H, a)*. The argument a was generated by applying the rule $C \rightarrow a$, where C is the set of conditions that should be fulfilled by the facts of the context. Therefore, it is correct to think that the set of conditions C is a subset of H. While the agent cannot be certain about the elements that compound the set C, it can define a function $m_a(h_j) : H \rightarrow [0, 1]$ which determines the grade of semantic relation m_{ah_j} between the argument a and each fact $h_j \epsilon H$. Therefore, we can define C as a fuzzy set where m_{ah_j}determines the grade of membership of each fact h_j into the set C. That is, each observed fact is included in a fuzzy transaction to a certain degree [8], given by the semantic of the relation between the observed fact and the argument.

Thus, for each observation $o^i = (\{h^i_1, ..., h^i_s\}, a^i) \epsilon O$, we define a fuzzy transaction $t^i = (h^i_1, ..., h^i_s, a^i)$ with $m^i_a(h^i_j) : H^i \rightarrow [0, 1]$ as the membership function that associates each item to the fuzzy transaction.

In order to define m^i_a, for each observation o^i, we extract a set of facts $AF^i = \{af^i : af^i = conc^i \vee af^i \epsilon prem^i\}$ from the argument $a^i = (ae^i, ar^i, conc^i, prem^i)$. That is, $AF^i = \{af^i_1, af^i_2, ..., af^i_l\}$ is composed of the conclusion and premises of a^i, and af^i_q represents the q^{th} fact extracted from a^i. Then, the membership function m^i_a is defined as follows:

$$m_a^i\left(h_j^i\right) = \begin{cases} \mu_j^i & \mu_j^i \leq 1 \\ 1 & \mu_j^i > 1 \end{cases}$$

where $\mu_j^i = \sum_{q=1}^{l} relatedTo(ae^i,\ ar^i,\ af_q^i,\ h_j^i)$ is the sum of the grades of relation between af_q^i and h_j^i, by taking also into account the sender and receiver of the argument (ae^i and ar^i respectively). Internally, the function $relatedTo(ae,\ ar,\ f,\ h)$ can be defined in different ways. In this work, we have chosen a trivial approach. We define a set of rules that determine the semantic relation between the information that can be extracted from the arguments (premises and conclusions) and each fact of the context (taking into account the definition of the predicates in L). This relation is determined by taking into account the semantics of the facts. For example, given an argument *reward(a1, a2, do(a2, accept(discusstopic(topic1))), [do(a1, accept(discusstopic(topic2)))])*, we can extract the conclusion *do(a2, accept(discusstopic(topic1)))* and the premise *do(a1, accept(discusstopic(topic2)))*. By observing these propositions we can strongly suppose that the fact *believe(a2, do(a2, accept(discusstopic(topic1))), discusstopic(topic1))* can be related to the conclusion and that the fact *believe(a1, do(a1, accept(discusstopic(topic2))), discusstopic(topic2))* can be related to the premise. Consequently, given the fact *believe(a1, do(a1, accept(discusstopic(topic2))), discusstopic(topic2))*, we can also suppose that a fact *hasgoal(a2, goal(discusstopic(topic2))* can be part of the conditions of the argument generation rule. Likewise, from *hasgoal(a2, goal(discusstopic(topic2))*, we can assume that a fact *prefer(a2, goal(discusstopic(topic2)), goal(discusstopic(topic4)))* can also be related to the rule to generate the reward. In contrast, in other situations, we can be certain that a fact is part of the conditions. For instance, given an appeal to past promises, one of its premises (see Section 2.2) is a fact *pastpromise(A_i, A_j, P)*, which must be unequivocally present in the negotiation context.

In Appendix A, we show how the function $relatedTo(ae,\ ar,\ f,\ h)$ can be defined.

[2] Taxonomy building

As mentioned above, we need to build a taxonomy with the items of the transactions. To build the taxonomy, we start putting the conditions and arguments of the observations in the leaves just as they were

stored by the agent (i.e. *reward(a1, a2, do(a2, accept(discusstopic(topic1))), [do(a1, accept(discusstopic(topic2)))])* and *believe(a2, do(a2, accept(discusstopic(topic1))), discusstopic(topic1))*), one condition or argument for each leaf. That is, for each item (condition or argument) of each transaction, we build a branch in the taxonomy by starting at this item (leaf) and ending at the root of the taxonomy.

To build this branch, we take an item and generate all the ancestors that represent the same condition or argument but replacing each terminal term (proposition of L that does not have another generalisable proposition as parameter) by the respective most general one. To determine this, we maintain a data structure (hash table) HT with propositions and their most general form. For example, for the proposition *hasgoal* the most general form stored in HT will be *hasgoal(Agent, goal(Goal))*; for agent *a1*, *Agent*; for *goal*, *goal(Goal)*; for *discusstopic*, *discusstopic(Topic)*, among others.

Thus, given the condition *hasgoal(a1, goal(discusstopic(topic1)))*, we add a leaf with it and create the following ancestors, taking into account that their terminal terms are *a1* and *discusstopic(topic1)*:

- *anc1*: *hasgoal(Agent, goal(discusstopic(topic1)))* by replacing the proposition *a1* with *Agent*, where *Agent* is the most general form of *a1*.
- *anc2*: *hasgoal(a1, goal(discusstopic(Topic)))* by replacing the proposition *discusstopic(topic1)* with *discusstopic(Topic)*, where *discusstopic(Topic)* is the most general form of *discusstopic(topic1)*.

Next, we successively perform the same action, with each ancestor, and create a new node in the taxonomy that represents the item, whose parents are the previously generated ancestors. Following the example, the new ancestor of *anc1* is *hasgoal(Agent, goal(discusstopic(Topic)))* (the same for *anc2*); and finally, we replace *goal(discusstopic(Topic))* with *goal(Goal)* and obtain the most general expression of the initial condition. When the most general expression is found, a new node is created in the taxonomy whose parent is the root. Figure 1 shows an example of this part of the taxonomy.

Figure 1 Part of the taxonomy of conditions and arguments.

[3] Execution of the algorithm for mining fuzzy generalised association rules

Having just built the taxonomy, for each transaction $t_i \epsilon D$, we replace each item $i_{ij}\, \epsilon\, t_i$ with its nearest ancestor that has no constants. Thus, we eliminate all possible constant rules, since we are interested in finding rules that can be completely instantiated in different negotiation contexts. For example, the item *hasgoal(a1, goal(discusstopic(topic3)))* will be replaced with the ancestor *hasgoal(Agent, goal(discusstopic(Topic)))*. In addition, the variable terms are numerated to keep the reference among the different items of the transaction. Therefore, the final version of the previous item will be: *hasgoal(Agent0, goal(discusstopic(Topic0)))*. Consequently, every time we find *a1* or *topic3* in an item of the same transaction, it will be replaced with *Agent0* and *Topic0* respectively.

After this pre-processing, we run the algorithm for mining fuzzy generalized association rules described by [11]. Notice that other algorithms for mining fuzzy generalized association rules can be used, since our approach is independent of the algorithm, provided the algorithm observes the definitions presented above. Regarding to the definition of the minimum support and minimum confidence values, useful information can be found in [11]. There, Hong et al. showed the relationship between number of rules mined, minimum support value, minimum confidence value and execution time.

[4] Post-processing fuzzy generalised association rules

The post-processing of fuzzy generalised association rules can be divided in three parts. First, we filter out the rules whose format is not adjusted to $C \rightarrow a$. That is, once all the rules have been obtained, we just keep those whose antecedent is only composed of conditions and whose consequent is a single argument. The remainder are filtered out. Since the algorithms for mining association rules process all the items of a transaction alike, there is no semantic difference between conditions and arguments. In this way, it is possible to find rules like $condition_x \rightarrow condition_y$ or $argument_w \rightarrow condition_z$, which fulfil the minimum levels of support and confidence, but are irrelevant to build arguments. For instance, an irrelevant rule could be *hasgoal(Agent1, goal(Goal1)), hasgoal(Agent1, goal(Goal2)) → prefer(Agent1, goal(Goal1), goal(Goal2))*. This rule is inappropriate because its three items are conditions. It is worth noticing that some aspects of this post-processing stage can be integrated into the algorithm for mining fuzzy generalised association rules. For instance, during the frequent itemset search, we can eliminate those itemsets that do not include arguments. This integration does not change the rules learned, but improves the performance of the algorithm.

Second, we determine how representative the rules are with respect to the argument generated by the agents and gathered in observations O. To perform this task, we define a sufficiency metric of an association rule. This metric represents the relation between the conditions of the transactions (observations) that support the rule and the conditions of this rule. It is calculated as the ratio between the number of conditions of a rule over the average of conditions of the transactions that support it. It is defined as:

$$Sufficiency(r) = \frac{totalConditions(r)}{averageConditionsOfTransactionsSupporting(r)}$$

For example, if we have the transactions $t_1 = (c_1, c_3, c_5, a_1)$, $t_2 = (c_1, c_2, c_4, a_1)$, $t_3 = (c_1, c_4, a_1)$, and $t_4 = (c_1, c_5, a_2)$; and we define a minimum support of 0.5 and a minimum confidence of 0.75, we will obtain, after the first post-processing step, the rules $r_1 : c_1 \rightarrow a_1$, $r_2 : c_4 \rightarrow a_1$ and $r_3 : c_1, c_4 \rightarrow a_1$. The support and confidence of the three rules exceed the minimum support and confidence values. However, we can see that rules r_1 and r_2 are not sufficiently representative with regard to transactions t_1, t_2 and t_3, because it is improbable that a single condition be sufficient to generate the argument a_1, since the conditions are not isolated in the transactions. The sufficiency metric aims to filter these rules by setting a threshold that determines how sufficient the con-

ditions (antecedent) of a rule must be to generate the consequent argument, independently of the values of support and confidence.

Rules	Support	Confidence	Sufficiency
$r_1 : c_1 \rightarrow a_1$	0.75	0.75	0.375
$r_2 : c_4 \rightarrow a_1$	0.5	1	0.4
$r_3 : c_1, c_4 \rightarrow a_1$	0.5	1	0.8

Table 1 Metrical comparison of rules.

Table 1 details the values of the three metrics. We can observe that rules r_1 and r_2 have a sufficiency value comparatively low with regard to rule r_3. Therefore, a threshold of 75% only allows rule r_3 to be valid. At the moment of selecting an argument, the agent can also use the value of this metric, favouring the selection of arguments generated by rules with higher value of sufficiency.

Finally, since the algorithm for mining generalised association rules can find rules at different level in the taxonomy, it is possible to find rules that are ancestors of other rules. In these cases, we keep the most general rules since it will be possible to apply these rules in a wide spectrum of negotiation contexts. For example, given the rules $r_4 : d_1, d_2 \rightarrow a_2$ and $r_5 : d_1, d_2' \rightarrow a_2$, we keep r_5 assuming that d_2' is an ancestor of d_2.

§4 Experimental results

The domain we chose to test our proposal was a multi-agent application for meeting scheduling. In this application, the agents must arrange meetings by discussing date, time, place, topics for discussion, and participants by taking into account the preferences and goals of the users that they represent. Since users have different goals, the agents must exchange arguments in order to reach agreements.

The aim of the experiments was to determine the argument generation rules that the agents use during the negotiation by using the proposed approach. To determine the efficiency of our approach, we carried out three experiments. First, we compared the rules learned by using our approach with the original rules used by the observed agents. Second, we compared the rules learned by using our fuzzy approach with the rules learned by using a *crisp* one, in order to assess the contribution of fuzziness to this problem. Finally, we compared the set of arguments that can be generated by using the original rules with the set of arguments that can be generated by using the rules learned by the fuzzy

approach as well as by the crisp one. In addition, we carried out a scalability analysis to assess the performance of the approach in different scenarios.

4.1 Experiment scenario

The experiments were carried out with four agents, where each agent represents a user. For each agent, we randomly generated a set of goals, preferences among goals, beliefs and historic facts (past promises, information about other meetings, goals achieved). Taking into account goals and beliefs, agents had to generate arguments to persuade other agents by applying the set of argument generation rules defined in Section 2.2 (we call *original rules* to this set of rules). After each negotiation, the arguments and context facts (goals, preferences, beliefs and historic facts) were stored in O for processing. The negotiation context was composed of 439 facts: 47 goals; 21 preference among goals; 286 beliefs; 37 past goals; 36 past actions; and 12 past promises.

The minimum support and minimum confidence values were defined in 2.0 (notice that in the algorithm proposed by [11] the support is not between 0 and 1 as in the traditional algorithms) and 0.9 respectively. We selected a low minimum support value to ensure that all the interesting rules were generated. In contrast, we selected a high minimum confidence value to ensure a strong relation between antecedent and consequent. Moreover, we defined a minimum sufficiency value of 0.7. These values were assigned after analysing the information and experiments provided in [11].

4.2 Results obtained

In total, 69 arguments and their contexts were registered in O. Table 2 shows the number of arguments observed for each argument type (column *Arguments*). For each observation, we built a fuzzy transaction. Then, from the set of fuzzy transactions, we built the taxonomy following the steps described in Section 3.1.2. Afterwards, we mined fuzzy generalized association rules using the algorithm defined by [11]. Table 2 shows the total number of rules obtained for each argument type[*2] (column *Rules*); the number of rules that follow the format $C \rightarrow a$ (column *Format*); the number of rules with the minimum value of sufficiency (column *Sufficiency*); and the number of final rules learned by the approach after selecting the most general ones (column *Generality*). As we

[*2] Notice that each type corresponds to an argument generation rule, as defined in Section 2.2.

can see, we obtained, at first, 7010 fuzzy association rules from 69 observations. Finally, 14 rules were kept after post-processing stage. It is worth pointing out that each observation was composed of 440 facts (439 context facts and an argument).

Argument type	Arguments	Rules	Format	Sufficiency	Generality
Reward	16	1351	483	176	2
Threat	4	95	33	1	1
Prevailing practice	8	47	18	1	1
Self interest	20	32	13	8	3
Past promise	12	5	1	1	1
Counterexample	9	5480	1380	195	6
Total	**69**	**7010**	**1928**	**382**	**14**

Table 2 Arguments observed and rules obtained by using the proposed approach.

As mentioned above, we analysed the results from different perspectives in order to assess the precision and contribution of the proposed approach.

[1] Analysis of the learned rules

To analyse the results, we first compared the learned rules with the original ones. This comparison was carried out by collating the number and format of the conditions and the format of the consequent argument. After this comparison, we classified each learned rule into four categories: *Correct*, when the learned rule was exactly the same rule applied by the agents; *Partial*, when the learned rule had less conditions that the original one; *Larger*, when the rule had additional conditions to the conditions of the original one; and *Wrong*, when the learned rule was unrelated to the original one.

Argument type	Correct	Partial	Larger	Wrong	Total
Reward	1	-	-	1	2
Threat	1	-	-	-	1
Prevailing practice	1	-	-	-	1
Self interest	2	-	1	-	3
Past promise	1	-	-	-	1
Counterexample	2	3	1	-	6
Total	**8**	**3**	**2**	**1**	**14**
Percentage	**57.2%**	**21.4%**	**14.3%**	**7.1%**	

Table 3 Results of the comparison between the original rules and the rules learned using the proposed approach.

Table 3 shows the results of the comparison between the learned rules and the original ones. As shown, 8 rules were correct (57.2%), 3 rules represented partial rules (21.4%), 2 rules were larger than the original ones (14.3%), and 1 rule was wrong (7.1%). Moreover, notice that all the 8 original rules were discovered by the proposed approach.

To illustrate these results, some rules are presented and analysed below:

- Rule #1:
  - Conditions: *hasgoal(AGENT1, goal(GOAL1)), hasgoal(AGENT1, goal(GOAL0)), prefer(AGENT1, goal(GOAL1), goal(GOAL0)), believe(AGENT1, imply(do(AGENT1, ACTION0), not(GOAL0))), believe(AGENT1, imply(do(AGENT0, ACTION1), not(GOAL1)))*
  - Argument: *threat(AGENT0, AGENT1, do(AGENT1, ACTION0), [do(AGENT0, ACTION1)])*
- Rule #2:
  - Conditions: *hasgoal(AGENT0, goal(GOAL0)), believe(AGENT1, imply(do(AGENT1, ACTION0), GOAL0)), believe(AGENT1, imply(do(AGENT0, ACTION0), GOAL0)), believe(AGENT0, imply(do(AGENT1, ACTION0), GOAL0))*
  - Argument: *reward(AGENT0, AGENT1, do(AGENT1, ACTION0), [do(AGENT0, ACTION1)])*
- Rule #3:
  - Conditions: *wasgoal(AGENT1, goal(GOAL0)), hasgoal(AGENT1, goal(GOAL0)), hasgoal(AGENT0, goal(not(GOAL0))), did(AGENT1, ACTION0), believe(AGENT1, imply(do(AGENT1, ACTION1),*

not(GOAL0)))

- Argument: *appeal(AGENT0, AGENT1, do(AGENT1, ACTION1), [did(AGENT1, ACTION0), imply(ACTION0, not(GOAL0))])*

- Rule #4:
  - Conditions: *hasgoal(AGENT0, goal(GOAL0)), believe(AGENT1, imply(do(AGENT0, ACTION0), GOAL0)), believe(AGENT0, imply(do(AGENT0, ACTION0), GOAL0))*
  - Argument: *appeal(AGENT1, AGENT0, do(AGENT0, ACTION0), [imply(do(AGENT0, ACTION0), GOAL0), hasgoal(AGENT0, GOAL0)])*

Rule #1 fits exactly the threat generation rule defined in Section 2.2, thus, it is correct. Rule #2 is a wrong rule to generate rewards, because takes into account only one goal in its conditions. Rule #3 is a rule for counterexample generation. This rule has just a belief included in its conditions, but the original one has two, thus, it is a partial rule. Finally, rule #4 has the conditions of the original rule and an additional condition, which is not semantically wrong, but restrictive. It is interesting to note that partial and wrong rules could generate wrong arguments, but larger ones would not, especially if the correct rule was also found.

[2] Analysis of the arguments generated by the learned rules

In order to assess the contribution of fuzziness to the problem of learning argument generation rules, we also applied a *crisp* approach to learn these rules. This experiment was carried out over the same knowledge base O used in the first experiment. In contrast to the fuzzy approach, the crisp transactions were traditional sets where we included all the facts related to the observed argument. Formally, we defined a crisp transaction $ct^i = (h_1^i, ..., h_s^i, a^i)$, where $m_a^i(h_j^i) > 0$. That is, a fact was included in a crisp transaction if the result of the membership function (*relateTo* function) was higher than 0. We used the membership function in both transaction definitions (fuzzy and crisp) to maintain equivalence in the results.

Table 4 shows the total number of rules obtained for each argument type applying the crisp approach. After filtering, the crisp approach learned 149 rules against 14 rules learned by the fuzzy approach. Then, we compared the crisp rules and the original rules. Table 5 shows the result of this comparison.

Argument type	Arguments	Rules	Format	Sufficiency	Generality
Reward	16	15013	3358	2470	97
Threat	4	510	126	29	29
Prevailing practice	8	94	30	6	4
Self interest	20	40	15	10	6
Past promise	12	25	13	13	2
Counterexample	9	5480	1380	195	11
Total	**69**	**21162**	**4922**	**2723**	**149**

Table 4 Arguments observed and rules obtained by using a crisp approach.

Argument type	Correct	Partial	Larger	Wrong	Total
Reward	1	23	22	51	97
Threat	1	14	6	8	29
Prevailing practice	1	1	1	1	4
Self interest	2	-	4	-	6
Past promise	1	-	1	-	2
Counterexample	1	4	1	5	11
Total	7	42	35	65	149
Percentage	4.7%	28.2%	23.5%	43.6%	

Table 5 Results of the comparison between the original rules and the rules learned using the crisp approach.

Comparing the results showed in Table 3 and Table 5, we can see that the total number of correct rules was similar in both approaches. However, the rate of correct rules in the crisp approach (4.7%) was considerably smaller than the rate of correct rules in the fuzzy approach (57.2%). This is because the crisp approach learns a great number of partial (28.2%), larger (23.5%) and wrong rules (43.6%).

To determine and compare the precision of the fuzzy and the crisp approaches, we compared the set of arguments that can be generated by the original rules and the arguments that can be generated by the rules learned using the fuzzy and the crisp approaches. To carry out this comparison, we simulated 1000 negotiation contexts. Each negotiation context was composed of goals, preferences among goals, beliefs and historic facts generated at random by taking into account the language L defined in Section 2.1. The average number of facts for each negotiation context was 182.72 (20.97 goals, 1.99 preferences, 99.48 beliefs,

20.41 past goals, 21.27 past actions, and 11.57 past promises).

In each negotiation context, arguments were generated by using the three sets of rules. Table 6 shows the results of this comparison. This table shows the number of arguments generated by: the set of original rules (row *Arguments - Original*), the set of rules learned by the fuzzy approach (row *Arguments - Fuzzy*), and the set of rules learned by the crisp approach (row *Arguments - Crisp*). We call *learned arguments* to the arguments generated by the learned rules. We compared both sets of learned arguments from two points of comparison. First, we analysed if the learned arguments were correct by checking if they were also included into the set of original arguments (row *Correct*). Thus, if a learned argument was not included into the set of original arguments, we assumed that it was wrong (row *Wrong*). Finally, we analysed if the original arguments were also generated by the set of learned rules. Thus, we distinguished between the original arguments that were generated and the original arguments that were not generated into the set of learned arguments (row *Generated* and *Not generated*). Table 6 shows the total number of arguments (row #) and the percentage of arguments in relation to the originals ones (row %) for each approach (*Fuzzy* or *Crisp*) and each comparison (*Correct-Wrong* and *Generated-Not generated*).

As shown in Table 6, 100% of the arguments generated by the rules learned by the fuzzy approach were correct, and the 100% of the original arguments were also generated by these rules. In contrast, the crisp approach showed a low precision. Although the 96.5% of the arguments generated with the original rules were also generated using the rules learned by the crisp approach, the percentage of wrong arguments was high (55.06%). This is because the number of partial, large and wrong rules learned by the crisp approach exceeds the number of correct ones.

These results indicate that the argument generation rules learned using the proposed approach have a high precision, despite the occurrence of partial and wrong rules. Moreover, the difference of precision between the fuzzy and the crisp approaches demonstrate the necessity of dealing with the uncertainty by using a fuzzy approach.

In summary, we claim that the results are promising since a high precision was obtained from a reduce number of arguments observed in a large negotiation context. Moreover, we think that partial and larger rules, which can produce a precision decrease, can be dismissed by the agent by checking the

			Arguments						Total
			R	T	PP	SI	PsP	C	
Arguments	Original		77260	41307	52518	20269	11576	49816	252746
	Fuzzy		77260	41307	52518	20269	11576	49816	252746
	Crisp		247175	108153	52518	82322	11576	40980	542724
Correct	Fuzzy	#	77260	41307	52518	20269	11576	49816	252746
		%	100%	100%	100%	100%	100%	100%	100%
	Crisp	#	77260	41307	52518	20269	11576	40980	243910
		%	31.26%	38.19%	100%	24.62%	100%	100%	44.94%
Wrong	Fuzzy	#	0	0	0	0	0	0	0
		%	0%	0%	0%	0%	0%	0%	0%
	Crisp	#	169915	66846	0	62053	0	0	298814
		%	68.74%	61.81%	0%	75.38%	0%	0%	55.06%
Generated	Fuzzy	#	77260	41307	52518	20269	11576	49816	252746
		%	100%	100%	100%	100%	100%	100%	100%
	Crisp	#	77260	41307	52518	20269	11576	40980	243910
		%	100%	100%	100%	100%	100%	82.26%	96.5%
Not generated	Fuzzy	#	0	0	0	0	0	0	0
		%	0%	0%	0%	0%	0%	0%	0%
	Crisp	#	0	0	0	0	0	8836	8836
		%	0%	0%	0%	0%	0%	17.74%	3.5%
R: Reward. T: Threat. PP: Prevaling Practice. SI: Self Interest. PsP: Past Promise. C: Counterexample. #: Number of arguments. %: Percentage of arguments.									

Table 6 Results of comparing arguments generated by using both the original and learned rules.

success of the arguments uttered.

[3] Scalability analysis

To finalize the experiments, we carried out a scalability analysis to evaluate the performance of the algorithm for mining fuzzy generalized association rules in the scope of this work. These experiments were implemented in Java on a personal computer with an Intel(R) Core(TM) i5 processor and 4 Gb of RAM. Experiments were made to show the relationship between number of arguments observed (number of transactions), number of facts in the context and the execution-time. We randomly generated negotiation context with different numbers of facts (from 100 to 1000 facts). From each negotiation context, different numbers of arguments were generated and stored in O (from 10 to 100 arguments). Then, the proposed approach was run to learn argument generation rules. This process was carried out 10 times for each combination of number of facts in the negotiation context and number of arguments observed. Finally, the average execution-time was computed.

Figure 2 shows the results. These results show that the execution-time tends to increase along with increase of number of facts in the context and number of argument observed. However, the increasing of the execution-time seems more sensible to the increasing of the number of argument observed than to the increasing of the number of facts in the context. This is reasonable using the fuzzy approach since the number of facts in the context does not influence directly the number of items in the transactions. On the other hand, it is well-known that the execution-time increases when the number of transactions also increases Agrawal and Srikant [1]. Moreover, this fact does not affect our approach since a good precision is obtained from a reduced number of observed arguments, as we showed in Section 4.2.1.

§5 Related work

We can distinguish some works related to argumentation and mining association rules. Governatori and Stranieri [10] applied an algorithm for mining association rules to facilitate the discovery of defeasible rules that represent the *ratio decidendi* underpinning legal decision making. Afterwards, these defeasible rules are used to build formal arguments, not rhetoric ones. Moreover, the rules obtained are only useful in the context in which they were discovered. Similarly, in [30] and [29] arguments are pooled from the agent's experience by means

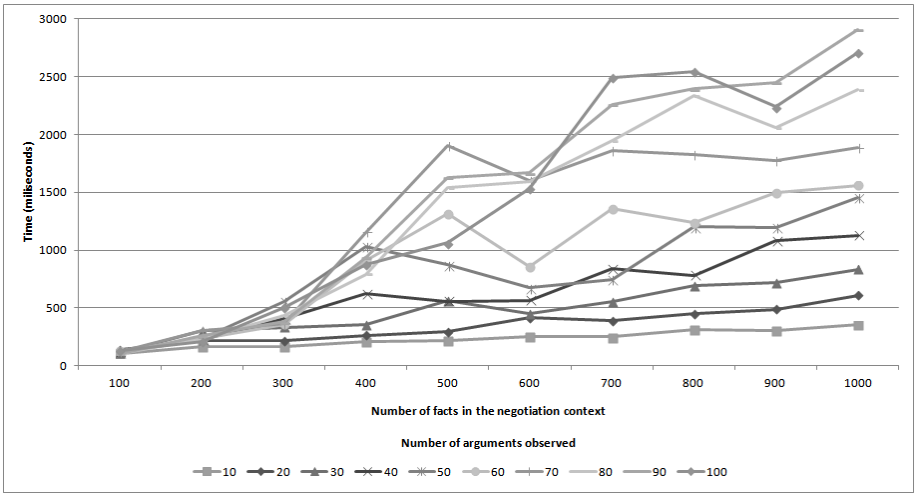

Figure 2 The relationships between number of arguments observed, number of facts in the context and the execution-time.

of association rule mining techniques. In that work, the authors present an argumentation protocol, PADUA (Protocol for Argumentation Dialogue Using Association Rules), designed to enable participants to debate on the basis of their experience. In PADUA, an association rule merely means that the antecedent is a set of reasons for believing the consequent, and it aids the agent, when arguing, in deciding what kind of dialogue move to play during an argumentative dialogue.

In a previous work, an approach to build user argumentative models was presented [18]. A user argumentative model captures the argumentative styles of the user and is depicted by the argument generation rules that the user utilises during the argumentation. In contrast to the current proposal, that approach applies an algorithm for mining generalized association rules to discover these rules and assumes that the observed conditions that the user observes to generate their arguments can be directly accessed. This assumption eliminates the uncertainty about what facts should be part of the conditions of such rules. Consequently, a fuzzy approach is not necessary.

§6 Conclusions and future work

In this work, we have presented an approach which allows agents to learn argument generation rules by observing how other agents argue in a negotiation context. Our approach applies an algorithm for mining fuzzy generalized association rules to accomplish this goal. Appling this algorithm, we can discover

rules with the format $C \rightarrow a$ from a set of observations, where C represents the set of conditions needed to generate an rhetorical argument a. Moreover, we can obtain rules with different grades of generality, which can be used in different negotiation contexts. Furthermore, the use of fuzzy sets allows us to handle the uncertainty involved in the knowledge about what information on the negotiation context is taken into account by the observed agents to generate their arguments.

The experimental results show that it is possible to learn the argument generation rules used by other agents from a small number of arguments observed. As shown above, the rules obtained were correct in a 57.2%, partial in a 21.4%, restricted in a 14.3% and wrong in a 7.1%. Moreover, it is worth noticing that all the rules applied to build the training arguments were learned correctly by our approach. Finally, our approach obtained a high precision by comparing the set of arguments that could be generated by the original rules with the set of arguments that could be generated by the rules learned by our approach. Furthermore, we confirmed the contribution of fuzziness to the problem of learning argument generation rules, by comparing the precision of a fuzzy and a crisp approach. The fuzzy approach obtained a precision of 100% whereas the precision of the crisp approach was 44.94%.

The main future research direction is to analyse other definitions of the membership function that relates the facts of the negotiation context with a given argument. In particular, we will explore the use of ontologies to determine the semantic context in which the argument was generated. Another future work aims to incorporate argument selection learning to this approach, that is, to allow the agent to learn from other agents how to select an argument from the set of candidate arguments generated previously. In this way, we will integrate this approach with a reinforcement learning approach to improve the argument selection effectiveness [19].

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93*, pages 207–216, New York, NY, USA, 1993. ACM. ISBN 0-89791-592-5. doi:

http://doi.acm.org/10.1145/170035.170072.

[3] L. Amgoud and H. Prade. Generation and evaluation of different types of arguments in negotiation. In *Proc. of the international workshop on non-monotonic reasoning*, pages 10–15, Whistler BC, Canada, 2004.

[4] L. Amgoud, S. Parsons, and N. Maudet. Arguments, dialogue, and negotiation. *Journal of Artificial Intelligence Research*, 23:2005, 2000.

[5] L. Amgoud, Y. Dimopoulos, and P. Moraitis. A unified and general framework for argumentation-based negotiation. In *AAMAS '07*, pages 1–8, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5. doi: http://doi.acm.org/10.1145/1329125.1329317.

[6] R. Ashri, I. Rahwan, and M. Luck. Architectures for negotiating agents. In V. Marik, J. Mueller, and M. Pechoucek, editors, *Proc. of Multi-Agent Systems and Applications III*, pages 136–146, Prague, Czech Republic, 2003. Springer. URL `http://eprints.ecs.soton.ac.uk/8316/`.

[7] M. Delgado, N. Marin, M.J. Martin-Bautista, D. Sanchez, and M.A. Vila. Mining fuzzy association rules: an overview. In *BISC International Workshop on Soft Computing for Internet and Bioinformatics*, 2003.

[8] M. Delgado, N. Marín, D. Sánchez, and M.A. Vila. Fuzzy association rules: general model and applications. *IEEE Transactions on Fuzzy Systems*, 11:214–225, 2003.

[9] M. M. Geipel and G. Weiss. A generic framework for argumentation-based negotiation. In *CIA '07: Proceedings of the 11th international workshop on Cooperative Information Agents XI*, pages 209–223, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-75118-2.

[10] G. Governatori and A. Stranieri. Towards the application of association rules for defeasible rule discovery. In R. Loui & A. Muntjewerff B. Verheij, A. Lodder, editor, *Frontiers in Artificial Intelligence and Applications, Proceedings of JURIX 2001*, volume 70. IOS Press, 2001.

[11] T. Hong, K. Lin, and S. Wang. Fuzzy data mining for interesting generalized association rules. *Fuzzy Sets Syst.*, 138(2):255–269, 2003. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/S0165-0114(02)00272-5.

[12] N. R. Jennings, S. Parsons, P. Noriega, and C. Sierra. On argumentation-based negotiation. In *Proceedings of the International Workshop on Multi-Agent Systems (IWMAS-98)*, Boston, MA, 1998.

[13] A. Kandel. *Fuzzy expert systems*. CRC Press, 1992.

[14] M. Karlins and H. I. Abelson. *Persuasion: how opinions and attitudes are changed*. Springer Verlag: Berlin, 1970.

[15] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artif. Intell.*, 104(1-2):1–69, 1998. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(98)00078-2.

[16] D. Kunkle, D. Zhang, and G. Cooperman. Mining frequent generalized itemsets and generalized association rules without redundancy. *Journal of Computer Science and Technology*, 23:77–102, 2008. ISSN 1000-9000. URL `http://dx.doi.org/10.1007/s11390-008-9107-1`. 10.1007/s11390-008-9107-1.

[17] C. M. Kuok, A. W. Fu, and Wong M. H. Mining fuzzy association rules in databases. *SIGMOD Record*, 27(1):41–46, 1998.

[18] A. Monteserin and A. Amandi. Building user argumentative models. *Applied Intelligence*, 32(1):131–145, 2010. ISSN 0924-669X. doi: http://dx.doi.org/10.1007/s10489-008-0139-6.

[19] A. Monteserin and A. Amandi. A reinforcement learning approach to improve the argument selection effectiveness in argumentation-based negotiation. *Expert Systems with Applications*, 40(6):2182 – 2188, 2013. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.10.045. URL `http://www.sciencedirect.com/science/article/pii/S0957417412011694`.

[20] D. J. O'Keefe. *Persuasion: Theory and Research*. SAGE Publications, 1990.

[21] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998. URL `http://eprints.ecs.soton.ac.uk/2113/`.

[22] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. Mcburney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *Knowledge Engineering Review*, 18(4):343–375, 2003. ISSN 0269-8889. doi: http://dx.doi.org/DOI:10.1017/S0269888904000098.

[23] I. Rahwan, L. Sonenberg, and P. Mcburney. Bargaining and argument-based negotiation: Some preliminary comparisons. In *LNCS 3366:176–191*, 2005.

[24] S. D. Ramchurn, N. R. Jennings, and C. Sierra. Persuasive negotiation for autonomous agents: A rhetorical approach. In *IJCAI Workshop on Computational Models of Natural Argument*, pages 9–17, 2003. URL `http://eprints.ecs.soton.ac.uk/8541/`.

[25] A. S. Rao and M. P. Georgeff. Bdi-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.9247`.

[26] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiation. In *ATAL '97: Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages*, pages 177–192, London, UK, 1998. Springer-Verlag. ISBN 3-540-64162-9.

[27] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 1–12, 1996.

[28] R. Srikant and R. Agrawal. Mining generalized association rules. *Future Generation Computer Systems*, 13(2–3):161–180, 1997. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.7602`.

[29] M. Wardeh, T. Bench-Capon, and F. Coenen. Multi-party argument from experience. In *ArgMAS*, pages 216–235, 2009.

[30] M. Wardeh, T. Bench-Capon, and F.P. Coenen. Padua: a protocol for argumentation dialogue using association rules. *AI and Law*, (3):Springer, 2009.

[31] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

§1 Appendix A: Definition of function *relatedTo*

As we stated in Section 3.1.1, the function $relatedTo(ae, ar, f, h)$ is defined by a set of rules that determine the relation between the information that can be extracted from the arguments (premises and conclusions) and each fact of the negotiation context. We define these rules by analysing the semantic relations among the predicates defined by the negotiation language L. These semantic relations arise directly from the given definitions of these predicates. From L, we recognize four grades of relation: complete (1), high (0.8), intermediate(0.6) and low (0.5).

The rules were defined using *Prolog*, where the parameter *Value* is the returned value and represents the grades of semantic relation between f and h. We list the rules (and their explanations) used during the experiments below.

- If the fact extracted from the argument (f) is the same to the fact observed in the negotiation context, we can be certain that the fact is part of the conditions. Therefore, the grade of relation is complete and *Value* is 1.
  - *relatedTo(_, _, Fact, Fact, Value):- Fact, Value = 1.*
- A special case of the previous rule is the relation between the predicate *imply* and a belief. If a predicate *imply* is included in a belief, the grade of relation between both facts is also complete.
  - *relatedTo(A, _, imply(ActionB, Goal), believe(A, imply(ActionB, Goal)), Value) :- believe(A, imply(ActionB, Goal)), Value = 1.*
- As defined in L, the conclusion of an argument (appeals, rewards and threats) and the premise of rewards and threats are predicates of the format: *do(A, Action)*. Thus, since an agent action is especially related to belief and goals [25], we can define some rules by relating these facts. Linking agent actions and beliefs, we distinguish two points of relation: (a) the agent action is included into the belief through the predicate *imply* and (b) the agent that performs the action is the same agent that has the belief. Then, if (a) and (b) are true, the grade of relation is high.
  - *relatedTo(_, A, do(A, Action), believe(A, imply(do(A, Action), Some)), Value) :- believe(A, imply(do(A, Action), Some)), Value = 0.8.*
- However, if the (a) is true, (b) is not, but these agents are the sender or the receiver agent, we assign an intermediate grade of relation.
  - *relatedTo(Ae, Ar, do(Ae, Action), believe(Ar, imply(do(Ae, Action),*

Some)), Value) :- believe(Ar, imply(do(Ae, Action), Some)), Value = 0.6.

- *relatedTo(Ae, Ar, do(Ar, Action), believe(Ar, imply(do(Ae, Action), Some)), Value) :- believe(Ar, imply(do(Ae, Action), Some)), Value = 0.6.*
- *relatedTo(Ae, Ar, do(Ae, Action), believe(Ae, imply(do(Ar, Action), Some)), Value) :- believe(Ae, imply(do(Ar, Action), Some)), Value = 0.6.*
- *relatedTo(Ae, Ar, do(Ar, Action), believe(Ae, imply(do(Ar, Action), Some)), Value) :- believe(Ae, imply(do(Ar, Action), Some)), Value = 0.6.*

- To link agent actions and goals, we need to know whether a goal is achieved or frustrated when the action is performed. In other words, we need to find in the negotiation context an *intermediate* belief that links the action and the goal. In this case, the relation between an action and a goal is high when the goal is achieved after performing the action.
  - *relatedTo(_, _, do(A, Action), hasgoal(A, goal(Goal)), Value) :- believe(A, imply(do(A, Action), Goal)), hasgoal(A, goal(Goal)), Value = 0.8.*
  - *relatedTo(Ae, Ar, do(Ae, Action), hasgoal(Ar, goal(Goal)), Value) :- believe(Ae, imply(do(Ae, Action), Goal)), hasgoal(Ar, goal(Goal)), Value = 0.8.*
  - *relatedTo(Ae, Ar, do(Ar, Action), hasgoal(Ae, goal(Goal)), Value) :- believe(Ae, imply(do(Ar, Action), Goal)), hasgoal(Ae, goal(Goal)), Value = 0.8.*
  - *relatedTo(Ae, Ar, do(Ae, Action), hasgoal(Ar, goal(Goal)), Value) :- believe(Ar, imply(do(Ae, Action), Goal)), hasgoal(Ar, goal(Goal)), Value = 0.8.*
  - *relatedTo(Ae, Ar, do(Ar, Action), hasgoal(Ae, goal(Goal)), Value) :- believe(Ar, imply(do(Ar, Action), Goal)), hasgoal(Ae, goal(Goal)), Value = 0.8.*
- On the other hand, the relation between actions and goals is also high when a goal is frustrated after performing an action.
  - *relatedTo(_, _, do(A, Action), hasgoal(A, goal(Goal2)), Value) :- believe(A, imply(do(A, Action), Goal1)), hasgoal(A, goal(Goal2)), oppos-*

iteTo(Goal1, Goal2)[*3], *Value = 0.8.*

- *relatedTo(Ae, Ar, do(Ae, Action), hasgoal(Ar, goal(Goal2)), Value) :- believe(Ae, imply(do(Ae, Action), Goal1)), hasgoal(Ar, goal(Goal2)), oppositeTo(Goal1, Goal2), Value = 0.8.*
- *relatedTo(Ae, Ar, do(Ar, Action), hasgoal(Ae, goal(Goal2)), Value) :- believe(Ae, imply(do(Ar, Action), Goal1)), hasgoal(Ae, goal(Goal2)), oppositeTo(Goal1, Goal2), Value = 0.8.*
- *relatedTo(Ae, Ar, do(Ae, Action), hasgoal(Ar, goal(Goal2)), Value) :- believe(Ar, imply(do(Ae, Action), Goal1)), hasgoal(Ar, goal(Goal2)), oppositeTo(Goal1, Goal2), Value = 0.8.*
- *relatedTo(Ae, Ar, do(Ar, Action), hasgoal(Ae, goal(Goal2)), Value) :- believe(Ar, imply(do(Ar, Action), Goal1)), hasgoal(Ae, goal(Goal2)), oppositeTo(Goal1, Goal2), Value = 0.8.*

• Another intuitive relation is the link among goals and preferences. In this case, given a goal and a preference, if the preference contains the goal, we assign a low grade of relation. However, since all the grades of relations between two facts are added in the membership function (see Section 3.1.1), the final grade of relation will be complete if the other goal of the preference is also a goal of the same agent.

- *relatedTo(_, _, hasgoal(A, goal(Goal)), prefer(A, goal(Goal), goal(GoalB)), Value) :- prefer(A, goal(Goal), goal(GoalB)), Value = 0.5.*
- *relatedTo(_, _, hasgoal(A, goal(Goal)), prefer(A, goal(GoalB), goal(Goal)), Value) :- prefer(A, goal(GoalB), goal(Goal)), Value = 0.5.*
- *relatedTo(_, _, hasgoal(A, goal(Goal)), prefer(A, goal(Goal), goal(GoalB)), Value) :- hasgoal(A, GoalB), prefer(A, goal(Goal), goal(GoalB)), Value = 0.5.*
- *relatedTo(_, _, hasgoal(A, goal(Goal)), prefer(A, goal(GoalB), goal(Goal)), Value) :- hasgoal(A, GoalB), prefer(A, goal(GoalB), goal(Goal)), Value = 0.5.*

• Since agent actions are related to beliefs, past actions are also related to current beliefs. In this case, we assign an intermediate grade of relation because of the temporal difference among both types of facts.

- *relatedTo(_, _, did(A, Action), believe(A, imply(do(A, Action), Something)), Value) :- believe(A, imply(do(A, Action), Something)), Value*

[*3] *Goal1* is opposite to *Goal2* if *Goal2* is equivalent to *not(Goal1)*

= 0.6.

- Following the same idea, past actions are related to past goals. In this case, the relation is high as well as the relation between present actions and present goals (including the same or opposite goals), because there is no temporal difference between both facts.
  - *relatedTo(_, _, did(A, Action), wasgoal(A, goal(Goal)), Value) :- did(A, Action), believe(A, imply(do(A, Action), Goal)), wasgoal(A, goal(Goal)), Value = 0.8.*
  - *relatedTo(_, _, did(A, Action), wasgoal(A, goal(Goal2)), Value) :- did(A, Action), believe(A, imply(do(A, Action), Goal)), wasgoal(A, goal(Goal2)), oppositeTo(Goal, Goal2), Value = 0.8.*
- Finally, past and present goals are also linked. We assign a intermediate grade of relation because of the temporal difference.
  - *relatedTo(_, _, hasgoal(A, goal(Goal)), wasgoal(A, goal(Goal)), Value) :- wasgoal(A, goal(Goal)), Value = 0.6.*