Appl Intell
DOI 10.1007/s10489-008-0139-6

Building user argumentative models

Ariel Monteserin · Analía Amandi

Abstract Knowing how a user builds his/her arguments during a discussion gives useful advantages if we want to assist the user or analyse his/her argumentative skills. This paper presents a novel mechanism to build user argumentative models, which captures the argumentative style to generate arguments. To this end, we observe how users generate arguments, and apply a generalised association rules algorithm to discover rules for argument generation. These rules depict the argumentative style of the user. They are composed of an antecedent, which represents the conditions to build an argument, and a consequent, which represents such argument. To evaluate this proposal, we show results obtained in the domain of meeting scheduling. We discovered interesting rules from a group of users discussing in that domain, and checked that about 60% of the arguments that users had generated in a test situation can be also generated from the rules previously learnt, at least partially. Finally, although this work focuses on modelling users' argumentative style, we discuss how this promising approach could be applied in different knowledge domains.

Keywords User modelling · Argumentation · Generalised association rules

A. Monteserin (✉) · A. Amandi
ISISTAN, Facultad de Ciencias Exactas, Universidad Nacional del Centro de la Pcia. Bs. As., Campus Universitario, Paraje Arroyo Seco, Tandil, Argentina
e-mail: amontese@exa.unicen.edu.ar

A. Monteserin · A. Amandi
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Buenos Aires, Argentina

A. Amandi
e-mail: amandi@exa.unicen.edu.ar

1 Introduction

During a discussion, in collaborative and cooperative environments as well as in competitive ones, users exchange proposals and arguments[1] in order to reach agreements. Proposals are motivated by their goals, and arguments are pieces of information that are generated by the user to justify such proposals or try to influence the position of the opponent during the discussion in order to persuade him/her to accept or resign a proposal. In this context, the ability to generate "good" arguments is crucial to influence the final result of the discussion. Nevertheless, all users do not have the same argumentative abilities. Therefore, if we know these user abilities we will use that information to take decisions or assist the user in a personalised way. For example, decide how tasks that demand argumentation must be allocated between users belonging to an organization, as part of the modelling of behaviour in it [1–3]; or to assist users by suggesting arguments during a discussion.

For instance, if two users A and B must agree on the time of a meeting, it will be normal to think that the agreement will not be easily reached, because the preferences about time are not the same for both users (e.g. a user A may prefer meeting in the morning, but B in the afternoon). Hence, user A can try to persuade B to accept a morning meeting saying that B has scheduled several meetings in the morning in the past, instead B can try to persuade A to reject a morning meeting because the lab is occupied (supposing that the lab is the place where A wants them to meet). Both arguments accomplish the same goal: persuading the

[1]The term "argument" is not referred to a discussion, often heated, in which a difference of opinion is expressed, but to a fact or circumstance that gives logical support to an assertion, claim, or proposal.

opponent to accept his/her proposal, but are built in different ways. User *A* makes use of historical information about *B* to build a counter-argument while user *B* employs current information to indicate that *A*'s proposal is unviable. Thus, we can preliminarily observe two different argumentative styles: one which uses historical information to attack the opponent's refusal, and another one that uses current information to refuse the opponent's proposal.

Each user has a personalised style to argue. This style characterises how the user builds arguments, in what situations he/she uses a given kind of argument and when not, and what factors of the context have an influence on these decisions. We call this style, *argumentative style*. For this reason, *A* and *B* build similar arguments, but with different information to support them and taking into account diverse factors to generate them.

In fact, users take into account the contextual information of the discussion (such as past proposals, current goals, preferences between goals and beliefs about the domain and users) to generate his/her arguments. To defend or defeat a proposal, the user evaluates the context of the discussion and determines which argument can be generated to support or refuse it. Particularly, the user must find in that context the information that satisfies the conditions under which an argument can be generated (e.g. in the previous example, *A* must find evidence about past morning meetings of *B*). These conditions implicitly form rules for argument generation. So, if the condition is satisfied in the discussion context, the argument can be built. We argue that the argumentative style of a user is exhibited by the tacit set of rules that he/she uses to generate arguments.

Therefore, to capture the argumentative style of a user, we can learn the rules for argument generation and build a user argumentative model with these, without the necessity of having a taxonomy or typology of argumentative styles.

These rules are implicitly used by the user in the discussion; whereby we will not be able to access to them directly. However, we can observe the participation of the user in the discussion, learn how the user performs the argument generation and discover the rules that depict his/her argumentative style.

In this work, we present a mechanism to build a user argumentative model, which captures the argumentative style of a user, learning the rules for argument generation that the user implicitly exhibits during the discussions. To carry out this idea, we first observe how the user builds his/her arguments during discussions, and store each argument generated and the information that the user determined as condition to build them in a knowledge base of observations. After that, we transform these observations, which are tuples of conditions and arguments, in transactions to be processed by a generalised association rule algorithm [4]. We propose to use this kind of algorithm because it allows us to obtain rules, whose antecedent is composed of the conditions to generate an argument, and whose consequent is this argument; this is the format of rules for argument generation. Moreover, in these algorithms, we can use a taxonomy of conditions and arguments with different levels of specificity in order to recognise the users' pattern of argument generation in a more abstract way (see Sect. 5 for more details).

The evaluation of this proposal was carried out in the scenario of meeting scheduling. We worked with a group of 25 users who have to arrange meetings through a distributed application. To carry out this goal, users must reach an agreement in several aspects of the meeting (topics, place, time, date, etc.), and exchange proposals and arguments through the application to do this. So, we observed how users generated their arguments (we gathered 1.234 arguments) in four different situations, and separate the observations in two sets: *training observations* and *test observations*. Then, we built an argumentative model for each user from the training observations. To validate these models, we compared for each user the arguments that we can obtain using his/her argumentative model, versus the test arguments stored on the test observations. From that comparison we found that a 42.95% of test arguments were completely generated by the argumentative models, a 16.72% were partially generated, and a 40.33% could not be generated. Also, we compared the rules obtained with other works in the area of argumentation based negotiation, for example, we automatically learnt several rules that had been explicitly defined in [5].

The work focuses on modelling the argumentative style of the user for several reasons. First, the user model is the baseline to assist the user in a personalised way, and due to the fact that there exists a wide variety of possible applications, we want to keep the modelling independent from its use.

However, although in this work we concentrate our efforts on the construction of the user argumentative models, we will not overlook its applicative side. Once the user argumentative model has been built, we can find several applications. User argumentative models can be used by a personal agent to assist the user during a discussion by suggesting automatically arguments according to his/her argumentative style. When the user is participating in a discussion, the personal agent could observe his/her participation and suggest arguments that help him/her to accomplish his/her goal in the discussion (e.g. reach a deal). As part of the evaluation of our proposal, we analyse and show how arguments can be generated from the user argumentative model. Note that the personalised assistance is directly performed from the rules, and it is not necessary to identify the type of the argumentative style to achieve this.

On the other side, building these models can be useful to discover and analyse users' argumentative skills. Knowing these skills is relevant from several perspectives: (a) to allocate tasks that demand argumentation by prioritising users

with "good" argumentative abilities; (b) to discern faults in these users' abilities; (c) since an organizational memory is defined in the area of knowledge management as a means by which knowledge from the past is used on present activities, thus resulting in higher or lower level of organizational effectiveness [3], we claim that argumentative models extracted from users can be added to this memory.

We consider that this paper makes a contribution to the state of the art in user modelling and argumentation based negotiation, since a mechanism to capture the argumentative style of users have not been modelled thus far. Moreover, defining how to automatically learn rules for argument generation through an algorithm of generalised association rules mining in turn represents an original application of this kind of algorithm.

The remainder of the paper is organised as follows. Section 2 shows a case study to illustrate the idea of learning rules for argument generation that depict the argumentative style of a user. Section 3 shows an overview about the importance of building a user argumentative model to represent his/her argumentative style. Section 4 shows how the user argumentative model is built. Section 5 shows the experimental results obtained in the domain of meeting scheduling. Section 6 shows the applicative side of our proposal. Section 7 places this work in the context of previous ones. Finally, in Sect. 8, we present some concluding remarks and future works.

2 Case study

With the purpose of illustrating our proposal, we present a case study where we can see how different users could generate different arguments in the same situation, and how the rules to generate them can be found in observations extracted from past discussions. Let's see the example. We have the following conditions and arguments observed from user *A*'s discussions:

- *A* knows that *B* accepted meeting in the *morning* last week, then *A* argues that *B* must accept a *morning* meeting because *B* accepted this *time* in the past.
- *A* knows that *C* accepted meeting in the *afternoon* last month, then *A* argues that *C* must accept an *afternoon* meeting because *C* accepted this *time* in the past.
- *A* knows that *B* made the *reservation of the lab* for last meeting, then *A* argues that *B* must make the *reservation* for tomorrow, because *B* made it in the past.
- *A* knows that *C* made the *memorandum* of the last meeting, then *A* argues that *C* must make the *memorandum*, because *C* made it in the past.

In these observations we can intuitively find several patterns that represent the rules that *A* use to generate arguments:

- *A* knows that *SOMEBODY* accepted *TIME* in the past, then *A* argues that *SOMEBODY* must accept *TIME* because *SOMEBODY* accepted this *TIME* in the past.
- *A* knows that *SOMEBODY* did *SOMETHING* for last meeting, then *A* argues that *SOMEBODY* must do *SOMETHING*, because *SOMEBODY* did it in the past.

On the other hand, we have another set of observations from user *B*:

- *B* knows that *A* wants to *discuss about vacations*, but not to *discuss about overtime payment*, then *B* argues that if *A* accepts to *discuss about overtime payment*, *B* will accept *discuss about vacations*.
- *B* knows that *C* needs to meet in the *lab*, but not in the *morning*; however *B* can only meet in the *lab* in the *morning*, then *B* argues that if *A* accepts a *morning* meeting, *B* will accept to meet in the *lab*.

And we can also observe the pattern:

- *B* knows that *SOMEBODY* wants/needs *SOMETHING*, but not *SOMETHING ELSE* that *B* wants/needs, then *B* argues that if *SOMEBODY* accepts *SOMETHING ELSE*, *B* will accept *SOMETHING*.

Now, supposing that the users have to support the same proposal, both of them want to meet in the evening, they have to generate arguments to persuade user *D* to accept this time. Both know the following:

- *D* accepted evening meetings in the past.
- *D* does not want to meet in the evening.
- *D* wants to meet on Monday.

Taking into account the rules and the information about *D*, *A* and *B* will generate different arguments to support the same proposal (*D accepts an evening meeting*):

- As *A* knows that *D* (*SOMEBODY*) accepted *evening* (*TIME*) in the past, then *A* can argue that *D* (*SOMEBODY*) must accept *evening* (*TIME*) because *he/she* accepted this time in the past.
- As *B* knows that *D* (*SOMEBODY*) wants to *meet on Monday* (*SOMETHING*), but does not want to *meet in the evening* (*SOMETHING ELSE*) that *B* wants, then *B* can argue that if *D* (*SOMEBODY*) accepts to *meet in the evening* (*SOMETHING ELSE*), *B* will accept to *meet on Monday* (*SOMETHING*).

Now, we can understand how users generate different arguments in the same situation in accordance with their argumentative style, and how the rules to achieve this can be extracted from previous arguments observed earlier. For this reason, we want to learn these rules to build user argumentative models.

3 Argumentative style and user argumentative model

Users have individual traits that can be represented in the user models [6]. We consider that one of these traits is the user argumentative style. We can build a user argumentative model, which extracts the argumentative style of a user, by observing how he/she argues during a discussion, especially how he/she builds his/her arguments.

Arguments give logical support to the proposals the user must defend, or attack proposals uttered by an opponent in order to defeat them. There are several kinds of arguments. In researches about psychology of persuasion [7, 8] some arguments types have been presented: *appeals* are used to justify a proposal; *rewards* to promise a future recompense; and *threats* to warn negative consequences if the counterpart does not accept or resign a proposal. In addition, there are different ways to build a particular argument for each argument type. For instance, an appeal can be built in several ways: as a counterexample, or appealing to prevailing practices, past promises or self-interests. However, though there are a set of well-known argument types, it is not possible to constitute an exhaustive typology of arguments [5], due to the fact that argument types are strongly related to a particular context and domain.

On the other hand, the different types of arguments are related not only to the context and the domain, but also to the way in which the arguments are generated. For each type or subtype of argument, there are a set of preconditions that must be satisfied in the context of the discussion to be able to generate it. These preconditions form rules for argument generation. For instance, in Fig. 1, we show two informal rules to generate appeals: Fig. 1a depicts a set of preconditions to generate counterexamples and Fig. 1b preconditions to build appeals to self-interest. So, when the context satisfies the argument preconditions, we have the necessary evidence to generate it. Moreover, some conditions may determine the situation in which the argument can be uttered. For example, being A and B, employee and boss respectively, A's rules should consider the relationship with B as condition to generate some type of argument, such as threats [9]. In other words, before generating a threat, user A should check whether his/her opponent is his/her boss or not, due to the fact that it is not advisable to threat to a boss.

A user who is participating in a discussion is constantly searching for evidence that allows him/her to utter an argument. In fact, a user checks before uttering an argument whether its preconditions are part of the contextual information of the discussion, and then he/she can decide to generate or not the argument. Thus, the user implicitly applies a set of rules for argument generation, which can be different for each user, depending on his/her personal conduct.

In this context, we claim that the argumentative style is implicit in the rules for argument generation that the user uses during a discussion. Moreover, since types of arguments are context and domain-dependent and rules for argument generation are influenced by user personal trait, it would be very useful to model these styles. In addition, it is worth noticing that we work under the assumption that the typology of argumentative models is unknown. That is, we have no information a priori about the styles which a user could have.

To build the user argumentative model, we must first observe how the user behaves during the discussions in a computational medium. In this medium, the user is involved in multilateral discussions, whose purpose is to reach agreements that resolve conflicts with other users, who are in this medium too. For instance, in the domain of meeting scheduling, the computational medium could be a distributed application that allows users to keep an agenda and arrange meetings by discussing with other participants: time, date, place,

Fig. 1 Rules for argument generation

Preconditions
χ *requests to execute an action* α *to* ψ &
ψ *refuses the request because to execute* α *denies an own goal* γ &
χ *knows that* ψ *have executed another action* β &
doing action β *denies the same goal* γ
then
χ *request to execute* α *to* ψ *with the follow justification:*
Do α *because it has already executed* β *and* β *deny* γ

(a) Appeal - Counterexample

Preconditions
χ *proposes to execute action* α *to* ψ &
χ *knows that* ψ *have a goal* γ &
χ *knows that do* α *implies to fulfil* γ
then
χ *request to execute* α *to* ψ *with the follow justification:*
Do α *because it will allow you to fulfil* γ

(b) Appeal to self interest

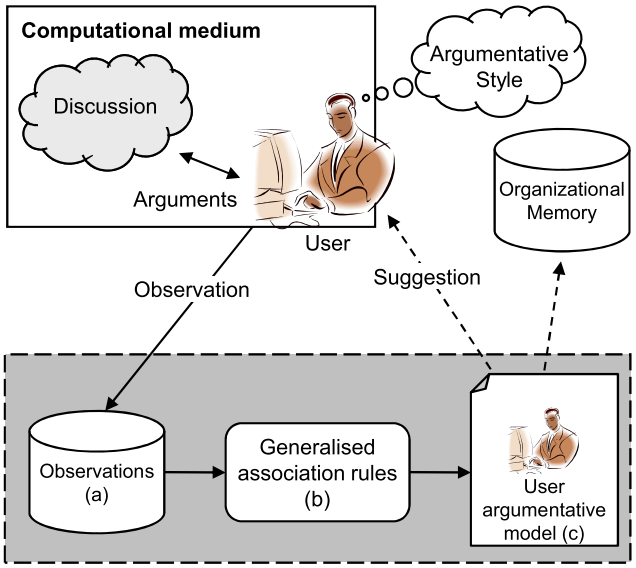

Fig. 2 Graphical representation of our proposal

among other attributes of these. In particular, we observe and store in a knowledge base of observations (Fig. 2a), the arguments that the user generates in accordance with his/her argumentative style, and the information that the user uses to support the argument.

Once the observations have been stored, we extract the rules for argument generation, which the user uses in the discussions, using an algorithm for generalised association rules mining (Fig. 2b). These rules are filtered and then form the argumentative model (Fig. 2c).

Finally, we have to determine how we use the argumentative model. In Fig. 2, the possible applications of the model are indicated by dotted arrows. We can use the model to suggest arguments to the user. In this proposal, suggestions are the arguments that we can build by using the rules for argument generation that compose the user argumentative model. For example, a personal agent could observe the discussion in which the user is arguing, check the rules, and if the conditions are true in the context of the discussion, then suggest the argument. On the other hand, we can simply store the argumentative models in the organizational memory. An organizational memory is defined in the area of knowledge management as a means by which knowledge from the past is used on present activities, thus resulting in higher or lower level of organizational effectiveness [3]. In this context, user argumentative models could be stored in this memory in order to keep the knowledge that a given user employs in the argumentation, for example, to train the argumentative skills of futures users belonging to the organization.

4 Building the user argumentative model

Once we have observed the user in some discussions, we are in the position of determining which rules, which summarise his/her argumentative style, the user utilises implicitly to generate his/her arguments.

We maintain a knowledge base O in which we gather the observations. Observations in O are tuples with the following format: (C, a) where $C = \{c_1, \ldots, c_n\}$ is the set of conditions that the user determined to generate the argument a (conditions and arguments are formally expressed). Taking into account these observations, we want to find the relations between the conditions observed and the argument generated, due to the fact that these relations constitute the rules that materialise the user argumentative style. Thus, we can learn which conditions are determined by the user to build the arguments. We think these relations can be discovered using an algorithm of association rule mining, which allows us to extract reliable patterns from the observed cases.

As introduced in [10], given a set of transactions, where each transaction is a set of items, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items too. 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 transaction in the database $(|D|)$:

$$Support(X \Rightarrow Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|D|}.$$

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:

$$Confidence(X \Rightarrow Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|}.$$

In our work, each transaction represents an observation (C, a), and we want to discover association rules of the type $X_C \Rightarrow Y_a$, where X_C is composed of items c_i (i.e. a set of conditions), and Y_a is composed of an item a_j (i.e. only one argument). In this way, our association rules will take the format $\{c_i, \ldots, c_p\} \Rightarrow a_j$.

However, that is not enough to extract the rules with which the user generate arguments, due to the fact that observations are expressed in constant terms. That is, the terms that compose such observations are constants that depict the context where the argument was generated.

For example, a condition, expressed informally, c_i: "*jack wants* (*has as goal*) *to discuss the topic salary*", which is present in an observation o_i, is expressed in constant terms because it concretely represents *Jack*'s goal to discuss the

topic *salary*. Nevertheless, we want that the conditions in the rules for argument generation are expressed in variable terms in such a way that we can instance them in any future discussion. In the example, the same condition expressed in variable terms could be "*U wants* (*has as goal*) *to discuss the topic T*" where U and T are variables that represent any user and any topic respectively, or to add more generality: "*U wants* (*has as goal*) *G*", where G is any goal of U.

In order to circumvent 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 [4]. 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 which is above it in some taxonomy. In this sense, we build a hierarchical taxonomy of conditions and arguments, in which its leaves are the conditions and arguments used by the user, and the upper levels are the same propositions but more general and expressed in variable terms. Then, the generalised association rules algorithm will especially be able to generate rules at upper levels in the taxonomy of conditions and argument. Hence, the rules will be variables.

We determine three steps to learn rules for argument generation:

1. Taxonomy building.
2. Execution of generalised association rules algorithm.
3. Post-processing of rules.

We discuss these steps below.

4.1 Taxonomy building

To execute the generalised association rules algorithm it is necessary to build a taxonomy with the facts that shape the items of the transactions, which will be the input of the algorithm too. In this work, these facts are the conditions and arguments that are present in the observations; in consequence, the taxonomy must be composed of the items in these. First, we define a language L in which conditions and arguments are expressed. Next, we outline how the taxonomy is built.

4.1.1 Language to express conditions and arguments

To build the taxonomy we need to formally express the conditions and arguments. To do this, we define a language L that is composed of the propositions that represent those facts.

- *user*(U): U is one of the users who integrates the computational environment.
- *goal*(G): G is a goal in the computational environment.
- *has_goal*(*user*(U), *goal*(G)): user U has a goal G.
- *believe*(*user*(U), B): user U believes B, in other words U has B in his/her beliefs.
- *prefer*(*user*(U), *goal*($G1$), *goal*($G2$)): user U prefers to fulfil the goal $G1$ instead of fulfilling the goal $G2$.
- *proposal*(P): P is a proposal uttered in the computational environment.
- *accept*(*user*(U), *proposal*(P)): user U accepts the proposal P.
- *refuse*(*user*(U), *proposal*(P)): user U refuses the proposal P.
- *imply*(Q, R): Q implies R, it represents the classical inference.
- *action*(A): A is an action that can be executed in the environment.
- *can_do*(*user*(U), *action*(A)): user U is able to perform the action A.
- *promised*(*user*(H), *user*(U), *proposal*(P)): user H has promised to fulfil the proposal P to user U.
- *accepted*(*user*(U), *proposal*(P)): in the past, user U accepted the proposal P.
- *argument*(*user*(H), *user*(U), *accept*()|*refuse*(), [J]): it represents an argument uttered by the user H to the user U. The goal of the argument is to support the acceptance or refusal of a proposal with the list of justifications J.

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

- *discuss_topic*(T): T is a topic that can be discussed in the meeting.
- *in_place*(P): the meeting can take place in P.
- *date*(D): the meeting can be in date D.
- *time*(M): the meeting can be at time M.

An example of an observation $o = (C, a)$ expressed in L could be:

- C: $\{c_1, c_2\}$
- c_1: *has_goal*(*user*(*user_2*), *goal*(*not*(*discuss_topic*(*topic_2*)))).
- c_2: *promised*(*user*(*user_2*), *user*(*user_1*), *discuss_topic*(*topic_2*)).
- a: *argument*(*user*(*user_1*), *user*(*user_2*), *accept*(*user*(*user_2*), *proposal*(*discuss_topic*(*topic_2*))), [*promised*(*user*(*user_2*), *user*(*user_1*), *discuss_topic*(*topic_2*))]).

The example expresses that *user_1* requests *user_2* to accept the discussion of the topic *topic_2*, because *user_2* promised it, but she does not wish to discuss it at that moment.

Fig. 3 Algorithm to build the taxonomy of propositions from the knowledge base of observations O

```
1.  root = new node (root of the taxonomy).
2.  O = {observations}.
3.  forall observation obs ∈ O {
4.    I = {items of obs}.
5.    forall items i ∈ I
6.      buildBranch(i, root).
7.  }
8.
9.  ------------------------------------------------------------------------
10. builtBranch(fact as String, root as Node): Node {
11.  if fact is in the most general expression
12.    return a new node with root as parent and fact as data.
13.  else {
14.    if the fact do not exist in the tree (root) {
15.      A_fact = Generate all ancestors of fact replacing each terminal term
                                by the respective more general term.
16.      P_fact = {array of parents}
17.      forall ancestors anc ∈ A_fact {
18.         b_anc = buildBranch(anc, root)
19.         P_fact = add b_anc. // add the ancestors as parents of fact
20.      }
21.      return a new node with P_fact as parents and fact as data.
22.    }
23.    else
24.      return the node that represent fact (avoid duplicate).
25.  }
26. }
```

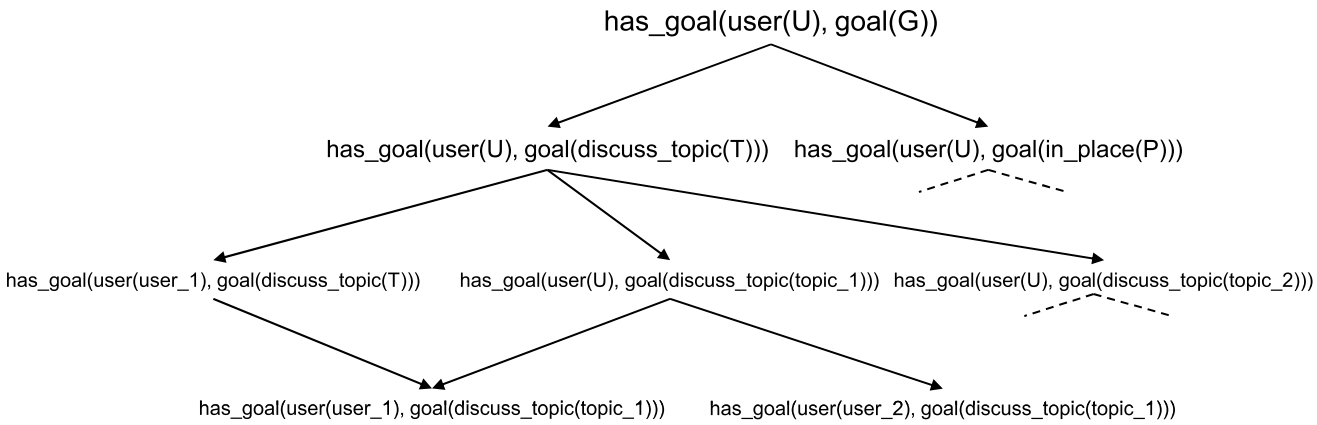

Fig. 4 Part of the taxonomy of propositions

4.1.2 Algorithm to build the taxonomy

Once we have defined the language L to build the taxonomy, we start putting in the leaves the conditions and arguments present in the observations just as they were generated by the user (i.e. c_1, c_2, and a), one condition or argument for each leaf (Fig. 3 shows the algorithm to build the taxonomy from the observations stored in O). That is, for each item (condition or argument) of each observation, we build a branch of the taxonomy by starting in this item (leaf) and ending in the root of the taxonomy (see Fig. 4).

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 has not another proposition as parameter that can be generalised) by the respective most general one (step 15 in Fig. 3). To determine this, we maintain a data structure (hash table) HT with propositions and its most general form. For example, for the proposition *has_goal* the most general form stored in HT will be *has_goal*(*user*(U), *goal*(G)); for *user*, *user*(U); for *goal*, *goal*(G); for *discuss_topic*, *discuss_topic*(T), among others.

So, given the condition *has_goal*(*user*(*user*_1), *goal*(*discuss_topic*(*topic*_1))), we add a leaf with this and create the following ancestors, taking into account that their terminal terms are *user*(*user*_1) and *discuss_topic*(*topic*_1):

– anc_1: *has_goal*(*user*(U), *goal*(*discuss_topic*(*topic*_1))) by replacing the proposition *user*(*user*_1) with *user*(U), where *user*(U) is the most general form of *user*(*user*_1).

– anc_2: $has\_goal(user(user\_1), goal(discuss\_topic(T)))$ by replacing the proposition $discuss\_topic(topic\_1)$ with $discuss\_topic(T)$, where $discuss\_topic(T)$ is the most general form of $discuss\_topic(topic\_1)$.

Next, we successively perform the same action, but with each ancestor (steps 17–20), and create a new node in the taxonomy that represents the item, whose parent are the ancestors generated previously (step 21). Following the example, the new ancestor of anc_1 is $has\_goal(user(U), goal(discuss\_topic(T)))$ (the same for anc_2); and finally, we replace $goal(discuss\_topic(T))$ with $goal(G)$ and obtain the most general expression of the initial condition. When the most general expression is found (step 11), a new node is created in the taxonomy whose parent is the root (step 12). In Fig. 4, we can observe an example of this part of the taxonomy.

4.2 Execution of generalised association rules algorithm

Having just built the taxonomy, we must generate the transactions over which the generalised association rules algorithm will work. As introduced previously, the observations are tuples with the format: (C, a) where $C = \{c^1, \ldots, c^n\}$. So, for all $o_i = (\{c_i^1, c_i^2, \ldots, c_i^n\}, a_i)$ belonging to O, we create a transaction t_i that includes the items $c_i^1, c_i^2, \ldots, c_i^n, a_i$. After obtaining the transaction t_i, we replace each item in it by the nearest ancestor that has no constants. Therefore, we eliminate any possible rule with constants[2] since we are interested in those rules that can be instantiated.

The set of transactions that will be obtained after this preprocessing is the input of the generalised association rule algorithm. It is worth noticing that our proposal is independent of the algorithm chosen. That is, we can build argumentative models with any algorithm.

To obtain generalised association rules, we must generate association rules for all the levels in the taxonomy. One approach to do this would be to take each transaction and expand each item to include all items above it in the hierarchy [4], that is, to add all the ancestors of each item in a transaction t_i to t_i. As it would be expected, when rules are generated for items at a higher level in the taxonomy, both the support and confidence increase. That is a desirable aspect due to the fact that the algorithm of association rules seeks rules with values of support and confidence higher than the minimum ones.

Although, this basic approach is quite expensive and other more efficient algorithms have been proposed, it is very simple. For this reason, and taking into account that our proposal is independent of the association rules algorithm, we used it in our experiments. Other algorithm that could be used is Cumulate, which uses several optimization strategies to reduce the number of ancestors that need to be added to each transaction [4]. Another approach, Stratification, counts itemsets by their levels in the taxonomy and uses the relationships about items in a taxonomy to reduce the number of items to be counted [4]. Several parallel algorithms to generate generalised association rules have also been proposed [11]. For more detail of implementation and comparison of performance of generalised association rules algorithm, we recommend to see [4, 12].

4.3 Post-processing generalised association rules

The post-processing of the generalised association rules can be divided in two parts. First, we filter out the rules whose format is not adjusted to $\{c_i, \ldots, c_p\} \Rightarrow a_j$. That is, once all rules have been obtained, we just select the rules whose antecedent is only composed of conditions and whose consequent is a single argument, and the remainder are filtered out. Since the association rule algorithm processes all items of a transaction alike, it does not exist a semantic difference between conditions and arguments. So, it is possible to find rules like $condition_i \Rightarrow condition_j$ or $argument_m \Rightarrow condition_n$, which fulfil the minimum levels of support and confidence, but are irrelevant to build the user argumentative model. For instance, an irrelevant rule could be $has\_goal(user(U1), goal(G1))$, $has\_goal(user(U1), goal(G2)) \Rightarrow prefer(user(U1), goal(G1), goal(G2))$. This rule is inappropriate because its three items are conditions.

The second part consists in determining how representative the rules are with respect to the argument generated by the user and gathered in the 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 total number of the conditions of a rule over the average of the conditions of the transactions that support it. It is defined as:

$$Sufficiency(r) = \frac{total\ Conditions(r)}{average\ Conditions(transaction\ Supporting(r))}.$$

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: $c_1, c_4 \Rightarrow a_1$. Three rules have minimum support and confidence. However, we can see that the rules r_1 and r_2 are not

[2] Constants are the parameters whose names start with a lower case character. For example, *user_1* is a constant instead of *User_I*, which is a variable.

Table 1 Metrical comparison of rules

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

sufficiently representative with regard to the transactions t_1, t_2 and t_3, because it is improbable that a single condition will be sufficient to generate the argument a_1, due to the fact that the conditions are not isolated in the transactions. The sufficiency metric aims to filter these rules setting a threshold that determines how sufficient the conditions (antecedent) of a rule must be to generate the consequent argument, independently of the values of support and confidence.

Table 1 details the values of the three metrics. We can observe that the rules r_1 and r_2 have a sufficiency value comparatively low with regard to the rule r_3. Therefore, a threshold of 75% only allows rule r_3 to be valid for the user argumentative model. The value of this metric can also be used by a personal agent at the moment of assisting the user, preferring to suggest arguments generated by rules with higher value of sufficiency.

5 Experimental results

The domain we chose to test our proposal was an application for meeting scheduling. In this application, the users can arrange meetings discussing date, time, place, topics to discuss during the meeting, and participants. Due to the fact that users have different goals, they must exchange arguments in order to reach an agreement.

We worked with a group of 25 users who had to generate arguments to support or defeat a set of proposals. We simulated information about the context of the discussion and we requested them to generate the greater amount of possible arguments using common sense. It is worth noticing that the users had no previous knowledge about argumentation.

The context of the discussion was composed of four participants[3] (*Jack*, *Kate*, *Ben* and *Juliet*), and sets of topics to discuss (*salary*, *vacations*, *overtime payments* and *new employees*), places where participants could meet (*room* 1, *room* 2, *lab*, *buffet*, *coffee-shop* and *restaurant*), periods of the day (*morning*, *afternoon*, *evening* and *night*), and days of the week. Moreover, for each participant we defined a set of goals, beliefs, preferences, topics that he/she could discuss, and historical information (promises uttered and proposals accepted in the past). Table 2 shows the information simulated for the participant *Jack*.

This information was presented to the users in the application for meeting scheduling, as if they should schedule a meeting. They were requested to generate arguments on four situations, one for each simulated participant, to support the proposals derived from the goals of each one, or to persuade another participant to resign theirs. For example, in the situation of participant *Jack*, some possible proposals to be supported by arguments are: *accept*(*user*(*kate*), *proposal*(*discuss_topic*(*overtime-payments*))), *refuse*(*user*(*ben*), *proposal*(*time*(*night*))), or *accept*(*user*(*juliet*), *place*(*coffee-shop*)), and so on with the others participants.

As a result, we gathered 1,234 observations, with the format detailed in the previous section, divided in a knowledge base for each user. All observations were expressed in the language L. We divided the observations of each user in two parts. The first part was composed of arguments belonging to the three first situations, we called it *training observations*, and the arguments of the fourth situation was put on the second part, which we called *test observations*. Then, we performed the process to build the user argumentative models using the training observations. First, we built a taxonomy of facts. Second, for each knowledge base of observations, we converted its content to a set of transactions, and executed the generalised association rule mining algorithm with the taxonomy and these transactions as input. Thus, we obtained a set of association rules. Then, we post-processed this set and built the argumentative model for each user. In Table 3, we show information about observations per user, rules per argumentative model, and a ratio between both parameters.

As we can see in the Table 3, there is not a relation between the amount of observations and the amount of rules in the user argumentative model. User 6 and 19 generated a lot of arguments, but the algorithm of association rules could not find relevant rules. We think this can indicate that the users did not have a well-defined argumentative style. That is, the user generated arguments but not following a specific pattern. This fact is denoted by a nil or lower ratio R/O (the number of rules divided the number of observations). On the other side, when the ratio increases, the argumentative style is well defined and the argumentative model is composed of several rules.

To illustrate the results obtained, we show the rules for argument generation learnt from the user 1 (see Fig. 5). From this user, we obtained 5 rules. In rule 5a, the antecedent, or condition for the argument generation, is *has_goal*(*user*($U1$), *goal*(*discuss_topic*($T1$)))); and the consequent (the argument) is *argument*(*user*($U2$), *user*($U1$), *accept*(*user*($U1$), *proposal*(*discuss_topic*($T1$))), [*has_goal*(*user*($U1$), *goal*(*discuss_topic*($T1$)))]). That means the

[3]The term "participant" refers to a simulated person in the context of the discussion and "user" refers to the person who participates in the experiments and whose argumentative model is built. That is, "users" generate arguments for the situations of each "participant".

Table 2 Information simulated for the participant *Jack*

U	*jack*	
has_goal(*user*(*U*), *goal*(*G*))	*G*	
	not(*place*(*room1*)); *not*(*time*(*night*))	
	and(*not*(*date*(*friday*)),*time*(*morning*))	
	time(*morning*); *time*(*evening*)	
	date(*wednesday*); *date*(*thursday*)	
	discuss_topic(*salary*); *not*(*date*(*saturday*))	
	discuss_topic(*overtime-payments*)	
	not(*discuss_topic*(*new-employees*))	
believe(*user*(*U*),*B*)	*B*	
	imply(*place*(*coffe-shop*),*time*(*morning*))	
	imply(*place*(*restaurant*),*not*(*time*(*night*)))	
	imply(*discuss_ topic*(*overtime-payments*),*user*(*kate*))	
	imply(*discuss_topic*(*salary*),*discuss_ topic*(*overtime-payments*))	
	imply(*place*(*room1*),*not*(*projection*))	
	imply(*discuss_topic*(*salary*),*projection*)	
prefer(*user*(*U*),*G1*,*G2*)	*G1*	*G2*
	goal(*not*(*time*(*night*)))	*goal*(*not*(*place*(*room1*)))
	goal(*date*(*wednesday*))	*goal*(*date*(*tuesday*))
	goal(*place*(*coffe-shop*))	*goal*(*place*(*buffet*))
	goal(*discuss_topic*(*salary*))	*goal*(*discuss_topic*(*overtime-payments*))
	goal(*discuss_topic*(*salary*))	*goal*(*not*(*discuss_topic*(*new-employees*)))
	goal(*discuss_topic*(*overtime-payments*))	*goal*(*not*(*discuss_topic*(*new-employees*)))
can_do(*action*(*A*))	*A*	
	discuss_topic(*salary*)	
	discuss_topic(*vacations*)	
	discuss_topic(*overtime-payments*)	
Historical information	*accepted*(*user*(*jack*),*discuss_topic*(*new-employees*))	
	accepted(*user*(*jack*),*discuss_topic*(*salary*))	
	accepted(*user*(*jack*),*time*(*night*))	
	accepted(*user*(*jack*),*place*(*lab*))	
	accepted(*user*(*jack*),*place*(*room2*))	
	promised(*user*(*jack*),*user*(*kate*),*discuss_ topic*(*new-employees*))	

condition the user 1 (in this case $U2$ because he is the sender of the argument) finds to express the argument, to support the proposal *discuss_topic*($T1$), is the user $U1$ has the same goal. Moreover, we can observe some similar rules, but with different level of detail, such as rules 5d and 5e. Although these rules are redundant since 5d is included in 5e, are valid for the argument generation. Note that this difference of generality is obtained by the generalised association rule algorithm.

It is also worth noticing that we cannot be sure that this set of rules represents completely the argumentative style of the user, but we claim that it is a good result since the number of observations for this user was only 25.

To evaluate that the rules of the argumentative models depict the argumentative styles of the users, we compared for each user the arguments that we can obtain using his/her argumentative model, versus the arguments stored on the test observations. The main idea was to validate the rules that compose the argumentative models by determining if the arguments generated by the user in a new situation (test observations) could be generated by means of the rules for argument generation learnt from the training observations. For each test argument (argument stored in test observations), we established one of the three values: NO (it cannot be generated from the argumentative model); PARTIAL (it can be partially generated, that is, the test argument was gener-

Table 3 Observations and rules obtained by user, and results of the evaluations

User	Observations			Rules	R/O	#			%		
	Total	Train.	Test			NO	PART	YES	NO	PART	YES
1	30	25	5	5	0.2	2	1	2	40	20	40
2	30	24	6	3	0.13	5	1	0	83.33	16.67	0
3	15	12	3	4	0.33	2	0	1	66.67	0	33.33
4	37	33	4	1	0.03	4	0	0	100	0	0
5	119	91	28	16	0.18	9	3	16	32.14	10.71	57.14
6	103	167	12	1	0.01	10	0	2	83.33	0	16.67
7	43	31	12	8	0.26	3	0	9	25	0	75
8	11	10	1	3	0.3	1	0	0	100	0	0
9	13	11	2	3	0.27	2	0	0	100	0	0
10	31	25	6	1	0.04	4	2	0	66.67	33.33	0
11	37	30	7	2	0.07	5	2	0	71.43	28.57	0
12	83	64	19	1	0.02	16	0	3	84.21	0	15.79
13	44	35	9	1	0.03	6	0	3	66.67	0	33.33
14	20	18	2	7	0.39	0	0	2	0	0	100
15	37	30	7	2	0.07	5	2	0	71.43	28.57	0
16	85	67	18	8	0.12	0	0	18	0	0	100
17	21	16	5	4	0.25	3	1	1	60	20	20
18	42	36	6	7	0.19	2	2	2	33.33	33.33	33.33
19	110	151	69	1	0.01	32	12	25	46.38	17.39	36.23
20	37	30	7	2	0.07	6	0	1	85.71	0	14.29
21	58	43	15	2	0.05	2	8	5	13.33	53.33	33.33
22	64	44	20	5	0.11	0	6	14	0	30	70
23	81	66	15	6	0.09	0	1	14	0	6.67	93.33
24	15	8	7	1	0.13	2	2	3	28.57	28.57	42.86
25	68	48	20	4	0.08	2	8	10	10	40	50
Total	1234	1115	305	98		123	51	131	40.33	16.72	42.95

Fig. 5 Rules for argument generation extracted for user 1

a. *has_goal(user(U1), goal(discuss_topic(T1)))) ⇒ argument(user(U2),user(U1), accept(user(U1), proposal(discuss_topic(T1))), [has_goal(user(U1), goal(discuss_topic(T1)))])*
b. *has_goal(user(U1), goal(not(date(D1)))) ⇒ argument(user(U2), user(U1), accept(user(U1), proposal(date(D2))), [has_goal(user(U1), goal(not(date(D1))))])*
c. *believe(user(U2), not(place(P1))) ⇒ argument(user(U2), user(U1), refuse(user(U1), proposal(place(P1)))), [believe(user(U2), not(place(P1)))])*
d. *accepted(user(U1), proposal(discuss_topic(T1))) ⇒argument(user(U2), user(U1), accept(user(U1), proposal(discuss_topic(T1)))), [accepted(user(U1), proposal(discuss_topic(T1)))])*
e. *accepted(user(U1), proposal(P1)) ⇒ argument(user(U2), user(U1),accept(user(U1), proposal(P1))), [accepted(user(U1), proposal(P1))])*

ated with more conclusions or premises that those present in the rule), and YES (it can be completely generated). For example, given the rules of user 1 (Fig. 5), the values for the following test arguments were:

– *has_goal*(*user*(*juliet*), *goal*(*not*(*place*(*buffet*)))) ⇒ *argument*(*user*(*juliet*), *user*(*ben*), *refuse*(*user*(*ben*), *proposal*(*place*(*buffet*)))), [*has_goal*(*user*(*juliet*), *goal*(*not* (*place*(*buffet*))))]): this argument cannot be generated, since no rule of the argumentative model matches this pattern.
– *accepted*(*user*(*kate*), *proposal*(*discuss_topic*(*overtime-payments*))), *can_do*(*user*(*kate*), *discuss_topic*(*overtime-payments*)) ⇒ *argument*(*user*(*juliet*), *user*(*kate*), *accept* (*user*(*kate*), *proposal*(*discuss_topic*(*overtime-payments*)))),

Fig. 6 Rules for argument generation

antecedent
- believe(user(U2),not(place(P1)))

consequent
- argument(user(U2),user(U1),refuse(user(U1), proposal(place(P1)))), [believe(user(U2),not(place(P1)))])

(a) Trivial appeal rule

antecedent
- accepted(user(U1),proposal(P1))

consequent
- argument(user(U2),user(U1),accept(user(U1), proposal(P1))), [accepted(user(U1),proposal(P1))])

(b) Prevailing practice appeal rule

antecedent
- has_goal(user(U1), goal(not(date(D1))))
- has_goal(user(U1), goal(discuss_topic(T1)))
- prefer(user(U1), goal(discuss_topic(T1)), goal(not(date(D1))))

consequent
- argument(user(U2), user(U1), accept(user(U1), proposal(date(D1))), [prefer(user(U1), goal(discuss_topic(T1)), goal(not(date(D1))))])

(c) Incomplete threat rule

antecedent
- promised(user(U1), user(U2), discuss_topic(T1)))

consequent
- argument(user(U2,user(U1),accept(user(U1), proposal(discuss_topic(T1)))), [promised(user(U1), user(U2), discuss_topic(T1))))])

(d) Past promise appeal rule

[*accepted*(*user*(*kate*), *proposal*(*discuss_topic*(*overtime-payments*)))]): this test argument can be generated, but partially, because the condition *can_do*(*user*(*U*), *discuss_topic*(*T*)) is not present in the conditions of rule 5d.
- *has_goal*(*user*(*jack*), *goal*(*not*(*date*(*saturday*)))) ⇒ *argument*(*user*(*juliet*), *user*(*jack*), *accept*(*user*(*jack*), *proposal*(*date*(*friday*))), [*has_ goal*(*user*(*jack*), *goal*(*not*(*date* (*saturday*))))]): it can be completely generated by rule 5b.

The results of the evaluations are showed in Table 3. In the last columns, we can see the total (#) and percentage (%) of NO (123 and 40.33%), PARTIAL (51 and 16.72%) and YES (131 and 42.95%) observations. As we can observe, a good percentage of the arguments stored in the test observations could be generated with the rules that compose the argumentative models. That means that the rules obtained from the training observations model the argumentative style with which users generate their arguments, at least partially. Note that for the 44% of the users, the total of arguments that can be generated by the argumentative model is higher than the total of arguments that cannot. In addition, we could, at least partially, generate some arguments for the 88% of the users.

On the other hand, as we introduced in Sect. 3, in the area of argumentation-based negotiation [13], rules for argument generation have been explicitly defined. In the work of Kraus et al. [5], several rules have been detailed taking into account the works in the area of psychology of the persuasion [7, 8]. The underlying idea is that finding the same rules that others works have specified from others perspectives (i.e. psychological theories), shows signs that the proposed mechanism to learn rules for argument generation works well. That is, the studies in the area of argumentation-based negotiation help us to validate our mechanism for building user argumentative models. For this reason, we show in Fig. 6 some rules discovered using the process specified above, which were defined in [5] too. In our work, these rules were found for several users. We analyse them below:

- Trivial appeal (Fig. 5a): it is a simple appeal: $U2$ knows that the place $P1$ is occupied (that means *not*(*place*($P1$))), and he/she uses this information to justify the refusal of $P1$ as the place for the meeting (this rule was found in 12 users).
- Appeal to prevailing practice (Fig. 5b): this is an appeal that resorts to historical information (*accepted*(*user*(U), *proposal*(P))) to persuade an opponent to accept a proposal. That is, "if you accepted the proposal *P1* in the past, now you should accept it too". In this rule, the ability of abstraction that the generalised association rules give to our approach can be assessed since all arguments present in the observation have the proposal instantiated (rule found in 16 users).
- Threat (Fig. 5c): we suppose this rule represents an incomplete one for threat generation. To be completed, this rule should have a proposition that represents $U2$'s intention to refuse the discussion of topic $T1$ in the justification of the argument. In other words, if $U1$ does not accept the date $D1$, $U2$ do not accept to discuss the topic $T1$. Finding incomplete rules can aid to improve the argumentative abilities of the user (rule found in 2 users).
- Appeal to past promise (Fig. 5d): it is also a simple appeal, in which $U2$ reminds $U1$ that he/she promised to discuss topic $T1$ (rule found in 18 users).

Note that some rules appear more frequently than others. That happens because some rules are simpler than others. Simple rules, such as appeals to prevailing practice, are easily discovered with few observations. Intuitively, this is because the number of conditions is low, so it is expected that smaller patterns on the conditions are more frequent than patterns with a high number of conditions. On the contrary,

some rules, such as threats, are not as common (they were only discovered from 2 users) as others, because it is structurally more complex.

6 Applications of the user argumentative model

As we said in the introduction, there are several applications for a user argumentative model. In this section, we will give an overview about two possible applications: the personalised suggestion of arguments and the discovery and analysis of users' argumentative skills.

6.1 Suggesting personalised arguments

As we introduced above, when a user participates in a discussion he/she must generate arguments to persuade his/her opponents. To achieve this, he/she builds these arguments according to his/her argumentative style. So, if we have a user argumentative model that depicts that style, we could automatically generate arguments in a personalised way.

We assume that we can access to the contextual information of the discussion in which the user is participating, and that this information is expressed in the language L described before. For example, a personal agent [14], which observes the computational environment, can detect the intentions [15, 16] of the user or can observe both the proposal that he/she utters and the additional information of the discussion. Once the intentions or the proposals are detected and the information is gathered, the personal agent can use the rules stored in the user argumentative model to generate arguments that give support to the intention and proposals of the user. The argument generation is performed by checking the rules, whose argument matches the proposal that we want to support, considering the contextual information of discussion. Then if the conditions of the rule are met, we will be allowed to generate the argument.

For example, given two users John and Peter, if the proposal of John is that Peter accepts to discuss about salary (*accept*(*user*(*peter*), *proposal*(*discuss_topic*(*salary*)))), the rules, whose condition we have to check, must generate an argument with the format: *argument*(*user*(*john*), *user*(*peter*), *accept*(*user*(*peter*), *proposal*(*discuss_topic*(*salary*)))), [J]), where J must be instantiated with any justification.

Once the arguments have been generated, they can be presented to the user in a graphical interface so that the user can choose one, modifying or not some of its parts, or reject the suggestions. This can be useful as a feedback for the correctness of the rules in the user argumentative model too. The order in which the arguments are presented to the user can be determined by the values of the metrics of the rule that generated it.

To exemplify this application, we will show how arguments can be generated using the rules included in the argumentative model of user 1 (Fig. 5), namely John. Notice that the argumentative style of John is modelled by the rules included in the argumentative model. Given a discussion between John and Peter, we suppose that both want to arrange a meeting to negotiate a contract. John wants to meet on Monday morning in his office. This information expressed in the language L is: *has_goal*(*user*(*john*), *goal*(*discuss_topic*(*contract*))), *has_goal*(*user*(*john*), *goal*(*date*(*monday*))), *has_goal*(*user*(*john*), *goal*(*time*(*morning*))), *has_goal*(*user*(*john*), *goal*(*place*(*my_office*))). Moreover, we can access to both the next information about Peter and the context of the discussion: *believe*(*user*(*john*), *not*(*place*(*meeting_room*))) (the meeting room is occupied), *accepted*(*user*(*peter*), *proposal*(*and*(*date*(*monday*), *time*(*morning*)))) (in the past, Peter accepted a morning meeting on Monday), *has_goal*(*user*(*peter*), *goal*(*not*(*date*(*friday*)))) (Peter has the goal of not meeting on Friday), among others.

In this scenario, we can see several situations to generate arguments from the perspective of John. For example:

– Peter proposed to meet in the meeting room.

As John wants to meet in his office, he has to generate an argument that refuses such proposal. So, we can suggest a trivial appeal with rule c (Fig. 5c), saying that the meeting room is occupied. The argument suggested will be: *argument*(*user*(*john*), *user*(*peter*), *refuse*(*user*(*peter*), *proposal*(*place*(*meeting_room*)))), [*believe*(*user*(*john*), *not*(*place*(*meeting_room*)))]).

– John proposed a morning meeting on Monday and Peter refused it, justifying that he cannot meet on Monday morning.

In this situation, John should give Peter a justification in order to persuade him to meet that day at that time. This justification can be reached with an argument generated by rule e (Fig. 5e). The argument suggested will be: *argument*(*user*(*john*), *user*(*peter*), *accept*(*user*(*peter*), *proposal*(*and*(*date*(*monday*), *time*(*morning*)))))), [*accepted*(*user*(*peter*), *proposal*(*and*(*date*(*monday*), *time*(*morning*))))]), or *argument*(*user*(*john*), *user*(*peter*), *accept*(*user*(*john*), *proposal*(*date*(*monday*))), [*has_goal*(*user*(*peter*), *goal*(*not*(*date*(*friday*))))]), which is not a complete argument, but could be an option.

6.2 Discovering and analysing users' argumentative skills

As we introduced earlier, modelling the argumentative style of a user can be interesting in the area of knowledge management. We keep on focus three perspectives in which user argumentative models have possible applications. We will briefly comment these applications:

– Discerning faults in the users' argumentative abilities: as we showed in Fig. 5c, we can detect some problems in users' argumentative abilities. Another problem could be seen in Table 3 with the users 6 and 19, they generate a lot of arguments, but apparently without a defined style. Once the problems are detected, we can take decisions to correct them.
– Task allocation: if we have to allocate a task among different users, the information about user's skills helps us to take decisions. So, we could analyse the argumentative models and determine if a user is able to perform a task that demands argumentation (i.e. to negotiate with a provider). That is, for example, if the user has no faults in his/her user's argumentative abilities, he/she will be able to perform the task properly.
– Organizational memory: as we said before, it could be useful to include in the organizational memory the user argumentative models in order to have this information for future decisions, or to train the argumentative skills of future employees in an organization.

7 Related work

It is hard to set this work in the context of the actual literature. First, we have not found works related to the construction of user argumentative models, and second, our proposal is interdisciplinary in its constitution as well as in its application.

On the one hand, the rules for argument generation that are learnt in our proposal and form the user argumentative model, have been used in the area of argumentation based negotiation among intelligent agents [13]. Kraus et al. [5] determine a set of possible argument types, and for each argument type they define preconditions for its usage. Only if the preconditions are fulfilled, the argument will be used. In the work of Ramchurn et al. [17] and Sierra et al. [9], the authors define preconditions for argument generation too, each one taking into account several factors (authority, utility of the proposals, etc.). However, these works define these rules statically, and do not establish a mechanism to learn them. With regard to the different factors that could be taken into account in the conditions of the rules, our proposal allows us to consider all that, storing the information about the context as a condition in the observations.

With respect to user models, they have been applied in the field of argument interpretation. Zukerman et al. [18] incorporate a user model in the process of argument evaluation, that is arguments that a user receives from other ones; but this work excludes the argument generation. The user model is represented by a Bayesian network. This network represents the users' beliefs, which are born in mind by an argument-interpretation mechanism. In addition to the differences between the user model representations (we use associations rules and they use Bayesian networks), both works are focused on different mechanisms of the argumentation process. Zukerman et al.'s work focuses on argument evaluation and we focus on argument generation.

8 Conclusions and future work

This paper contributes to the areas of user modelling and argumentation based negotiation. We have been presented a original mechanism to build user argumentative models that captures the argumentative styles of the users, which have not been modelled to date. For this task, we observe how the users generate his/her argument during a discussion in a computational medium, and on the basis of these observations, we apply a generalised association rules algorithm to extract rules for argument generation. The rules obtained are composed of an antecedent and a consequent. The antecedent is a set of conditions that the context of the discussion has to fulfil in order to be able to generate an argument, which form the consequent of the rule. We defined some filters to select the most interesting rules. First we select the rules whose format is interesting for our goals, and then we calculate a metric to determine if the conditions of the rule are sufficient to generate the argument.

The evaluation of this proposal was carried out in the scenario of meeting scheduling. We worked with a group of 25 users who have to arrange meetings in a distributed application. We observed how users generated their arguments, we learnt the rules for argument generation that they use tacitly, and built an argumentative model for each of them. We learnt several rules for each user though the number of observations was limited, with the exception of some users with a high number of observation, but with a low number of rules discovered.

To validate this result, we found that a 42.95% of the arguments generated by users in a test situation, can be completely generated using the user argumentative models and a 16.72% can be partially generated. This indicates that the rules in the argumentative model represent the argumentative style of the user, at least partially.

In addition, we compared the rules obtained with other works in the area of argumentation based negotiation. We found several coincidences between the rules obtained by our proposal and the rules defined explicitly in the work of Kraus et al. [5]. Moreover, we showed for several situations how arguments can be generated from the user argumentative model, and gave an overview about the applicative scope of our work.

There are a number of future research directions, such as determining how to analyse the user argumentative model to detect flaws in the argumentative skill of the users, and which corrective actions take to solve this. In addition, we

can apply the argumentative model to evaluate the arguments that the user receives. Most importantly, future research will assess how the users respond to the suggestion of a personal agent, and how this interaction between user and agent could be taken into account in the suggestion of arguments.

References

1. Ilgen DR, Hulin CL (2000) Computational modeling of behavior organizations: the third scientific discipline. American Psychological Association, Washington. Chap. 1
2. Hedberg B (1981) How organizations learn and unlearn. In: Nystron PC, Starbuck WH (eds) Handbook of organizational design. Oxford University Press, New York, pp 3–27
3. Stein E, Zwass V (1995) Actualizing organizational memory with information systems. Inform Syst Res 6(2):85–117
4. Srikant R, Agrawal R (1997) Mining generalized association rules. Future Gener Comput Syst 13(2–3):161–180
5. Kraus S, Sycara K, Evenchik A (1998) Reaching agreements through argumentation: a logical model and implementation. Artif Intell 104(1–2):1–69
6. Brusilovsky P, Millán E (2007) User models for adaptive hypermedia and adaptive educational systems. In: The adaptive web. Springer, Berlin, pp 3–53
7. Karlins M, Abelson HI (1970) Persuasion: how opinions and attitudes are changed. Springer, Berlin
8. O'Keefe D (1990) Persuasion: theory and research. SAGE, London
9. Sierra C, Jennings NR, Noriega P, Parsons S (1998) A framework for argumentation-based negotiation. In: Proceedings of the 4th international workshop on agent theories, architectures and languages, Rode Island, USA, pp 177–192
10. Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on management of data, Washington, DC, pp 207–216
11. Shintani T, Kitsuregawa M (1998) Parallel mining algorithms for generalized association rules with classification hierarchy. In Proceedings of the ACM SIGMOD international conference on management of data, SIGMOD 1998, Seattle, WA, pp 25–36
12. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th international conference very large data bases, Santiago de Chile, Chile, pp 487–499
13. Rahwan I, Ramchurn SD, Jennings NR, McBurney P, Parsons S, Sonenberg L (2003) Argumentation-based negotiation. Knowl Eng Rev 18(4):343–375
14. Maes P (1994) Agents that reduce work and information overload. Commun ACM 37(7):30–40
15. Kautz H (1987) A formal theory of plan recognition. PhD thesis, Department of Computer Science, University of Rochester
16. Charniak E, Goldman RP (1993) A Bayesian model of plan recognition. Artif Intell 64(1):53–79
17. Ramchurn SD, Jennings R, Sierra C (2003) Persuasive negotiation for autonomous agents: a rhetorical approach. In: Proceedings of the IJCAIl workshop on computational models of natural argument, Acapulco, Mexico, pp 9–17
18. Zukerman I, George S, George M (2003) Incorporating a user model into an information theoretic framework for argument interpretation. In: Proceedings of the ninth international conference on user modeling, Johnstown, PA, pp 106–116

Ariel Monteserin is a PhD student at ISISTAN Research Institute at Univ. Nac. del Centro de la Pcia. de Bs. As. (UNCPBA), Argentina. His main interests are negotiation and argumentation among intelligent agents. He received his Bachelor degree in Systems in 2003. He is a teaching assistant in the Computer Science Department at UNCPBA.

Analía Amandi is a professor in the Computer Science Department at Univ. Nac. del Centro de la Pcia. de Bs. As. (UNCPBA), where she leads the ISISTAN Research Institute's Knowledge Management Group. She received her PhD in 1997. Her research interests include personal assistants and knowledge management.