

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

Building User Argumentative Models

Ariel Monteserin, Analía Amandi

ISISTAN, Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Pcia. Bs. As.
Campus Universitario, Paraje Arroyo Seco, Tandil, Argentina
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
{amontese, amandi}@exa.unicen.edu.ar

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.

1. Introduction

During a discussion, in collaborative and cooperative environments as well as in competitive ones, users exchange proposals and arguments¹ in order to reach agree-

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

ments. 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 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.

1
2
3
4
5
6
7
8
9
10 Each user has a personalised style to argue. This style characterises how the user
11 builds arguments, in what situations he/she uses a given kind of argument and when
12 not, and what factors of the context have an influence on these decisions. We call this
13 style, *argumentative style*. For this reason, *A* and *B* build similar arguments, but with
14 different information to support them and taking into account diverse factors to gener-
15 erate them.
16
17
18
19
20

21 In fact, users take into account the contextual information of the discussion (such
22 as past proposals, current goals, preferences between goals and beliefs about the do-
23 main and users) to generate his/her arguments. To defend or defeat a proposal, the
24 user evaluates the context of the discussion and determines which argument can be
25 generated to support or refuse it. Particularly, the user must find in that context the in-
26 formation that satisfies the conditions under which an argument can be generated (e.g.
27 in the previous example, *A* must find evidence about past morning meetings of *B*).
28 These conditions implicitly form rules for argument generation. So, if the condition is
29 satisfied in the discussion context, the argument can be built. We argue that the argu-
30 mentative style of a user is exhibited by the tacit set of rules that he/she uses to gener-
31 ate arguments.
32
33
34
35
36
37
38
39
40
41

42 Therefore, to capture the argumentative style of a user, we can learn the rules for
43 argument generation and build a user argumentative model with these, without the ne-
44 cessity of having a taxonomy or typology of argumentative styles.
45
46
47

48 These rules are implicitly used by the user in the discussion; whereby we will not
49 be able to access to them directly. However, we can observe the participation of the
50 user in the discussion, learn how the user performs the argument generation and dis-
51 cover the rules that depict his/her argumentative style.
52
53
54
55
56
57

1
2
3
4
5
6
7
8
9
10 In this work, we present a mechanism to build a user argumentative model, which
11 captures the argumentative style of a user, learning the rules for argument generation
12 that the user implicitly exhibits during the discussions. To carry out this idea, we first
13 observe how the user builds his/her arguments during discussions, and store each ar-
14 gument generated and the information that the user determined as condition to build
15 them in a knowledge base of observations. After that, we transform these observa-
16 tions, which are tuples of conditions and arguments, in transactions to be processed by
17 a generalised association rule algorithm [4]. We propose to use this kind of algorithm
18 because it allows us to obtain rules, whose antecedent is composed of the conditions
19 to generate an argument, and whose consequent is this argument; this is the format of
20 rules for argument generation. Moreover, in these algorithms, we can use a taxonomy
21 of conditions and arguments with different levels of specificity in order to recognise
22 the users' pattern of argument generation in a more abstract way (See Section 5 for
23 more details).
24
25
26
27
28
29
30
31
32
33
34
35

36 The evaluation of this proposal was carried out in the scenario of meeting schedul-
37 ing. We worked with a group of 25 users who have to arrange meetings through a dis-
38 tributed application. To carry out this goal, users must reach an agreement in several
39 aspects of the meeting (topics, place, time, date, etc.), and exchange proposals and ar-
40 guments through the application to do this. So, we observed how users generated their
41 arguments (we gathered 1.234 arguments) in four different situations, and separate the
42 observations in two sets: *training observations* and *test observations*. Then, we built
43 an argumentative model for each user from the training observations. To validate
44 these models, we compared for each user the arguments that we can obtain using
45 his/her argumentative model, versus the test arguments stored on the test observa-
46 tions.
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 tions. From that comparison we found that a 42.95 % of test arguments were com-
11 pletely generated by the argumentative models, a 16.72 % were partially generated,
12 and a 40.33 % could not be generated. Also, we compared the rules obtained with
13 other works in the area of argumentation based negotiation, for example, we auto-
14 matically learnt several rules that had been explicitly defined in [5].
15
16

17
18
19 The work focuses on modelling the argumentative style of the user for several rea-
20 sons. First, the user model is the baseline to assist the user in a personalised way, and
21 due to the fact that there exists a wide variety of possible applications, we want to
22 keep the modelling independent from its use.
23
24

25
26
27 However, although in this work we concentrate our efforts on the construction of
28 the user argumentative models, we will not overlook its applicative side. Once the
29 user argumentative model has been built, we can find several applications. User ar-
30 gumentative models can be used by a personal agent to assist the user during a discus-
31 sion by suggesting automatically arguments according to his/her argumentative style.
32
33 When the user is participating in a discussion, the personal agent could observe
34 his/her participation and suggest arguments that help him/her to accomplish his/her
35 goal in the discussion (e.g. reach a deal). As part of the evaluation of our proposal, we
36 analyse and show how arguments can be generated from the user argumentative
37 model. Note that the personalised assistance is directly performed from the rules, and
38 it is not necessary to identify the type of the argumentative style to achieve this.
39
40
41
42
43
44
45
46
47

48 On the other side, building these models can be useful to discover and analyse us-
49 ers' argumentative skills. Knowing these skills is relevant from several perspectives:
50 (a) to allocate tasks that demand argumentation by prioritising users with "good" ar-
51 gumentative abilities; (b) to discern faults in these users' abilities; (c) since an organ-
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 izational memory is defined in the area of knowledge management as a means by
11 which knowledge from the past is used on present activities, thus resulting in higher
12 or lower level of organizational effectiveness [3], we claim that argumentative models
13 extracted from users can be added to this memory.
14
15
16

17
18 We consider that this paper makes a contribution to the state of the art in user mod-
19 elling and argumentation based negotiation, since a mechanism to capture the argu-
20 mentative style of users have not been modelled thus far. Moreover, defining how to
21 automatically learn rules for argument generation through an algorithm of generalised
22 association rules mining in turn represents an original application of this kind of algo-
23 rithm.
24
25
26
27
28

29 The remainder of the paper is organised as follow. Section 2 shows a case study to
30 illustrate the idea of learning rules for argument generation that depict the argumenta-
31 tive style of a user. Section 3 shows an overview about the importance of building a
32 user argumentative model to represent his/her argumentative style. Section 4 shows
33 how the user argumentative model is built. Section 5 shows the experimental results
34 obtained in the domain of meeting scheduling. Section 6 shows the applicative side of
35 our proposal. Section 7 places this work in the context of previous ones. Finally, in
36 Section 8, we present some concluding remarks and future works.
37
38
39
40
41
42
43
44

45 **2. Case study**

46
47
48 With the purpose of illustrating our proposal, we present a case study where we can
49 see how different users could generate different arguments in the same situation, and
50 how the rules to generate them can be found in observations extracted from past dis-
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 cussions. Let's see the example. We have the following conditions and arguments ob-
11 served from user *A*'s discussions:

- 12
13
14 – *A* knows that *B* accepted meeting in the *morning* last week, then *A* argues that *B*
15 must accept a *morning* meeting because *B* accepted this *time* in the past.
16
17 – *A* knows that *C* accepted meeting in the *afternoon* last month, then *A* argues that *C*
18 must accept an *afternoon* meeting because *C* accepted this *time* in the past.
19
20 – *A* knows that *B* made the *reservation of the lab* for last meeting, then *A* argues that
21 *B* must make the *reservation* for tomorrow, because *B* made it in the past.
22
23 – *A* knows that *C* made the *memorandum* of the last meeting, then *A* argues that *C*
24 must make the *memorandum*, because *C* made it in the past.
25
26
27
28
29

30
31 In these observations we can intuitively find several patterns that represent the rules
32 that *A* use to generate arguments:

- 33
34
35 – *A* knows that *SOMEBODY* accepted *TIME* in the past, then *A* argues that
36 *SOMEBODY* must accept *TIME* because *SOMEBODY* accepted this *TIME* in the
37 past.
38
39
40 – *A* knows that *SOMEBODY* did *SOMETHING* for last meeting, then *A* argues that
41 *SOMEBODY* must do *SOMETHING*, because *SOMEBODY* did it in the past.
42
43
44

45
46 On the other hand, we have another set of observations from user *B*:

- 47
48 – *B* knows that *A* wants to *discuss about vacations*, but not to *discuss about overtime*
49 *payment*, then *B* argues that if *A* accepts to *discuss about overtime payment*, *B* will
50 accept *discuss about vacations*.
51
52
53
54
55
56
57

- 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
- *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*.

18 And we can also observe the pattern:

- 19
20
21
22
23
24
25
26
- *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*.

27 Now, supposing that the users have to support the same proposal, both of them want
28 to meet in the evening, they have to generate arguments to persuade user *D* to accept
29 this time. Both know the following:
30
31
32

- 33
34
35
36
37
38
39
- *D* accepted evening meetings in the past.
 - *D* does not want to meet in the evening.
 - *D* wants to meet on Monday.

40
41
42
43
44

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):

- 45
46
47
48
49
50
51
52
53
54
55
56
57
- 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*

1
2
3
4
5
6
7
8
9
10 can argue that if *D* (*SOMEBODY*) accepts to *meet in the evening* (*SOMETHING*
11 *ELSE*), *B* will accept to *meet on Monday* (*SOMETHING*).
12
13
14

15
16 Now, we can understand how users generate different arguments in the same situation
17
18 in accordance with their argumentative style, and how the rules to achieve this can be
19
20 extracted from previous arguments observed earlier. For this reason, we want to learn
21
22 these rules to build user argumentative models.
23
24

25 **3. Argumentative Style and User Argumentative Model**

26

27
28 Users have individual traits that can be represented in the user models [6]. We con-
29
30 sider that one of these traits is the user argumentative style. We can build a user ar-
31
32 gumentative model, which extracts the argumentative style of a user, by observing
33
34 how he/she argues during a discussion, especially how he/she builds his/her argu-
35
36 ments.
37

38 Arguments give logical support to the proposals the user must defend, or attack
39
40 proposals uttered by an opponent in order to defeat them. There are several kinds of
41
42 arguments. In researches about psychology of persuasion [7, 8] some arguments types
43
44 have been presented: *appeals* are used to justify a proposal; *rewards* to promise a fu-
45
46 ture recompense; and *threats* to warn negative consequences if the counterpart does
47
48 not accept or resign a proposal. In addition, there are different ways to build a particu-
49
50 lar argument for each argument type. For instance, an appeal can be built in several
51
52 ways: as a counterexample, or appealing to prevailing practices, past promises or self-
53
54 interests. However, though there are a set of well-known argument types, it is not pos-
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 sible to constitute an exhaustive typology of arguments [5], due to the fact that argu-
11 ment types are strongly related to a particular context and domain.
12
13

14 **Fig. 1.** Rules for argument generation.
15

16
17 On the other hand, the different types of arguments are related not only to the context
18 and the domain, but also to the way in which the arguments are generated. For each
19 type or subtype of argument, there are a set of preconditions that must be satisfied in
20 the context of the discussion to be able to generate it. These preconditions form rules
21 for argument generation. For instance, in Figure 1, we show two informal rules to
22 generate appeals: Figure 1.a depicts a set of preconditions to generate counterexam-
23 ples and Figure 1.b preconditions to build appeals to self-interest. So, when the con-
24 text satisfies the argument preconditions, we have the necessary evidence to generate
25 it. Moreover, some conditions may determine the situation in which the argument can
26 be uttered. For example, being *A* and *B*, employee and boss respectively, *A*'s rules
27 should consider the relationship with *B* as condition to generate some type of argu-
28 ment, such as threats [9]. In other words, before generating a threat, user *A* should
29 check whether his/her opponent is his/her boss or not, due to the fact that it is not ad-
30 visable to threaten to a boss.
31
32

33
34 A user who is participating in a discussion is constantly searching for evidence that
35 allows him/her to utter an argument. In fact, a user checks before uttering an argu-
36 ment whether its preconditions are part of the contextual information of the discus-
37 sion, and then he/she can decide to generate or not the argument. Thus, the user im-
38 plicitly applies a set of rules for argument generation, which can be different for each
39 user, depending on his/her personal conduct.
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 In this context, we claim that the argumentative style is implicit in the rules for ar-
11 gument generation that the user uses during a discussion. Moreover, since types of ar-
12 guments are context and domain-dependent and rules for argument generation are in-
13 fluenced by user personal trait, it would be very useful to model these styles. In
14 addition, it is worth noticing that we work under the assumption that the typology of
15 argumentative models is unknown. That is, we have no information a priori about the
16 styles which a user could have.
17
18
19
20
21
22

23 To build the user argumentative model, we must first observe how the user behaves
24 during the discussions in a computational medium. In this medium, the user is in-
25 volved in multilateral discussions, whose purpose is to reach agreements that resolve
26 conflicts with other users, who are in this medium too. For instance, in the domain of
27 meeting scheduling, the computational medium could be a distributed application that
28 allows users to keep an agenda and arrange meetings by discussing with other partici-
29 pants: time, date, place, among other attributes of these. In particular, we observe and
30 store in a knowledge base of observations (Figure 2.a), the arguments that the user
31 generates in accordance with his/her argumentative style, and the information that the
32 user uses to support the argument.
33
34
35
36
37
38
39
40
41

42 Once the observations have been stored, we extract the rules for argument genera-
43 tion, which the user uses in the discussions, using an algorithm for generalised asso-
44 ciation rules mining (Figure 2.b). These rules are filtered and then form the argumen-
45 tative model (Figure 2.c).
46
47
48
49

50 Finally, we have to determine how we use the argumentative model. In Figure 2,
51 the possible applications of the model are indicated by dotted arrows. We can use the
52 model to suggest arguments to the user. In this proposal, suggestions are the argu-
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

ments 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.

Fig. 2. Graphical representation of our proposal.

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 $C = \{c_1, \dots, c_n\}$ 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

1
2
3
4
5
6
7
8
9
10 style. Thus, we can learn which conditions are determined by the user to build the ar-
11 guments. We think these relations can be discovered using an algorithm of association
12 rule mining, which allows us to extract reliable patterns from the observed cases.
13

14
15 As introduced in [10], given a set of transactions, where each transaction is a set of
16 items, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items too.
17 That is, the transactions in the database which contain the items of X will also contain
18 the items of Y . That is assured by computing the support and the confidence of the
19 rule. Support and confidence are the main measures in association rule mining algo-
20 rithm. The support of a rule $X \Rightarrow Y$ is the ratio (in percent) of the transactions (T) that
21 contain $X \cup Y$ to the total number of transaction in the database ($|D|$):
22
23
24
25
26
27
28
29

$$30 \text{ Support } (X \Rightarrow Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|D|}$$

31
32
33
34
35
36 The confidence is the ratio (in percent) of the number of records that contain $X \cup Y$
37 to the number of records that contain X .
38
39
40
41

$$42 \text{ Confidence } (X \Rightarrow Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|}$$

43
44
45
46
47 In our work, each transaction represents an observation (C, a) , and we want to dis-
48 cover association rules of the type $X_C \Rightarrow Y_a$, where X_C is composed of items c_i (i.e. a
49 set of conditions), and Y_a is composed of an item a_j , (i.e. only one argument). In this
50 way, our association rules will take the format $\{c_i, \dots, c_p\} \Rightarrow a_j$.
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 However, that is not enough to extract the rules with which the user generate ar-
11 guments, due to the fact that observations are expressed in constant terms. That is, the
12 terms that compose such observations are constants that depict the context where the
13 argument was generated.
14

15
16
17 For example, a condition, expressed informally, c_i : “*jack wants (has as goal) to*
18 *discuss the topic salary*”, which is present in an observation o_i , is expressed in con-
19 stant terms because it concretely represents *Jack’s* goal to discuss the topic *salary*.
20 Nevertheless, we want that the conditions in the rules for argument generation are ex-
21 pressed in variable terms in such a way that we can instance them in any future dis-
22 cussion. In the example, the same condition expressed in variable terms could be “*U*
23 *wants (has as goal) to discuss the topic T*” where U y T are variables that represent
24 any user and any topic respectively, or to add more generality: “*U wants (has as goal)*
25 *G*”, where G is any goal of U .
26
27
28
29
30
31
32
33
34

35 In order to circumvent this problem, we employ an algorithm of generalised asso-
36 ciation rules. These algorithms use the existence of a hierarchical taxonomy of the
37 data to generate different association rules at different levels in the taxonomy [4]. A
38 generalised association rule $X \Rightarrow Y$ is defined identically to that of regular association
39 rules, except that no item in Y can be an ancestor of any in X . An ancestor of an item
40 is one which is above it in some taxonomy. In this sense, we build a hierarchical tax-
41 onomy of conditions and arguments, in which its leaves are the conditions and argu-
42 ments used by the user, and the upper levels are the same propositions but more gen-
43 eral and expressed in variable terms. Then, the generalised association rules algorithm
44 will especially be able to generate rules at upper levels in the taxonomy of conditions
45 and argument. Hence, the rules will be variables.
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

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.

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10 – *prefer*(*user*(*U*), *goal*(*G1*), *goal*(*G2*)): user *U* prefers to fulfil the goal *G1* instead of
- 11 fulfilling the goal *G2*.
- 12
- 13 – *proposal*(*P*): *P* is a proposal uttered in the computational environment.
- 14
- 15 – *accept*(*user*(*U*), *proposal*(*P*)): user *U* accepts the proposal *P*.
- 16
- 17 – *refuse*(*user*(*U*), *proposal*(*P*)): user *U* refuses the proposal *P*.
- 18
- 19 – *imply*(*Q*, *R*): *Q* implies *R*, it represents the classical inference.
- 20
- 21 – *action*(*A*): *A* is an action that can be executed in the environment.
- 22
- 23 – *can_do*(*user*(*U*), *action*(*A*)): user *U* is able to perform the action *A*.
- 24
- 25 – *promised*(*user*(*H*), *user*(*U*), *proposal*(*P*)): user *H* has promised to fulfil the pro-
- 26 posal *P* to user *U*.
- 27
- 28 – *accepted*(*user*(*U*), *proposal*(*P*)): in the past, user *U* accepted the proposal *P*.
- 29
- 30 – *argument*(*user*(*H*), *user*(*U*), *accept*()|*refuse*(), [*J*]): it represents an argument ut-
- 31 tered by the user *H* to the user *U*. The goal of the argument is to support the accep-
- 32 tance or refusal of a proposal with the list of justifications *J*.
- 33
- 34
- 35
- 36
- 37
- 38

39 Moreover, other propositions strongly related to the domain exist, especially those re-

40 lated to goals, proposals and actions the user can execute. For instance, in the domain

41 of meeting scheduling, the extra propositions are:

42

43

- 44 – *discuss_topic*(*T*): *T* is a topic that can be discussed in the meeting.
- 45
- 46 – *in_place*(*P*): the meeting can take place in *P*.
- 47
- 48 – *date*(*D*): the meeting can be in date *D*.
- 49
- 50 – *time*(*M*): the meeting can be at time *T*.
- 51
- 52
- 53
- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65

1
2
3
4
5
6
7
8
9
10 An example of an observation $o = (C, a)$ expressed in L could be:

- 11
12 – $C: \{c_1, c_2\}$
13
14 – $c_1: \text{has_goal}(\text{user}(\text{user_2}), \text{goal}(\text{not}(\text{discuss_topic}(\text{topic_2}))))).$
15
16 – $c_2: \text{promised}(\text{user}(\text{user_2}), \text{user}(\text{user_1}), \text{discuss_topic}(\text{topic_2})).$
17
18 – $a: \text{argument}(\text{user}(\text{user_1}), \text{user}(\text{user_2}), \text{accept}(\text{user}(\text{user_2}), \text{pro-}$
19
20 $\text{posal}(\text{discuss_topic}(\text{topic_2})), [\text{promised}(\text{user}(\text{user_2}), \text{user}(\text{user_1}), \text{dis-}$
21
22 $\text{cuss_topic}(\text{topic_2}))])$.
23
24

25 The example expresses that $user_1$ requests $user_2$ to accept the discussion of the
26 topic $topic_2$, because $user_2$ promised it, but she does not wish to discuss it at that
27 moment.
28
29

30
31 **Fig. 3.** Algorithm to build the taxonomy of propositions from the knowledge base of observa-
32 tions O .
33
34

35 36 4.1.2. Algorithm to build the taxonomy 37

38 Once we have defined the language L to build the taxonomy, we start putting in the
39 leaves the conditions and arguments present in the observations just as they were gen-
40 erated by the user (i.e. c_1, c_2 , and a), one condition or argument for each leaf (Figure 3
41 shows the algorithm to build the taxonomy from the observations stored in O). That
42 is, for each item (condition or argument) of each observation, we build a branch of the
43 taxonomy by starting in this item (leaf) and ending in the root of the taxonomy (See
44 Figure 4).
45
46
47
48
49
50

51 To build this branch, we take an item and generate all the ancestors that represent
52 the same condition or argument but replacing each terminal term (proposition of L
53
54
55
56
57

1
2
3
4
5
6
7
8
9
10 that has not another proposition as parameter that can be generalised) by the respec-
11 tive most general one (step 15 in Figure 3). To determine this, we maintain a data
12 structure (hash table) *HT* with propositions and its most general form. For example,
13
14 for the proposition *has_goal* the most general form stored in *HT* will be
15 *has_goal(user(U), goal(G))*; for *user*, *user(U)*; for *goal*, *goal(G)*; for *discuss_topic*,
16 *discuss_topic(T)*, among others.
17
18
19
20

21 So, given the condition *has_goal(user(user_1), goal(discuss_topic(topic_1)))*, we
22 add a leaf with this and create the following ancestors, taking into account that their
23 terminal terms are *user(user_1)* and *discuss_topic(topic_1)*:
24
25

- 26 – *anc₁*: *has_goal(user(U), goal(discuss_topic(topic_1)))* by replacing the proposition
27 *user(user_1)* with *user(U)*, where *user(U)* is the most general form of
28 *user(user_1)*.
29
30
31
32
- 33 – *anc₂*: *has_goal(user(user_1), goal(discuss_topic(T)))* by replacing the proposition
34 *discuss_topic(topic_1)* with *discuss_topic(T)*, where *discuss_topic(T)* is the most
35 general form of *discuss_topic(topic_1)*.
36
37
38

39 **Fig. 4.** Part of the taxonomy of propositions.
40
41

42 Next, we successively perform the same action, but with each ancestor (steps 17-20),
43 and create a new node in the taxonomy that represents the item, whose parent are the
44 ancestors generated previously (step 21). Following the example, the new ancestor of
45 *anc₁* is *has_goal(user(U), goal(discuss_topic(T)))* (the same for *anc₂*); and finally, we
46 replace *goal(discuss_topic(T))* with *goal(G)* and obtain the most general expression of
47 the initial condition. When the most general expression is found (step 11), a new node
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 is created in the taxonomy whose parent is the root (step 12). In Figure 4, we can ob-
11 serve an example of this part of the taxonomy.
12
13

14 15 *4.2. Execution of generalised association rules algorithm*

16
17 Having just built the taxonomy, we must generate the transactions over which the
18 generalised association rules algorithm will work. As introduced previously, the ob-
19 servations are tuples with the format: (C, a) where $C = \{c^1, \dots, c^n\}$. So, for all $o_i =$
20 $(\{c_i^1, c_i^2, \dots, c_i^n\}, a_i)$ belonging to O , we create a transaction t_i that includes the items
21 $c_i^1, c_i^2, \dots, c_i^n, a_i$. After obtaining the transaction t_i , we replace each item in it by the
22 nearest ancestor that has no constants. Therefore, we eliminate any possible rule with
23 constants² since we are interested in those rules that can be instantiated.
24
25
26
27
28
29

30 The set of transactions that will be obtained after this pre-processing is the input of
31 the generalised association rule algorithm. It is worth noticing that our proposal is in-
32 dependent of the algorithm chosen. That is, we can build argumentative models with
33 any algorithm.
34
35
36
37

38 To obtain generalised association rules, we must generate association rules for all
39 the levels in the taxonomy. One approach to do this would be to take each transaction
40 and expand each item to include all items above it in the hierarchy [4], that is, to add
41 all the ancestors of each item in a transaction t_i to t_i . As it would be expected, when
42 rules are generated for items at a higher level in the taxonomy, both the support and
43 confidence increase. That is a desirable aspect due to the fact that the algorithm of as-
44 sociation rules seeks rules with values of support and confidence higher than the
45 minimum ones.
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 Although, this basic approach is quite expensive and other more efficient algo-
11 rithms have been proposed, it is very simple. For this reason, and taking into account
12 that our proposal is independent of the association rules algorithm, we used it in our
13 experiments. Other algorithm that could be used is Cumulate, which uses several op-
14 timization strategies to reduce the number of ancestors that need to be added to each
15 transaction [4]. Another approach, Stratification, counts itemsets by their levels in the
16 taxonomy and uses the relationships about items in a taxonomy to reduce the number
17 of items to be counted [4]. Several parallel algorithms to generate generalised associa-
18 tion rules have also been proposed [11]. For more detail of implementation and com-
19 parison of performance of generalised association rules algorithm, we recommend to
20 see [4, 12].
21
22
23
24
25
26
27
28
29
30
31

32 4.3. Post-processing generalised association rules

33 The post-processing of the generalised association rules can be divided in two parts.
34 First, we filter out the rules whose format is not adjusted to $\{c_b \dots, c_p\} \Rightarrow a_j$. That is,
35 once all rules have been obtained, we just select the rules whose antecedent is only
36 composed of conditions and whose consequent is a single argument, and the remain-
37 der are filtered out. Since the association rule algorithm processes all items of a trans-
38 action alike, it does not exist a semantic difference between conditions and argu-
39 ments. So, it is possible to find rules like $condition_i \Rightarrow condition_j$ or $argument_m \Rightarrow$
40 $condition_n$, which fulfil the minimum levels of support and confidence, but are irrele-
41 vant to build the user argumentative model. For instance, an irrelevant rule could be
42 $has_goal(user(U1), goal(G1)), has_goal(user(U1), goal(G2)) \Rightarrow prefer(user(U1),$
43 $goal(G1), goal(G2))$. This rule is inappropriate because its three items are conditions.
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 The second part consists in determining how representative the rules are with re-
11 spect to the argument generated by the user and gathered in the observations O . To
12 perform this task, we define a sufficiency metric of an association rule. This metric
13 represents the relation between the conditions of the transactions (observations) that
14 support the rule and the conditions of this rule. It is calculated as the ratio between the
15 total number of the conditions of a rule over the average of the conditions of the
16 transactions that support it. It is defined as:
17
18
19
20
21
22
23
24

$$25 \text{Sufficiency}(r) = \frac{\text{totalConditions}(r)}{\text{averageConditions}(\text{transactionSupporting}(r))}$$

26
27
28
29

30 For example, if we have the transactions $t1 = (c1, c3, c5, a1)$, $t2 = (c1, c2, c4, a1)$, $t3$
31 $= (c1, c4, a1)$, and $t4 = (c1, c5, a2)$; and we define a minimum support of 0.5 and a
32 minimum confidence of 0.75, we will obtain, after the first post-processing step, the
33 rules $r_1: c1 \Rightarrow a1$, $r_2: c4 \Rightarrow a1$ y $r_3: c1, c4 \Rightarrow a1$. Three rules have minimum support and
34 confidence. However, we can see that the rules r_1 and r_2 are not sufficiently represen-
35 tative with regard to the transactions t_1 , t_2 and t_3 , because it is improbable that a single
36 condition will be sufficient to generate the argument a_1 , due to the fact that the condi-
37 tions are not isolated in the transactions. The sufficiency metric aims to filter these
38 rules setting a threshold that determines how sufficient the conditions (antecedent) of
39 a rule must be to generate the consequent argument, independently of the values of
40 support and confidence.
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 **Table 1.** Metrical comparison of rules.

11
12 Table 1 details the values of the three metrics. We can observe that the rules r_1 and r_2
13 have a sufficiency value comparatively low with regard to the rule r_3 . Therefore, a
14 threshold of 75% only allows rule r_3 to be valid for the user argumentative model.
15
16 The value of this metric can also be used by a personal agent at the moment of assist-
17 ing the user, preferring to suggest arguments generated by rules with higher value of
18 sufficiency.
19
20
21
22
23

24 **Table 2.** Information simulated for the participant *Jack*..

25 26 27 28 **5. Experimental Results**

29
30
31 The domain we chose to test our proposal was an application for meeting scheduling.
32 In this application, the users can arrange meetings discussing date, time, place, topics
33 to discuss during the meeting, and participants. Due to the fact that users have differ-
34 ent goals, they must exchange arguments in order to reach an agreement.
35
36
37

38 We worked with a group of 25 users who had to generate arguments to support or
39 defeat a set of proposals. We simulated information about the context of the discus-
40 sion and we requested them to generate the greater amount of possible arguments us-
41 ing common sense. It is worth noticing that the users had no previous knowledge
42 about argumentation.
43
44
45
46
47

48 The context of the discussion was composed of four participants³ (*Jack, Kate, Ben*
49 and *Juliet*), and sets of topics to discuss (*salary, vacations, overtime payments* and
50 *new employees*), places where participants could meet (*room 1, room 2, lab, buffet,*
51 *coffee-shop* and *restaurant*), periods of the day (*morning, afternoon, evening* and
52
53
54
55
56
57

1
2
3
4
5
6
7
8
9
10 *night*), and days of the week. Moreover, for each participant we defined a set of goals,
11 beliefs, preferences, topics that he/she could discuss, and historical information
12 (promises uttered and proposals accepted in the past). Table 2 shows the information
13 simulated for the participant *Jack*.
14
15

16
17 This information was presented to the users in the application for meeting schedul-
18 ing, as if they should schedule a meeting. They were requested to generate arguments
19 on four situations, one for each simulated participant, to support the proposals derived
20 from the goals of each one, or to persuade another participant to resign theirs. For ex-
21 ample, in the situation of participant *Jack*, some possible proposals to be supported by
22 arguments are: *accept(user(kate), proposal(discuss _topic(overtime-payments)))*, *re-*
23 *fuse(user(ben),proposal(time(night)))*, or *accept(user(juliet), place(coffee-shop))*, and
24 so on with the others participants.
25
26
27
28
29
30
31
32

33 As a result, we gathered 1,234 observations, with the format detailed in the previ-
34 ous section, divided in a knowledge base for each user. All observations were ex-
35 pressed in the language *L*. We divided the observations of each user in two parts. The
36 first part was composed of arguments belonging to the three first situations, we called
37 it *training observations*, and the arguments of the fourth situation was put on the sec-
38 ond part, which we called *test observations*. Then, we performed the process to build
39 the user argumentative models using the training observations. First, we built a taxon-
40 omy of facts. Second, for each knowledge base of observations, we converted its con-
41 tent to a set of transactions, and executed the generalised association rule mining al-
42 gorithm with the taxonomy and these transactions as input. Thus, we obtained a set of
43 association rules. Then, we post-processed this set and built the argumentative model
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 for each user. In Table 3, we show information about observations per user, rules per
11 argumentative model, and a ratio between both parameters.
12
13

14 **Table 3.** Observations and rules obtained by user, and results of the evaluations.
15

16
17 As we can see in the Table 3, there is not a relation between the amount of observa-
18 tions and the amount of rules in the user argumentative model. User 6 and 19 gener-
19 ated a lot of arguments, but the algorithm of association rules could not find relevant
20 rules. We think this can indicate that the users did not have a well-defined argumenta-
21 tive style. That is, the user generated arguments but not following a specific pattern.
22 This fact is denoted by a nil or lower ratio R/O (the number of rules divided the num-
23 ber of observations). On the other side, when the ratio increases, the argumentative
24 style is well defined and the argumentative model is composed of several rules.
25
26
27
28
29
30
31

32 **Fig. 5.** Rules for argument generation extracted for user 1.
33

34
35 To illustrate the results obtained, we show the rules for argument generation learnt
36 from the user 1 (see Figure 5). From this user, we obtained 5 rules. In rule 5.a, the an-
37 tecedent, or condition for the argument generation, is *has_goal(user(U1),*
38 *goal(**discuss_topic(T1))*); and the consequent (the argument) is *argu-*
39 *ment(user(U2),user(U1), accept(user(U1), proposal(discuss_topic(T1))),*
40 *[has_goal(user(U1), goal(discuss_topic(T1)))]*). That means the condition the user 1
41 (in this case *U2* because he is the sender of the argument) finds to express the argu-
42 ment, to support the proposal *discuss_topic(T1)*, is the user *U1* has the same goal.
43 Moreover, we can observe some similar rules, but with different level of detail, such
44 as rules 5.d and 5.e. Although these rules are redundant since 5.d is included in 5.e,
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 are valid for the argument generation. Note that this difference of generality is ob-
11 tained by the generalised association rule algorithm.

12
13
14 It is also worth noticing that we cannot be sure that this set of rules represents
15 completely the argumentative style of the user, but we claim that it is a good result
16 since the number of observations for this user was only 25.
17
18

19
20 **Fig. 6.** Rules for argument generation.
21

22
23 To evaluate that the rules of the argumentative models depict the argumentative styles
24 of the users, we compared for each user the arguments that we can obtain using
25 his/her argumentative model, versus the arguments stored on the test observations.
26 The main idea was to validate the rules that compose the argumentative models by de-
27 termining if the arguments generated by the user in a new situation (test observations)
28 could be generated by means of the rules for argument generation learnt from the
29 training observations. For each test argument (argument stored in test observations),
30 we established one of the three values: NO (it cannot be generated from the argumen-
31 tative model); PARTIAL (it can be partially generated, that is, the test argument was
32 generated with more conclusions or premises that those present in the rule), and YES
33 (it can be completely generated). For example, given the rules of user 1 (Figure 5), the
34 values for the following test arguments were:
35
36
37
38
39
40
41
42
43
44

- 45
46 – $has_goal(user(juliet), goal(not(place(buffet)))) \Rightarrow argument(user(juliet), us-$
47 $er(ben), refuse(user(ben), proposal(place(buffet))))$, $[has_goal(user(juliet),$
48 $goal(not(place(buffet))))]$: this argument cannot be generated, since no rule of the
49 argumentative model matches this pattern.
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

- 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
- *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))))*, [*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 5.d.
 - *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 5.b.

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 Section 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

1
2
3
4
5
6
7
8
9
10 finding the same rules that others works have specified from others perspectives (i.e.
11 psychological theories), shows signs that the proposed mechanism to learn rules for
12 argument generation works well. That is, the studies in the area of argumentation-
13 based negotiation help us to validate our mechanism for building user argumentative
14 models. For this reason, we show in Figure 6 some rules discovered using the process
15 specified above, which were defined in [5] too. In our work, these rules were found
16 for several users. We analyse them below:
17
18

- 19 – Trivial appeal (Figure 5.a): it is a simple appeal: $U2$ knows that the place PI is oc-
20 cupied (that means $not(place(PI))$), and he/she uses this information to justify the
21 refusal of PI as the place for the meeting (this rule was found in 12 users).
22
- 23 – Appeal to prevailing practice (Figure 5.b): this is an appeal that resorts to historical
24 information ($accepted(user(U), proposal(P))$) to persuade an opponent to accept a
25 proposal. That is, “if you accepted the proposal PI in the past, now you should ac-
26 cept it too”. In this rule, the ability of abstraction that the generalised association
27 rules give to our approach can be assessed since all arguments present in the obser-
28 vation have the proposal instantiated (rule found in 16 users).
29
- 30 – Threat (Figure 5.c): we suppose this rule represents an incomplete one for threat
31 generation. To be completed, this rule should have a proposition that represents
32 $U2$'s intention to refuse the discussion of topic TI in the justification of the argu-
33 ment. In other words, if $U1$ does not accept the date DI , $U2$ do not accept to dis-
34 cuss the topic TI . Finding incomplete rules can aid to improve the argumentative
35 abilities of the user (rule found in 2 users).
36
- 37 – Appeal to past promise (Figure 5.d): it is also a simple appeal, in which $U2$ re-
38 minds $U1$ that he/she promised to discuss topic TI (rule found in 18 users).
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10
11
12 Note that some rules appear more frequently than others. That happens because some
13 rules are simpler than others. Simple rules, such as appeals to prevailing practice, are
14 easily discovered with few observations. Intuitively, this is because the number of
15 conditions is low, so it is expected that smaller patterns on the conditions are more
16 frequent than patterns with a high number of conditions. On the contrary, some rules,
17 such as threats, are not as common (they were only discovered from 2 users) as oth-
18 ers, because it is structurally more complex.
19
20
21
22
23
24
25
26

27 **6. Applications of the user argumentative model**

28
29
30 As we said in the introduction, there are several applications for a user argumentative
31 model. In this section, we will give an overview about two possible applications: the
32 personalised suggestion of arguments and the discovery and analysis of users' argu-
33 mentative skills.
34
35
36
37
38

39 **6.1 Suggesting personalised arguments**

40
41 As we introduced above, when a user participates in a discussion he/she must gener-
42 ate arguments to persuade his/her opponents. To achieve this, he/she builds these ar-
43 guments according to his/her argumentative style. So, if we have a user argumentative
44 model that depicts that style, we could automatically generate arguments in a person-
45 alised way.
46
47
48
49

50 We assume that we can access to the contextual information of the discussion in
51 which the user is participating, and that this information is expressed in the language
52 *L* described before. For example, a personal agent [14], which observes the computa-
53
54
55
56
57

1
2
3
4
5
6
7
8
9
10 tional environment, can detect the intentions [15, 16] of the user or can observe both
11 the proposal that he/she utters and the additional information of the discussion. Once
12 the intentions or the proposals are detected and the information is gathered, the per-
13 sonal agent can use the rules stored in the user argumentative model to generate ar-
14 guments that give support to the intention and proposals of the user. The argument
15 generation is performed by checking the rules, whose argument matches the proposal
16 that we want to support, considering the contextual information of discussion. Then if
17 the conditions of the rule are met, we will be allowed to generate the argument.
18
19
20
21
22
23
24

25 For example, given two users John and Peter, if the proposal of John is that Peter
26 accepts to discuss about salary (*accept(user(peter), proposal(discuss_topic(salary)))*),
27 the rules, whose condition we have to check, must generate an argument with the
28 format: *argument(user(john), user(peter), accept(user(peter), pro-*
29 *posal(discuss_topic(salary))), [JJ]*, where *J* must be instantiated with any justifica-
30 tion.
31
32
33
34
35
36

37 Once the arguments have been generated, they can be presented to the user in a
38 graphical interface so that the user can choose one, modifying or not some of its parts,
39 or reject the suggestions. This can be useful as a feedback for the correctness of the
40 rules in the user argumentative model too. The order in which the arguments are pre-
41 sented to the user can be determined by the values of the metrics of the rule that gen-
42 erated it.
43
44
45
46
47

48 To exemplify this application, we will show how arguments can be generated using
49 the rules included in the argumentative model of user 1 (Figure 5), namely John. No-
50 tice that the argumentative style of John is modelled by the rules included in the ar-
51 gumentative model. Given a discussion between John and Peter, we suppose that both
52
53
54
55
56
57

1
2
3
4
5
6
7
8
9
10 want to arrange a meeting to negotiate a contract. John wants to meet on Monday
11 morning in his office. This information expressed in the language L is:
12
13 *has_goal(user(john), goal(discuss_topic(contract))), has_goal(user(john),*
14 *goal(date(monday))), has_goal(user(john), goal(time(morning))),*
15 *has_goal(user(john), goal(place(my_office)))*. Moreover, we can access to both the
16
17 next information about Peter and the context of the discussion: *believe(user(john),*
18 *not(place(meeting_room)))* (the meeting room is occupied), *accepted(user(peter),*
19 *proposal(and(date(monday), time(morning))))* (in the past, Peter accepted a morning
20
21 meeting on Monday), *has_goal(user(peter), goal(not(date(friday))))* (Peter has the
22
23 goal of not meeting on Friday), among others.
24
25
26
27
28

29 In this scenario, we can see several situations to generate arguments from the per-
30
31 spective of John. For example:

32
33 – Peter proposed to meet in the meeting room.

34
35 As John wants to meet in his office, he has to generate an argument that refuses such
36
37 proposal. So, we can suggest a trivial appeal with rule c (Figure 5.c), saying that the
38
39 meeting room is occupied. The argument suggested will be: *argument(user(john),*
40 *user(peter), refuse(user(peter), proposal(place(meeting_room))))*, [*be-*
41 *lieve(user(john), not(place(meeting_room))))*].
42
43

44 – John proposed a morning meeting on Monday and Peter refused it, justifying that
45
46 he cannot meet on Monday morning.

47
48 In this situation, John should give Peter a justification in order to persuade him to
49
50 meet that day at that time. This justification can be reached with an argument gener-
51
52 ated by rule e (Figure 5.e). The argument suggested will be: *argument(user(john),*
53 *user(peter), accept(user(peter), proposal(and(date(monday), time(morning))))*), [*ac-*
54
55

1
2
3
4
5
6
7
8
9
10 *cepted(user(peter), proposal(and(date(monday), time(morning))))]*, or *argu-*
11 *ment(user(john), user(peter), accept(user(john), proposal(date(monday))),*
12 *[has_goal(user(peter), goal(not(date(friday))))]*), which is not a complete argument,
13
14
15 but could be an option.
16
17
18

19 **6.2 Discovering and analysing users' argumentative skills**

20
21
22

23 As we introduced earlier, modelling the argumentative style of a user can be interest-
24 ing in the area of knowledge management. We keep on focus three perspectives in
25 which user argumentative models have possible applications. We will briefly com-
26 ment these applications. We will briefly com-
27 ment these applications:
28
29

- 30 – Discerning faults in the users' argumentative abilities: as we showed in Figure 5.c,
31 we can detect some problems in users' argumentative abilities. Another problem
32 could be seen in the Table 3 with the users 6 and 19, they generate a lot of argu-
33 ments, but apparently without a defined style. Once the problems are detected, we
34 can take decisions to correct them.
35
36
- 37 – Task allocation: if we have to allocate a task among different users, the information
38 about user's skills helps us to take decisions. So, we could analyse the argumen-
39 tative models and determine if a user is able to perform a task that demands argu-
40 mentation (i.e. to negotiate with a provider). That is, for example, if the user has
41 not faults in his/her user's argumentative abilities, he/she will be able to perform
42 the task properly.
43
44
- 45 – Organizational memory: as we said before, it could be useful to include in the or-
46 ganizational memory the user argumentative models in order to have this informa-
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 tion for future decisions, or to train the argumentative skills of future employees in
11 an organization.
12
13
14

15 **7. Related Work**

16
17
18 It is hard to set this work in the context of the actual literature. First, we have not
19 found works related to the construction of user argumentative models, and second,
20 our proposal is interdisciplinary in its constitution as well as in its application.
21
22

23
24 On the one hand, the rules for argument generation that are learnt in our proposal
25 and form the user argumentative model, have been used in the area of argumentation
26 based negotiation among intelligent agents [13]. Kraus et al. [5] determine a set of
27 possible argument types, and for each argument type they define preconditions for its
28 usage. Only if the preconditions are fulfilled, the argument will be used. In the work
29 of Ramchurn et al. [17] and Sierra et al. [9], the authors define preconditions for ar-
30 gument generation too, each one taking into account several factors (authority, utility
31 of the proposals, etc.). However, these works define these rules statically, and do not
32 establish a mechanism to learn them. With regard to the different factors that could be
33 taken into account in the conditions of the rules, our proposal allows us to consider all
34 that, storing the information about the context as a condition in the observations.
35
36
37
38
39
40
41
42
43
44

45 With respect to user models, they have been applied in the field of argument inter-
46 pretation. Zukerman et al. [18] incorporate a user model in the process of argument
47 evaluation, that is arguments that a user receives from other ones; but this work ex-
48 cludes the argument generation. The user model is represented by a Bayesian net-
49 work. This network represents the users' beliefs, which are born in mind by an argu-
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 ment-interpretation mechanism. In addition to the differences between the user model
11 representations (we use associations rules and they use Bayesian networks), both
12 works are focused on different mechanisms of the argumentation process. Zukerman
13 et al.'s work focuses on argument evaluation and we focus on argument generation.
14
15
16
17
18

19 **8. Conclusions and Future Work**

21
22 This paper contributes to the areas of user modelling and argumentation based nego-
23 tiation. We have been presented a original mechanism to build user argumentative
24 models that captures the argumentative styles of the users, which have not been mod-
25 elled to date. For this task, we observe how the users generate his/her argument dur-
26 ing a discussion in a computational medium, and on the basis of these observations,
27 we apply a generalised association rules algorithm to extract rules for argument gen-
28 eration. The rules obtained are composed of an antecedent and a consequent. The an-
29 tecedent is a set of conditions that the context of the discussion has to fulfil in order to
30 be able to generate an argument, which form the consequent of the rule. We defined
31 some filters to select the most interesting rules. First we select the rules whose format
32 is interesting for our goals, and then we calculate a metric to determine if the condi-
33 tions of the rule are sufficient to generate the argument.
34
35
36
37
38
39
40
41
42
43
44

45 The evaluation of this proposal was carried out in the scenario of meeting schedul-
46 ing. We worked with a group of 25 users who have to arrange meetings in a distrib-
47 uted application. We observed how users generated their arguments, we learnt the
48 rules for argument generation that they use tacitly, and built an argumentative model
49 for each of them. We learnt several rules for each user though the number of observa-
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10 tions was limited, with the exception of some users with a high number of observa-
11 tion, but with a low number of rules discovered.

12
13 To validate this result, we found that a 42.95% of the arguments generated by users
14 in a test situation, can be completely generated using the user argumentative models
15 and a 16.72% can be partially generated. This indicates that the rules in the argumen-
16 tative model represent the argumentative style of the user, at least partially.
17
18
19
20

21 In addition, we compared the rules obtained with other works in the area of argu-
22 mentation based negotiation. We found several coincidences between the rules ob-
23 tained by our proposal and the rules defined explicitly in the work of Krauss et al. [5].
24 Moreover, we showed for several situations how arguments can be generated from the
25 user argumentative model, and gave an overview about the applicative scope of our
26 work.
27
28
29
30
31
32

33 There are a number of future research directions, such as determining how to ana-
34 lyse the user argumentative model to detect flaws in the argumentative skill of the us-
35 ers, and which corrective actions take to solve this. In addition, we can apply the ar-
36 gumentative model to evaluate the arguments that the user receives. Most importantly,
37 future research will assess how the users respond to the suggestion of a personal
38 agent, and how this interaction between user and agent could be taken into account in
39 the suggestion of arguments.
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10
11 **Footnotes**
12
13

- 14 1. The term “argument” is not referred to a discussion, often heated, in which a dif-
15 ference of opinion is expressed, but to a fact or circumstance that gives logical
16 support to an assertion, claim, or proposal.
17
18 2. Constants are the parameters whose names start with a lower case character. For
19 example, *user_I* is a constant instead of *User_I*, which is a variable.
20
21 3. The term “participant” refers to a simulated person in the context of the discus-
22 sion and “user” refers to the person who participates in the experiments and
23 whose argumentative model is built. That is, “users” generate arguments for the
24 situations of each “participant”.
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10
11
12 **References**
13
14

- 15 1. D.R Ilgen and C.L. Hulin, "Computational Modeling of Behavior Organizations: The Third
16 Scientific Discipline", American Psychological Association: Washington DC, chapter 1,
17 2000.
- 18 2. B. Hedberg, "How Organizations Learn and Unlearn", in Handbook of Organizational De-
19 sign, edited by P.C. Nystrom and WH. Starbuck, Oxford University Press: New York, pp.3-
20 27, 1981.
- 21 3. E. Stein and V. Zwass, "Actualizing Organizational Memory with Information Systems", In-
22 formation Systems Research, vol.6(2), pp.85-117, 1995.
- 23 4. R. Srikant and R. Agrawal, "Mining Generalized Association Rules," Future Generation
24 Computer Systems, vol.13(2-3), pp.161-180, 1997.
- 25 5. S. Kraus, K. Sycara and A. Evenchik, "Reaching Agreements through Argumentation: a
26 Logical Model and Implementation", Artificial. Intelligence, vol.104(1-2), pp.1-69, 1998.
- 27 6. P. Brusilovsky and E. Millán, "User Models for Adaptive Hypermedia and Adaptive Educa-
28 tional Systems", in The Adaptive Web, Springer-Verlag: Berlin, pp. 3-53, 2007.
- 29 7. M. Karlins and H.I. Abelson, Persuasion: How Opinions and Attitudes are Changed,
30 Springer: Berlin, 1970.
- 31 8. D. O'Keefe, Persuasion: Theory and Research, SAGE Publications, London, 1990.
- 32 9. C. Sierra, N.R. Jennings, P. Noriega and S. Parsons, "A Framework for Argumentation-
33 based Negotiation", in Proc. 4th International Workshop on Agent Theories, Architectures
34 and Languages, Rode Island, USA, 1998, pp.177-192.
- 35 10. R. Agrawal, T. Imieliński, and A. Swami, "Mining Association Rules between Sets of Items
36 in Large Databases," in Proc. of the 1993 ACM SIGMOD international Conference on
37 Management of Data, Washington, D.C., United States, 1993, pp.207-216.

11. T. Shintani and M. Kitsuregawa, "Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy", in Proc. ACM SIGMOD International Conference on Management of Data, SIGMOD 1998, Seattle, Washington, USA, pp 25-36, 1998.
12. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in Proc. 20th Int. Conf. Very Large Data Bases, Santiago de Chile, Chile, 1994, pp.487-499.
13. I. Rahwan, S.D. Ramchurn, N.R. Jennings, P. McBurney, S. Parsons and L. Sonenberg, "Argumentation-based Negotiation," The Knowledge Engineering Review, vol.18(4), pp.343-375, 2003.
14. P. Maes, "Agents That Reduce Work and Information Overload", Communications of the ACM, vol.37(7), pp.30-40, 1994.
15. H. Kautz, "A Formal Theory of Plan Recognition", PhD thesis, Dept. of Computer Science, University of Rochester, 1987.
16. E. Charniak and R. P. Goldman, "A Bayesian Model of Plan Recognition," Artificial Intelligence, vol.64(1), pp.53-79, 1993.
17. S.D. Ramchurn N.R. Jennings and C. Sierra, "Persuasive Negotiation for Autonomous Agents: a Rhetorical Approach," in Proc. IJCAI Workshop on Computational Models of Natural Argument, 2003, Acapulco, Mexico, pp.9-17.
18. I. Zukerman, S. George and M. George, "Incorporating a User Model into an Information Theoretic Framework for Argument Interpretation," in Proc. Ninth Int. Conf. on User Modeling, Johnstown, Pennsylvania, United States, pp.106-116, 2003.

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

Table 2. Information simulated for the participant Jack

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

Table 3. Observations and rules obtained by user, and results

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. 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

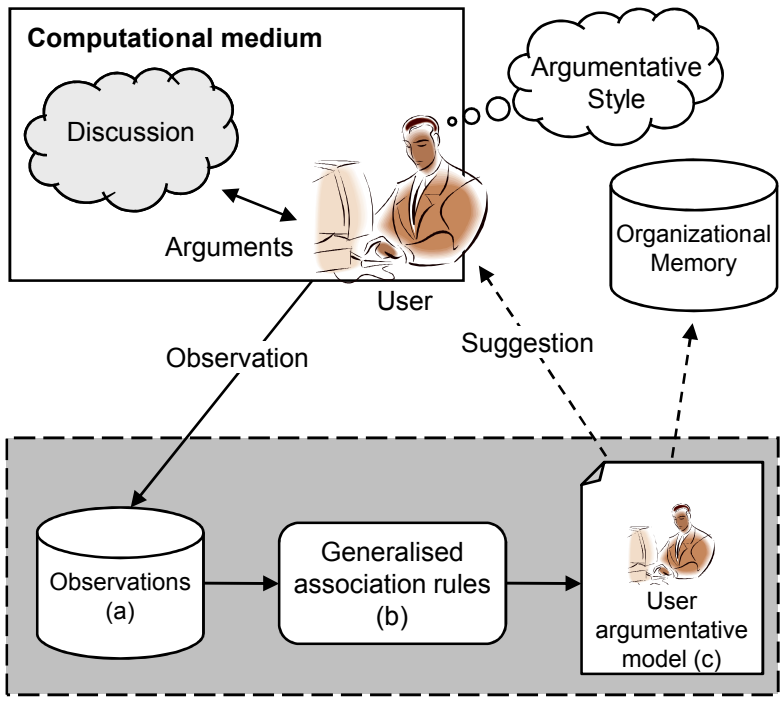


Fig. 3. Algorithm to build the taxonomy

```
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
16.                by the respective more general term.
17.       $P_{fact}$  = {array of parents}
18.      forall ancestors anc ∈  $A_{fact}$  {
19.         $b_{anc}$  = buildBranch(anc, root)
20.         $P_{fact}$  = add  $b_{anc}$ . // add the ancestors as parents of fact
21.      }
22.      return a new node with  $P_{fact}$  as parents and fact as data.
23.    }
24.  else
25.    return the node that represent fact (avoid duplicate).
26.  }
```

Fig. 4. Part of the taxonomy of propositions

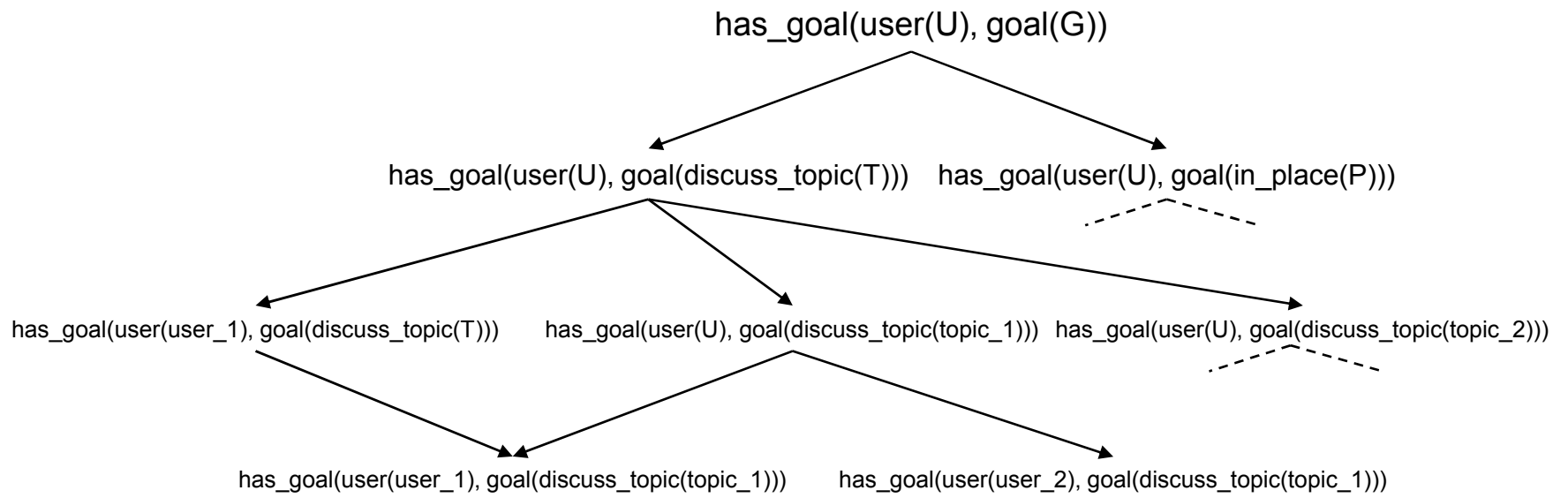


Fig. 5. Rules for argument generation extracted for user 1

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

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