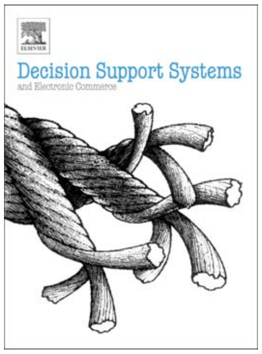


Argumentation-based negotiation planning for autonomous agents

Ariel Monteserin, Analía Amandi

PII: S0167-9236(11)00087-X
DOI: doi: 10.1016/j.dss.2011.02.016
Reference: DECSUP 11876

To appear in: *Decision Support Systems*

Received date: 23 February 2010
Revised date: 9 February 2011
Accepted date: 27 February 2011

Argumentation-based negotiation planning for autonomous agents

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. When we negotiate, the arguments uttered to persuade the opponent are not the result of an isolated analysis, but of an integral view of the problem that we want to agree about. Before the negotiation starts, we have in mind what arguments we can utter, what opponent we can persuade, which negotiation can finish successfully and which cannot. Thus, we plan the negotiation, and in particular, the argumentation. This fact allows us to take decisions in advance and to start the negotiation more confidently. With this in mind, we claim that this planning can be exploited by an autonomous agent. Agents plan the actions that they should execute to achieve their goals. In these plans, some actions are under the agent's control, while some others are not. The latter must be negotiated with other agents. Negotiation is usually carried out during the plan execution. In our opinion, however, negotiation can be considered during the planning stage, as in real life. In this paper, we present a novel approach to integrate argumentation-based negotiation planning into the general planning process of an autonomous agent. This integration allows the agent to take key decisions in advance. We evaluated this proposal in a multiagent scenario by comparing the performance of agents that plan the argumentation and agents that do not. These evaluations demonstrated that performance improves when the argumentation is planned, specially, when the negotiation alternatives increase.

Keywords: argumentation-based negotiation, autonomous agents, multiagent systems, negotiation planning, conflict resolution.

1. Introduction

In multi-agent environments, autonomous agents need to interact with one another to achieve their goals because reciprocal dependencies exist among them. In this context, negotiation is a fundamental tool to reach agreements among agents with conflictive goals in both competitive and collaborative scenarios.

The essence of the negotiation process is the exchange of proposals. Agents make and respond to proposals in order to converge towards a mutually acceptable agreement. However, not all approaches are restricted to that exchange. Several approaches to automated negotiation have been developed. One of them is the argumentation-based approach [23, 42, 31, 33, 2, 18]. In argumentation-based approaches, agents are allowed to exchange some additional information as arguments, besides the information uttered on the proposal [33]. Thus, in the negotiation context, an argument is seen as a piece of information that supports a proposal and allows an agent (a) to justify its position in the negotiation, or (b) to influence other agents' position in the negotiation [20].

An agent following a course of action to fulfil its goals has to negotiate the execution of the actions that are not under its control (*intend-that* actions, [23]). In this context, negotiation becomes a critical action within its course of action. However, an agent traditionally builds a plan that determines a course of action with no regard for the negotiation. Then, it negotiates the execution of the *intend-that* actions and the resources needed. But, what happens if the negotiation fails? The plan is interrupted and the agent must build an alternative one. In this context, a new question arises: is it possible to view the negotiation as an action needed to achieve the agent's goals when it builds the course of action?

When we negotiate, the arguments uttered are not the result of an isolated analysis, but of an integral view of the problem that we want to agree about. In every conflictive situation where negotiation is necessary, in real life as well as in a system composed of multiple agents, the ability to plan the course of action that will be executed to resolve the conflict allows the negotiator (a) to anticipate the problems that he/she could find during the interaction, and also, (b) to analyze anticipated solutions to the conflict in order to avoid or minimize its problematic effects. If the negotiator knows the arguments he/she can utter before the negotiation starts, he/she can take decisions in advance, especially when there are several alternatives to choose from. For example, if a certain resource may be obtained from two different opponents, the agent can choose to negotiate with the opponent that it can persuade.

For these reasons, we claim that the negotiation and, in particular, the argumentation that the agent develops during that process can be planned before the negotiation takes place. Consequently, once the process to plan the argumentation is defined, we can integrate it into the general planning process of the agent as a key action.

Therefore, in the first place, we want to plan the argumentation that an agent carries out during a negotiation. Intuitively, planning can aid in this task. Planning algorithms are able to provide a plan of action that, when executed on the specified initial state, allows the agent to achieve an expected final state [15]. Then, we can model the argumentation-based negotiation process as a planning problem, thus obtaining an argumentation-based negotiation plan, which we call *argumentation plans*. In this kind of plan, the actions represent the arguments that the agent will use during the argumentation process to persuade its opponents and finally reach a profitable agreement. To carry out this modelling, we identify every element of a planning problem (initial and final states and actions) in the argumentation process. Moreover, we emulate the main mechanism of this process (argument generation and argument selection) within the planning algorithm. An important remark is that the arguments generated by the agent should be determined taking into account its mental state. The agent's mental state is composed of beliefs, goals, and preferences about type of arguments, among other information. For example, given the same conflictive situation, two agents will generate different arguments because the information the agents have about the problem and the conflict context are not the same. Nevertheless, the original planning algorithms do not consider this issue in the plan conception. Hence, to generate argumentation plans we propose the use of a planning algorithm based on preferences, in which the agent's mental state impacts on the action selection process and, as a result, on the argument selection.

Finally, once we have modelled the argumentation process as a planning problem, its integration with the agent's general planning is very simple. From the set of general actions that the agent uses to achieve its goals, we distinguish the *intend-to-do* actions and the *intend-that* actions [23]. The first ones are under the direct control of the agent, but the second ones are not. Consequently, the later have to be negotiated. Thus, we embed an argumentation plan for each *intend-that* action that has to be negotiated within the general plan. When an argumentation plan cannot be generated to support an *intend-that* action, the planning algorithm looks for an alternative action, whose argumentation plan indeed can be built in planning time. However, if there is no alternative action for which an argumentation plan may be generated, but the general plan can be built, the planning algorithm uses unconditional *intend-that* actions. An unconditional action indicates that the action must be negotiated in the traditional way.

With this integration, we give the agent the ability to decide, in planning time, what alternative can be negotiated. This includes to decide: (a) what action or (b) what resource to negotiate (if we see the acquisition of a resource as an action), (c) which agent to negotiate with, and (d) in which order the negotiations should be accomplished. In contrast, traditional approaches take these decisions in execution time.

Upon deciding on these points in planning time instead of execution time, the agent improves its performance in several aspects. First, the agent reduces the number of failures during the negotiations since it chooses the actions supported by an argumentation plan in planning time. Consequently, the number of plans and replans (next solution of an initial plan) generated by the planning algorithm as well as the number of negotiations in which the agent takes part are smaller. This has a direct influence on the total time spent and the total messages exchanged by the agent to reach its goals.

We evaluated our proposal in an extension of the negotiation scenario proposed by Parsons and Jennings [28]. In the original scenario, there are two agents that have scarce resources to execute some tasks and reach their goals by exchanging proposals and arguments. We extended it by adding new agents and resources in order to turn the negotiation into a multilateral one. We compared the performance of an agent that builds argumentation plans (*Picty_arg*) and an agent that does not (*Picty_sim*). We found that when the negotiation alternatives increased in the scenario, the *Picty_arg*'s performance was better than the *Picty_sim*'s performance.

The paper is organized in the following way. Section 2 shows the argumentation process as a valid problem to be solved with planning. Section 3 shows how the argumentation problem is represented on the planner, as an initial state, a final state and actions. In Section 4, we show how a planning algorithm based on preferences is applied to build argumentation plans. Section 5 explains the integration of the argumentation plan generation into the general planning process. Section 6 presents the experimental results. Section 7 discusses the application of argumentation plans to support decision-making. Section 8 places this work in the context of previous ones. Finally, in section 9 we state our conclusions and suggest future work.

2. Defining an argumentation process as a planning problem

In accordance with the work of Rahwan et al. [32], there are two major strands in the literature on argumentation-based negotiation: (a) attempts to adapt dialectical logics for defeasible argumentation by embedding negotiation concepts within these [3, 30, 38]; and (b) attempts to extend bargaining-based frameworks by allowing agents to exchange rhetorical arguments, such as promises and threats [23, 33, 5]. Our work is situated in the second strand.

In this section, we explain how the argumentation process[1] may be modelled as a planning problem in order to obtain an *argumentation plan*. We define an *argumentation plan* as a partial order sequence of arguments that allows the agent to reach an expected agreement when it is uttered in a specified conflictive situation. An argumentation plan will determine how the agent must perform the argumentation process during a given negotiation. In other words, it will provide the set of arguments and the order in which they should be presented to the counterparts in order to come to an agreement.

To show how this works, we will describe the main characteristics of an argumentation process and some mechanisms of a planning algorithm (those relevant to our proposal). Also, we will conceptually define a planning problem. Then, we will match every component of the argumentation process with its corresponding one in a planning problem. Figure 1 represents this idea.

Figure 1. Graphical representation of an argumentation process as a planning problem.

First, we start depicting the argumentation-based negotiation scenario. Conflicts can arise in competitive scenarios as well as in collaborative ones [19]. Usually, the difference between these scenarios is that in a competitive scenario the negotiation is *win-lose*, because the conflict results from the conflicting goals of the agents. On the other hand, negotiations in a collaborative scenario are *win-win* because the agents have shared goals, but conflicts can arise from differences on how shared goals can be reached. When an agent detects a conflict, it can access information about this conflict and its context before the negotiation begins. By definition, the conflict that will generate the negotiation is rooted in the conflictive interests that the involved agents have. These conflictive interests are represented, for example, in the mental states of such agents. Thus, the information that the agent can access before starting the negotiation process includes:

– Self-information: agent's mental state, such as beliefs, preferences and goals[2]. This information conditions the agreement that the agent seeks.
– Information about its opponents: information about other agents' beliefs and goals that the agent has gathered in previous interactions. In realistic situations agents only have incomplete information about their opponents, because agents have some private information about their state that is unavailable to the other agents. As a result, this information is tentative and incomplete. However, we assume that the agent can determine the extent to which each fact that makes up this information is believable.
– Information about the conflict context: relevant knowledge about the conflict and its resolution, for example the space of potential agreements [21]; and historic information about past negotiations.

Henceforth, the available information to be used by the agent during the negotiation will be called *negotiation information*.

As mentioned above, in an argumentation-based negotiation approach, agents can exchange arguments in order to justify their proposals, to persuade their opponent, and to reach an expected agreement. In addition to evaluating and generating proposals, agents with the ability for argumentation, must be able to (a) evaluate incoming arguments and update its mental state as a result; (b) generate candidate outgoing arguments; and (c) select an argument from the set of candidate arguments [6]. Thus, we can say that the argumentation process is composed of the evaluation of outgoing arguments and the generation and selection of incoming arguments. In this context, the agent starts the argumentation process, and takes every decision related to this process on the basis of *negotiation information*. In other words, this information is part of the input of the evaluation, generation and selection of arguments. Also, we observe that the agent uses the argumentation process in combination with the proposal evaluation and generation to reach an agreement in order to resolve an initial conflictive situation (See Figure 1.a).

In this work, we focus on incoming argument processes: argument generation and argument selection. These processes may be described as follows:

– *Argument generation* is related to the generation of candidate arguments to present to a counterpart. To this end, rules for arguments creation are defined (e.g. [23, 33]). Such rules specify conditions for argument generation. So, if the condition is satisfied in the negotiation context, the argument may be generated and it becomes a candidate argument.

– *Argument selection* is concerned with selecting the argument that should be uttered to a counterpart from the set of candidate arguments generated by the argument generation process. Once candidate arguments have been generated, then the argument selection mechanism must apply a policy, in accordance with the agent's mental state, to select the best argument. Policies are diverse, they can order all arguments by their severity selecting the weakest first [23]; take into account the counterpart's reputation or trust [33]; or choose the shortest argument in order to reduce the target to counter-argue [40]; among others.

Moreover, a planning algorithm is used to find a plan of action. In our approach, a *plan* is a partial order sequence of actions which will achieve a desired final state when executed in any world satisfying the initial state description [48]. Conceptually, a planning problem is defined as a tree-tuple <*i, f, A*>, where:

– *i* (initial state) is a complete description of the world in which the plan will be executed.
– *f* (final state) describes the agent's goals. In other words, it describes where the agent wants to arrive by executing the plan.
– *A* (actions) is the set of available actions to build a plan. For each action its precondition and effects are defined. Thus, for an action to be added to a plan, its preconditions must be satisfied, and it is assumed that the world will be modified by the effects of its execution.

These pieces of information are the input of a planning algorithm. Internally, the planner will use this input to search for a plan. There is one mechanism in a planning algorithm that is important for our proposal: the *action selection* mechanism. This mechanism chooses the action to be added to the plan in a particular iteration of the algorithm. For instance, Weld [48] defines this mechanism by the nondeterministic *choose* function; nevertheless, it might be redefined in accordance with every specific planning problem.

At this point we can explain how the argumentation process may be modelled as a planning problem. To this end, we outline how each input of the planning problem should be defined in order to generate argumentation plans:

– *Initial state*: the conflict is the beginning point of argumentation, and it is described in the negotiation information. The initial state describes the world where the conflict takes place.
– *Final state*: in a conflictive situation, the agent's goal is to reach an expected agreement. Therefore, this is the argumentation process goal, too. Thus, the final state represents the expected agreement, which is a proposal generated by the agent. Hence, the obtained argumentation plan will support the expected agreement in the same way as an argument supports a proposal.

  For instance, if the agent *ag1* needs to execute an action *alpha* to fulfil a goal *g1*, but that action execution keeps its opponent *ag2* from fulfilling a goal *g2*, the initial state *i* should include information as *isgoal(ag1, g1)*, *isgoal(ag2, g2)*, *believe(ag1, imply(alpha, g1))* and *believe(ag2, imply(alpha, not(g2)))*; and the final state *f* should represent the execution of the action *alpha* (Section 3 provides more detailed information about this).
– *Actions*: as we described above, a rule for argument generation defines the condition to create an argument, so we can say that the argument is the effect of the rule. For that reason, we can define actions to generate arguments with the same rule patterns, where the action preconditions are the conditions to generate an argument and where

the action effect represents the argument. Furthermore, we define actions that outline the opponent's possible responses in order to represent the effects caused by argument acceptances[3].

Moreover, we should emulate both argument generation and selection mechanisms in the planner. In what follows, we describe how both mechanisms are present in the planner:

– *Argument generation*: the planner adds new actions to the plan by checking its preconditions and its effects in view of the current state and the expected final state of the world. Moreover, the rules to generate arguments can be seen as actions in a plan. Then, when the planner establishes what actions might be added to the plan, implicitly, the planner actually generates the candidate arguments.

– *Argument selection*: for the same reason as explained above, the action selection mechanism of the planner emulates the argument selection. However, it should be taken into account that in a traditional planning algorithm the action selection mechanism is implemented as a nondeterministic function. This function does not consider the preferences stored in the agent's mental state. In contrast, the argument selection is traditionally made on the basis of these preferences. Therefore, we need to adapt the action selection function of a planning algorithm in order to consider the agent's preferences about argument selection.

All in all, the argumentation in a negotiation process can be modelled as a planning problem representing the main characteristics of this process as inputs and mechanisms of a planning algorithm.

3. Definition of initial state, final state, and actions to generate argumentations plans

The negotiation information, such as the conflict context, the possible agreements, the amount of agents involved and the agents' mental states, varies from one negotiation to another. However, some characteristics of the negotiation process, such as the types of arguments the agent can use, the rules to generate them and their preconditions and effects, do not change with the negotiation. Instead, they change with the growing the experience of the negotiator. Thus, the actions for argument generation can be similar in each negotiation, whereas the initial and final states should be defined for each one.

In the following sections, we will identify general predicates that are part of the negotiation language. Furthermore, we will describe how the initial and final states and the actions for argument generation are built.

3.1 Negotiation language

First, we will define a simple negotiation language *L* that we will use in the definition of the planning problem[4]. This language is composed of predicates that represent the information that the agent has in its mental state. This information includes goals, beliefs, preferences and abilities (both current and historic), which are inspired in the BDI agents [36] and the negotiation language described in [23]. Also, the predicates represent the information about the negotiation context and types of arguments (appeals, rewards and threats [4, 35]). The basic predicates of *L* are:

– *iam(X)*: *X* is the negotiator agent.

– *isagent(X)*: *X* is an agent.

– *believe(X, B)*: *X* believes *B*. Agent *X* has *B* in its beliefs.

– *isgoal(X, G)*: *X* pursues goal *G*. Agent *X* has *G* in its goals.

– *prefer(X, G1, G2)*: *G1* and *G2* are *X*'s goals, and *X* prefers to fulfil *G1* over *G2*.

- *imply(A, B)*: *A* implies *B* (it represents the classical inference).
- *cando(X, A)*: *X* can perform action *A*. It means that agent *X* has the resources to perform action *A*, or that another agent has committed to perform it.
- *do(X, A)*: *X* will perform action *A*.
- *action(A, C, P, E):* it represents an action used by the planning algorithm: name *A*, contidions *C*, preconditions *P* and effects *E*.
- *has(X, S): X* has resource *S*.
- *pastpromise(X, Y, P)*: *X* promised *P* to *Y*, but has not fulfilled it yet.
- *wasgoal(X, G)*: *X* pursued goal *G* in the past.
- *did(X, A)*: *X* performed action *A* in the past.
- *fulfilled(X, G, A)*: in the past, *X* fulfilled goal *G* by performing action *A*.
- *appeal(X, Y, Q, J)*: *X* will use an appeal to persuade *Y*. *Q* represents the proposal and *J* is its justification.
- *reward(X, Y, Q, R)*: *X* will use a promise of a future reward to persuade *Y*. *Q* represents the proposal and *R* is the promised reward.
- *threat(X, Y, Q, T)*: *X* will use a threat to persuade *Y*. *Q* represents the proposal and *T* is the damaging effect.

We assume that the negotiation information is expressed in *L*.

3.2 Initial and final states

As mentioned before, the initial state *i* must describe the world where the conflict takes place in order to generate argumentation plans. This is the negotiation information, where the information about the conflict is represented. Therefore, the initial state will be defined as the negotiation information, and since this information changes from one negotiation to another, the initial state will vary according to the negotiation problem, too.

On the other hand, the final state *f* represents the state of agreement that the agent wants to arrive at through the argumentation. Consequently, the predicates contained in this state will depend on the kind of agreement that the agent can reach. In this work, we have considered agreements about task execution, but there are other possibilities as well. For example, if the expected agreement is that agent *ag1* accepts to perform action *alpha*, the final state should include *do(ag1, alpha)*. However, if our agent only wants to persuade *ag1* by appealing to *ag1* to believe *b*, the final state might be expressed as *believe(ag1, b)*.

In addition, our proposal makes it possible for the final state to have some free variables, in order to instantiate them conveniently. For example, the final state may be composed of *do(X, alpha)*, where *X* will be instantiated with an opponent that the agent can persuade by executing *alpha*. So, after executing the planner, the agent will obtain the arguments that it should utter and the opponent whom it should negotiate with.

3.3. Actions

In order to emulate argument generation in the planning algorithm, the plan actions represent the arguments that the agent can utter to other counterparts. Before defining these actions, we briefly introduce the argument types that the agent can generate in the argumentation-based negotiation context. Three general argument types are defined in the literature about argumentation-based negotiation: appeals (Amgoud and Prade [4] define it as explanatory arguments), rewards and threats [42, 23]. *Appeals* are used to justify a proposal; *rewards* are used to promise a future reward; and *threats* are used to warn about negative consequences in case the counterpart does not accept a proposal. Next, for each argument type, we will present the actions to generate it, according to the axioms defined in the framework of Kraus et al. [23]. Figure 2 illustrates how an explicit rule (informally expressed) may be modelled as an action for a plan construction.

Figure 2. From rules to actions for argument generation.

We distinguish between two general structures of actions: *create-argument* actions (*action(create-argumentX(), [Conditions], [Preconditions: to generate the argument], [Effects: argument(I,J,A,B)])*) and *accept-argument* actions (*action(accept-argumentX(), [Conditions], [Preconditions: argument(I,J,A,B)], [Effects: of accepting the argument])*). The first depict argument generation such as in rules, whereas the latter represent the counterpart's acceptance of the argument. Thereby, we can define several *create-argument* actions to generate the same argument type, but we must only define one *accept-argument* action, whose precondition is the argument, to reflect the argument effect in the current state.

3.3.1. Appeals

By varying the premises of the appeals, we can define several of them: past promise, counterexample, prevailing practice, self-interest, transitive and trivial appeals. Moreover, we separate the appeals into two parts. The first subgroup includes appeals to our opponents to perform a given action. For example, we can define the following action:

– Action: *createCounterexampleAppeal(X, Y, Action, Goal)*
  Description: it is similar to the previous appeal, but the historic information is about its opponent.
  Preconditions: *iam(X), isagent(Y), isgoal(Y, Goal), believe(Y, imply(Action, not(Goal))), believe(Y, imply(ActionB, not(Goal))), wasgoal(Y, Goal), did(Y, ActionB)*
  Effects: *appeal(X, Y, do(Y, Action), [did(Y, ActionB), imply(ActionB, not(Goal))])*
– Action: *acceptAppeal(X, Y, Alpha, Justif)*
  Description: it represents the acceptance of the appeal by an opponent. The effects include: the agent's commitment to perform *Alpha*, and the agent *X*'s ability to count on that execution (*cando(X, Alpha)*).
  Preconditions: *appeal(X, Y, do(Y, Alpha), Justif)*
  Effects: *do(Y, Alpha), cando(X, Alpha)*

The second subgroup includes appeals to our opponent to support a given belief. For instance:

– Action: *createPrevailingPracticeJustificationAppeal(X, Y, Z, Action, Goal)*
  Description: it uses historic information (*fulfilled(Z, Goal, Action)*) about a third agent as justification.
  Preconditions: *iam(X), isagent(Y), isagent(Z), believe(X, fulfilled(Z, Goal, Action))*
  Effects: *appeal(X, Y, believe(Y, imply(Action, Goal)), [fulfilled(Z, Goal, Action)])*
– Action: *acceptJustificationAppeal(X, Y, Alpha, Justif)*
  Description: it represents the acceptance of the appeal by the opponent. Therefore, *Y* believes *Alpha*.
  Preconditions: *appeal(X, Y, believe(Y, Alpha), Justif)*
  Effects: *believe(Y, Alpha)*

Also, other more specific appeals can be defined:

– Action: *createNIAppeal(X, Y, ActionA, ActionB, Goal)*

  Description: this appeal is used to rebut a refusal to execute an action, whose justification states that the execution does not make it possible to achieve a *Goal*. The appeal is justified on the grounds that another action, with which the same *Goal* can be achieved, exists.

  Preconditions: *iam(X), isagent(Y), notEqual(ActionA, ActionB), believe(Y, imply(do(Y, ActionA), not(Goal))), believe(X, imply(do(Y, ActionB), Goal)), believe(X, imply(Resources, do(Y, ActionB))), action(ActionB, C, P, E)*

  Effects*: appeal(X, Y, not(imply(do(Y, ActionA), not(Goal))), [imply(Resources, do(Y, ActionB)), imply(do(Y, ActionB), Goal), action(ActionB,C,P,E)])*

– Action: *acceptNIAppeal(X, Y, ActionA, ActionB)*

  Description: it represents the acceptance of the new alternative action.

  Preconditions: *appeal(X, Y, not(imply(do(Y, ActionA), not(Goal))), [imply(Resources, do(Y, ActionB)), imply(do(Y, ActionB), Goal), action(ActionB, C, P, E)])*

  Effects: *do(Y, ActionA), not(believe(Y, imply(do(Y, ActionA), not(Goal)))), forAll(member(R, Resources), [believe(Y, imply(do(X, giveResourceTo(X, Y, R)), Goal)])*

3.3.2 Rewards

Different actions related to the promise of future rewards can be defined. We show a general action to generate this kind of argument below:

– Action: *createReward(X, Y, ActionR, ActionP, Goal)*

  Description: it is similar to the previous appeal, but *Goal* only belongs to *Y*.

  Preconditions: *iam(X), isagent(Y), isgoal(Y, Goal), believe(Y, imply(ActionR, Goal)), cando(Y, ActionP), cando(X, ActionR)*

  Effects: *reward(X, Y, do(Y, ActionP), [do(X, ActionR)])*

– Action: *acceptReward(X, Y, Alpha, Beta)*

  Description: it represents the acceptance of rewards. Consequently, *Y* undertakes to execute *Alpha* in exchange for the execution of *Beta* by *X*. As in the acceptance of appeals, *X* obtains the ability to execute *Alpha*.

  Preconditions: *reward(X, Y, do(Y, Alpha), do(X, Beta))*

  Effects: *do(Y, Alpha), do(X, Beta), cando(X, Alpha)*

3.3.3 Threats

We define the most general threats, but others may be defined by changing their preconditions.

– Action: *createThreat(X, Y, ActionT, ActionP, Goal)*

  Description: *X* threatens *Y* in the following way: if *Y* does not perform *ActionP*, *X* will perform *ActionT*, because *ActionT* contradicts a goal *GoalA* preferred by *Y*.

  Preconditions: *iam(X), isagent(Y), isgoal(Y, GoalA), isgoal(Y, GoalB), cando(X, ActionT), cando(Y, ActionP), prefer(Y, GoalA, GoalB), believe(X, imply(ActionT, not(GoalA))), believe(X, imply(ActionP, not(GoalB)))*

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

– Action: *acceptThreat(X, Y, Alpha, Beta)*

  Description: it represents the acceptance of a threat. *X* will not perform *Beta* if *Y* performs *Alpha*.

  Preconditions: *threat(X, Y, do(X, Alpha), do(X, Beta))*

  Effects: *do(Y, Alpha), not(do(X, Beta)), cando(X, Alpha)*

Additional actions can be found in Appendix A. In order to clarify our proposal, we have only defined actions for the main arguments. Nevertheless, other actions may be defined by changing preconditions and effects.

4. Planning algorithm for argumentation plan generation

Rahwan et al. [31] consider argument selection as the essence of the argumentation-based negotiation strategy. This mechanism consists of selecting one argument to be uttered from among the set of candidate arguments that might be uttered. The agent selects an argument on the basis of, for instance, the type of argument and the opponents that it must persuade by considering the argument strength [23] or its trust in its opponents [33, 34]. Thus, if there are two different candidate arguments, the agent will select one by applying one of those policies to determine which one should be uttered.

In this context, the action selection mechanism of the planning algorithm must take into account the argument selection policy of the agent in order to generate argumentation plans. In our work, we represent this policy into the agent's mental state as preferences for actions and goals. So, if the agent prefers to utter appeals instead of threats, the *create-argument* actions for appeals will have a higher preference level than the *create-argument* actions for threats in the agent's mental state. Moreover, these preferences may change depending on other factors, such as the opponent's trust. For example, when the opponent's trust is low, the agent can prefer to use a strong argument (e.g. a threat), whereas when the trust is high, the agent may prefer to use a weak one (e.g. an appeal) [33].

To represent this capability in the planning algorithm, we propose the use of a preference-based planning algorithm. Planning algorithms do not usually use all the available knowledge in the agent's mental state for the plan conception. In traditional planning [48] action selection is carried out in a nondeterministic way. It is modelled by a nondeterministic *choose* function. The *choose* function can be implemented by a random selection algorithm or a domain specific algorithm. In contrast, in preference-based planning algorithm, this function takes into account the user's or agent's preferences to choose among different alternatives. In recent years, several works have emphasized the importance of considering preferences in the plan construction [44, 7, 43]. In our proposal, we have used a partial order planning based on preferences [9], but another algorithm can be used to generate argumentation plans. In this algorithm, the *choose* function was redefined in order to take into account the preferences for actions and goals in the context of argument generation.

Since our *choose* function must select the most preferable action representing an argument, we define the next format of the agent's mental attitude to be used by the planner: *preference(Q, Ac, Ad, level)*; where *Q* represents the goal to be achieved, *Ac* is the action that produces *Q*, *Ad* is the action that needs to accomplish *Q*, and *level* determines how preferable the attitude is. Also, it is possible to add a body to the rule; the body can restrict, in a more precise way, the scope of the preference.

For instance, with *preference(appeal(_,ag2,_,_), createPrevailingPracticeAppeal(_, ag2, _,_,_), acceptAppeal(ag2,_,_,_), 80)* the agent specifies a preference (e.g. between 0 and 100) of 80 for the goal *appeal(_,ag2,_,_)* (when the goal is to generate an appeal to persuade agent *ag2*) when this is produced by the action *createPrevailingPracticeAppeal*, and consumed by the action *acceptAppeal*. In the context of the argumentation process, this mental attitude represents a relative high preference level to use an appeal to prevailing practice, instead of another kind of argument, when the opponent is agent *ag2*, probably because trust in *ag2* is high. However, if the trust in the opponent is low, the agent might prefer a threat to an appeal. In this case, the attitude should be: *preference(threat(_,ag2,_,_), createThreat(_,ag2,_,_,_), acceptThreat(ag2,_,_,_), 80)*.

Initially, we adopt the approach described by the framework of Kraus et al. [23]. In this approach, the preferences over arguments are given by the argument strength, taking into account the appeals as the weakest argument, and the threats as the strongest argument. That is, preferences with respect to the general argument types: *preference(_, acceptJustificationAppeal(), _, 40)*; *preference(_, acceptAppeal(), _, 30)*; *preference(_, acceptReward(), _, 20)*; and *preference(_, acceptThreat(), _, 10)*; also, preferences within each general type of arguments: *preference(_, createPrevailingPracticeAppeal(), acceptAppeal(), 40)*; *preference(_, createCounterexampleAppeal(), accept-Appeal(), 30)*; *preference(_, createPastPromiseAppeal(), acceptAppeal(), 20)*; *preference(_, create-SelfInterestAppeal(), acceptAppeal(), 10)*; *preference(_, createPrevailingPracticeJustificationAppeal(), accept-JustificationAppeal(), 40)*; *preference(_, createCounterexampleJustificationAppeal(), acceptJustificationAppeal(), 30)*; *preference(_, createTransitiveJustificationAppeal(), acceptJustificationAppeal(), 20)*; *preference(_, create-TrivialJustificationAppeal(), acceptJustificationAppeal(), 10)*; *preference(_, createRewardBoth(), acceptReward(), 20)*; *preference(_, createReward(), acceptReward(), 10)*; *preference(_, createThreat(), acceptThreat(), 10).*

However, the agent can modify the existing preferences and add more general ones (e.g. differentiating by agents), as long as it keeps obtaining useful information from its opponents (see [26] for more details).

Moreover, as we have stated previously, several factors influence the argument selection process. The preference format presented in this section facilitates its consideration and combination in that selection. We show some examples in the following preferences:

a. *preference(appeal(_, _, do(_,Action), _), createPastPromiseAppeal(_, _, Action), acceptAppeal(_,_,Action,_), 40) :- utility(Action, medium).*

b. *preference(reward(_, _, Action, _), createReward(_, _, Action, _, _), acceptReward(_, _, _, _), 85) :- urgency(Action, high).*

c. *preference(threat(AgS, AgD, _, _), createThreat(AgS, AgD, _, _, _), acceptThreat(AgD, AgS, _, _), 0) :- authority_role(AgS, AgD, boss).*

These mental attitudes indicate different levels of preference according to three different factors: the utility associated to the action execution [33], the urgency to execute the action or to get the resource [13] and the relation of authority among the agents [42]. Additionally, factors can be combined to enhance the accuracy of the situation in which the preference must be applied.

4.1. Using argument selection preferences to consider the degree of believability of the negotiation information

Arguments are built with tentative information that the agent gathered from previous interactions (i.e. beliefs). That is, the agent can suppose that some information is tentative or probable after observing other agents, but the agent cannot be completely certain about this information. As Amgoud and Prade [4] have proposed, it is necessary to take into account this fact during the argument selection process, by selecting the arguments with higher support. Admittedly, we have defined a simple belief format that does not represent any degree of believability in order to simplify the knowledge representation. However, we suppose that the agent knows the certainty degree of each fact of the negotiation information. That is, we assume that the agent is able to calculate these values independently of the method that it applies to obtain them. Different methods can be applied to compute these values. For example, we can assume that any set of facts has a preference order in it which derives from the stratification of the knowledge base in which the negotiation information is stored by considering its degree of believability [2]. Also, a graded BDI model can be used for this task [10]. Taking into account this idea, we can define a preference to prioritise the selection of arguments that have been built with the highest degree of believability.

- *preference(Arg, ActionCreateArg, ActionAcceptArg, believability(ActionCreateArg, Arg))*.

Where *Arg* is the argument generated, *ActionCreateArg* is the action that generates the argument, and *ActionAcceptArg*, the action that consumes it. The function *believability(ActionCreateArg, Arg)* returns the believability degree of the argument considering the degree of believability of the preconditions of *ActionCreateArg*, since these preconditions are the facts observed to build the argument. In other words, the believability degree of the argument is calculated as the result of multiplying the believability degrees of the facts that support it. Thus, the planning algorithm will prefer to add the arguments with the highest degree of believability to the plan that is being built.

4.2. Example of argumentation plan construction

With the purpose of illustrating the ideas presented in the previous section, we show an example of the utilization of planning for the generation of argumentation plans. To this end, we use the proposed planning algorithm introduced above.

Given the agents *ag1*, *ag2*, *ag3* and *ag4* on a competitive environment, *ag1* should reach an agreement with some opponent in order to perform an action *a5*, since this action is needed to fulfil one of its goals. The negotiation information of *ag1* is: *iam(ag1), isagent(ag2), isagent(ag3), isagent(ag4), isgoal(ag1, g1), cando(ag1, a1), cando(ag1, a6), believe(ag1, imply(a5, g1)), believe(ag1, fulfilled(ag4, g2, a4)), believe(ag1, imply(a4, g2)), isgoal(ag2, g2), cando(ag2, a2), cando(ag2, a5), isgoal(ag3, g3), cando(ag3, a3), cando(ag3, a4), believe(ag3, imply(a1, g3))*, as well as its preferences about the arguments (as we described in Section 4). So, the first information constitutes the initial state of the planner, the preferences are part of the agent's mental attitudes, and the final state is composed of the predicate *do(X, a5)*. This final state represents the agreement that *ag1* expects to reach, because of the need to perform *a5* in order to fulfil *g1*. The actions of the planner have already been defined in Section 3.3.

Figure 3. Argumentation plan.

The resulting argumentation plan is shown in Figure 3, where we can see the actions that represent the arguments, which *ag1* should utter during the argumentation process. Firstly, *ag1* should persuade *ag4*, using a reward, to carry out the execution of *a4* in order to offer it to *ag2* in exchange for *a5*. And secondly, *ag1* should use an appeal to prevailing practice in order to lead *ag2* to believe that the execution of *a4* implies reaching goal *g2*.

The preferences of our algorithm provide a useful versatility to planning. For example, if *ag1* increases the preference level of the action *createTrivialJustificationAppeal* to 50, the planner will prefer that argument instead of *createPrevailingPracticeJustificationAppeal*.

5. Integrating argumentation plan generation into the general planning process

Once the mechanism to construct argumentation plans is defined, we can integrate this construction into the general planning of the agent. The agent plans the course of action that it must follow to achieve its goals. This plan may be composed of several actions. Some of these actions are under the direct control of the agent. Other actions are not under the control of the agents (according to the work of Kraus et al. [23], *intend-to-do* and *intend-that* actions respectively). The latter must be negotiated. The agent usually builds a plan and starts to execute it by stopping at every non-controlled action in order to negotiate its execution with some of the agents that can do this action. However, when the negotiation fails, in the least expensive case, the agent must find another alternative to reach the same agreement. In the most expensive case, the agent must replan its course of action. This happens because when planning the course of action, the agent did not take into account the negotiation as a critical action within the general plan. In this situation, both the failed negotiation and the actions executed before the negotiation, which cannot be reused (in Figure 4.a, *Action 1*), consume resources.

Figure 4. Planning and negotiation.

At this point, the argumentation plans play a fundamental role. If the agent has an argumentation plan, which indicates the necessary arguments to agree on the execution of an *intend-that* action, it will have a good hint to trust the viability of the agreement. Therefore, if we integrate the construction of these plans into the construction of the general plan, the agent will have a clear vision of the negotiations that it should carry out, with whom it should negotiate and which arguments it should utter to reach an agreement. Thus, the impossibility of reaching an agreement can be detected at an early stage (planning time), and the agent may modify its general plan, without the need to start executing it. As shown in Figure 4.b, the planning algorithm cannot generate an argumentation plan to agree on the execution of *Action 2*, so this action is discarded and other actions for which the argumentation plans can be generated are added. In other words, the agent has the necessary arguments to reach the agreement during the negotiation. It is worth noticing that the failure of the negotiation of *Action 2* is detected in planning time, and not in execution time as in the previous example, where the negotiation was not taken into account as a part of the general planning of the agent.

The integration can easily be accomplished. On the one hand, we have the definitions of the initial state, final state and actions of the general planning problem. On the other hand, we need the same definitions for each argumentation plan required for each action *intend-that* that is added to the general plan.

The initial state of the general plan includes the information that the agent has about the world in which it is performing. Additionally, the general actions, which are settled before the argumentation plan, modify the world (effects) in which the argumentation plan will be executed. Therefore, these actions also contribute to form the implicit initial state of the argumentation plan. Figure 5.a shows a figurative view of the plan. This figure illustrates how the initial state of the argumentation plan is composed of facts of the general initial state and facts generated by the previous actions. As mentioned above, this initial state is implicit, because it is not defined explicitly by the agent, but it is created during the general plan construction (Figure 5.b).

Something similar occurs with the final state of the argumentation plan. No intermediate final state is explicitly defined, but the actions which are not under the control of the agent have a special precondition that forces the planning algorithm to build such a plan. The precondition added is *do(Agent, Action)*, and it implicitly represents the final state of the argumentation plan: the agreement which the agent wants to reach. We call these actions *conditionals*.

Besides adding the new precondition to the conditional actions, it is necessary to have an alternative to the argumentation plan when actions of the general plan cannot be supported by one of these plans and when substitute actions do not exist. In other words, there are situations in which an argumentation plan that agrees on the execution of a necessary action cannot be built, either because there is not sufficient information or because there is no evidence indicating that the opponent, from whom the agent should request such execution, rejects it. In these cases, we cannot claim that there is not a viable general plan, only that there is not an argumentation plan. Therefore, in order not to deprive the planning algorithm of the possibility to build a valid general plan in these situations, we maintain one *unconditional* action for each existing conditional one. This unconditional actions are included in the initial definition of the problem, without the precondition *do(Agent, Action)*. One could think that this would lead back to the original problem (Figure 4). However, to avoid such a problem, we add preferences that prioritise the usage of conditional actions over unconditional ones. Thus, the planning algorithm first searches for the *intend-that* actions that can be supported by an argumentation plan, but if this plan cannot be built, the algorithm includes the unconditional actions that must be requested and negotiated traditionally.

Figure 5. Figurative and real view of the integration.

To formalise the plan construction, Figure 6 shows the algorithm that allows the agents to build a general plan. Initially, the agent, which must achieve a certain goal (step 2), defines the initial and final states, the actions and the preferences as defined above (step 4-7). Then, the agent builds a general plan (step 8). If the planning algorithm can build a plan, the agent tries to execute it (step 13); otherwise the agent waits for a change in the context that allows it to build a new plan (step 15). During the execution of the plan three situations may occur: (a) the plan is executed successfully; (b) an argumentation plan fails, then the agent can continue negotiating traditionally and following the general plan; or (c) the general plan fails, then the agent must replan the course of action, due to the fact that its goals remain unachieved. After the plan execution, the agent checks if its entire goals have been fulfilled (step 2). If so, the agent finishes the execution and stays on standby until new goals appear. On the other hand, if there are unachieved goals, the agent needs to obtain a new plan. To do this, the agent can request a new plan from the planning algorithm (replanning, step 10) if the context information has not changed (this method has not arguments because the replanning is performed using the same information used to plan). Otherwise, a new plan must be built, since the initial definition (specifically, the initial state) is no longer valid (step 4-8). Notice that the replanning time can be improved if the replanning process is carried out concurrently with the execution of the original plan. Thus, if the original plan fails, the replanning process could be able to build a replan more quickly.

Figure 6. Algorithm to general plan construction.

6. Experimental results

We evaluated our proposal within the multiagent platform JADE [8], on which we implemented two kinds of negotiator agents: one with the ability of building argumentation plans, and the other without it. To carry out the experiments, we extended a well-known and widespread negotiation scenario [30, 39, 29, 32]. In this scenario, originally presented in the work of Parsons and Jennings [28], agents have to execute tasks to achieve their goals with scarce resources. Due to scarce resources, conflicts arise among the agents' goals, which must be resolved through negotiation.

6.1. Negotiation scenario

In the original negotiation scenario, there are two home-improvement agents, which we named *Picty* and *Mirry*. These agents have to carry out a task in a house. Both of them have specific goals: *Picty* must hang a picture on a wall, and *Mirry* must hang a mirror. To achieve their goals, the agents have a set of resources:

– *Picty* has a picture (*p1*), a screwdriver (*sd1*), a screw (*s1*) and a hammer (*h1*).
– *Mirry* has a mirror (*m1*) and a nail (*n1*).

In addition, *Picty* knows that it can hang the picture by using a hammer and a nail, and it believes that it is possible to hang the mirror with a screwdriver and a screw. *Mirry* supposes that it can only hang the mirror by using a hammer and a nail.

In this context, *Picty* should request *Mirry* a nail. However, *Mirry* will refuse the request because it would make it impossible to fulfil its goal (to hang the mirror). Therefore, if the negotiations were limited to the proposal exchange, they would stagnate in a cycle of requests and refusals. In this cycle, *Mirry* would also request *Picty* the hammer, to which *Picty* would refuse, because it is the only tool to hang its painting. In contrast, if the agents exchange arguments, the conflicts can be resolved. *Picty* can utter an argument to persuade *Mirry* that it is possible to hang a mirror with a screwdriver and a screw. Moreover, *Picty* can offer these resources in exchange for the nail. Thus, both agents would achieve their goals. Thus, the conflict would not be resolved without arguments that modify *Mirry's* beliefs.

The preceding paragraphs have introduced the original scenario. We transformed this scenario into a multilateral one, in order to evaluate the characteristics of the argumentation plan generation more accurately. Thus, we added a third agent, *Chairy*, which must fix a chair (*bc1*). Additionally, we added resources and modified some of the agents' beliefs. In the extended scenario, *Mirry* has a carpenter's glue pot (*g1*), which is necessary to fix the chair (*bc1*), and Picty needs a fixed chair to hang the picture.

These new facts constitute new interdependences among the three agents, which have to reach agreements to resolve the conflicts. In the new scenario, we can find the following conflicts:

– Conflict between *Picty* and *Mirry*: it is the original conflict detailed by Parsons and Jennings [28]. *Picty* needs the nail that *Mirry* has, and *Mirry* needs the hammer that *Picty* has.
– Conflict between *Picty* and *Chairy*: the first one needs the chair to fulfil its goals, but the second must fix this one before.
– Conflict between *Chairy* and *Mirry*: *Chairy* needs carpenter's glue to repair the chair and *Mirry* wants to keep it.

Notice that there are links between the second and the third conflicts. So, if the conflict between *Chairy* and *Mirry* is settled, the conflict between *Picty* and *Chairy* could be resolved, too. Moreover, the conflict between *Picty* and *Mirry* must be solved before solving the one between *Chairy* and *Mirry*, as the following section shows. In this way, the three conflicts represent circular dependencies that must be solved by the planning algorithm: *Picty* needs *Chairy*'s chair, *Chairy* needs *Mirry*'s glue, and *Mirry* needs *Picty*'s hammer, among others dependencies.

Table 1. Mental states of the agents *Picty*, *Mirry* and *Chairy*.

6.2. Conflict resolution using argumentation plans

First, we expressed the facts of the world that define the scenario in language *L*: *isagent(picty), isagent(mirry), isagent(chairy), has(picty, picture(p1)), has(picty, screwdriver(sd1)), has(picty, screw(s1)), has(picty, hammer(h1)), has(mirry, mirror(m1)), has(mirry, nail(n1)), has(mirry, glue(g1)), has(chairy, brokenChair(bc1)), cando(picty, hangAPicture), cando(mirry, hangAMirror), cando(chairy, fixAChair).*

Next, we defined the agents' mental states and the general actions that can be used to build plans. Table 1 shows the facts that compose the mental states of the agents *Picty*, *Mirry* and *Chairy*. In our scenario, *Picty* is able to build argumentation plans and integrate these plans into the general planning. In contrast, *Mirry* and *Chairy* will first build a general plan, and then, they will try to negotiate the *intend-that* actions.

Moreover, we defined the actions that each agent knows. Notice that each action has four parameters: name and action parameters; constraints (checks over actions variables, such as domain variables); preconditions (facts that the world must fulfil so that the action can be executed); and effects (changes that the world will experience after action execution).

The actions for *Picty* are the following:

– *action(hangAPicture(Agent, Picture, Hammer, Nail, Chair), [iam(Agent)], [cando(Agent, hangAPicture), has(Agent, picture(Picture)), has(Agent, hammer(Hammer)), has(Agent, nail(Nail)), has(Agent, chair(Chair))], [pictureHanging(Picture), not(has(Agent, nail(Nail)))]).*

Note: As this action is under the control of *Picty*, it does not need to add the precondition *do(Agent, Action)*.

– *action(giveResourceTo(AgentS, AgentD, Resource), [isagent(AgentD), isagent(AgentS), notEqual(AgentS, AgentD)], [has(AgentS, Resource), do(AgentS, giveResourceTo(AgentS, AgentD, Resource))], [has(AgentD, Resource), not(has(AgentS, Resource))]).*

The precondition *do(AgentS, giveResourceTo(AgentS, AgentD, Resource))* is necessary to generate the argumentation plan that supports the request of the Resource as shown in Section 5.

– *action(giveResourceToUnconditional(AgentS, AgentD, Resource), [isagent(AgentD), isagent(AgentS), notEqual(AgentS, AgentD)], [has(AgentS, Resource)],[has(AgentD, Resource), not(has(AgentS, Resource))]).*

– *action(fixAChair(Agent, BrokenChair, Glue), [isagent(Agent)], [cando(Agent, fixAChair), has(Agent, brokenChair(BrokenChair)), has(Agent, glue(Glue)), do(Agent, fixAChair(Agent, BrokenChair, Glue))], [has(Agent, chair(BrokenChair)), not(has(Agent, glue(Glue)))]).*

– *action(fixAChairUnconditional(Agent, BrokenChair, Glue), [isagent(Agent)], [cando(Agent, fixAChair), has(Agent, brokenChair(BrokenChair)), has(Agent, glue(Glue))], [has(Agent, chair(BrokenChair)), not(has(Agent, glue(Glue)))]).*

- *action(hangAMirror(Agent, Mirror, Screwdriver, Screw), [isagent(Agent)], [cando(Agent, hangAMirror), has(Agent, mirror(Mirror)), has(Agent, screwdriver(Screwdriver)) ,has(Agent, screw(Screw))], [mirrorHanging(Mirror), not(has(Agent, screw(Screw)))]).*

Due to the fact that the action *giveResourceTo* can also be executed by agent *Picty*, we added the following facts to prevent the planning algorithm from generating argumentation plans to persuade *Picty* itself:

- *do(picty, giveResourceTo(picty, AgentD, Resource)),* where *AgentD* is some agent and *Resource* some resource that *Picty* has.

The actions that Mirry could execute are:

- *action(hangAMirror(Agent, Mirror, Hammer, Nail), [iam(Agent)], [has(Agent, mirror(Mirror)), has(Agent, hammer(Hammer)), has(Agent, nail(Nail))], [mirrorHanging(Mirror), not(has(Agent, nail(Nail)))]).*
- *action(giveResourceTo(AgentS, AgentD, Resource), [isagent(AgentD), isagent(AgentS), notEqual(AgentD, AgentS)], [has(AgentS, Resource)], [has(AgentD, Resource), not(has(AgentS, Resource))]).*

Agent *Mirry* is not able to generate argumentation plans, so it does not distinguish among agents under or beyond its control during the planning stage.

Last, *Chairy* can execute the following actions:

- *action(fixAChair(Agent, BrokenChair Glue), [iam(Agent)], [has(Agent, brokenChair(BrokenChair)), has(Agent, glue(Glue))], [has(Agent, chair(BrokenChair)), not(has(Agent, brokenChair(BrokenChair)))]).*
- *action(giveResourceTo(AgentS, AgentD, Resource), [isagent(AgentD), isagent(AgentS), notEqual(AgentD, AgentS)], [has(AgentS, Resource)], [has(AgentD, Resource), not(has(AgentS, Resource))]).*

Taking into account these mental states and the action definitions, each agent must build a plan that guides its actions to reach its goals. Next, we will analyze how *Picty* builds its plan.

6.2.1. Building Picty's general plan

To build a general plan, *Picty* must first define the initial and final states. As we detailed in Section 3.2, the initial state is composed of all information that the agent has about the negotiation context. In this scenario, the initial state was composed of facts, beliefs, facts about opponents and actions. The final state was determined by the goal *pictureHanging(p1)*, which represents the picture *p1* hanged on the wall. Finally, the actions used were those defined in the previous section and the actions for argument generation defined in Section 3.3. Moreover, the preferences for argument generation were defined in Section 4.

Figure 7. *Picty*'s general plan.

Figure 7 shows the general plan built by *Picty*. The references of this figure are detailed and explained below:

- pArg$_1$: Arg$_1$ preconditions fulfilled by the initial state.
  - *believe(mirry, imply(do(mirry, giveResourceTo(mirry, picty, nail(n1))), not(mirrorHanging(m1)))); isgoal(mirry, mirrorHanging(m1)); believe(picty, imply(do(mirry, hangAMirror(mirry, m1, sd1, s1)), mirrorHanging(m1))); believe(picty, imply([has(mirry, mirror(m1)), has(mirry, screwdriver(sd1)), has(mirry, screw(s1))], do(mirry, hangAMirror(mirry, m1, sd1, s1)))).*
  - *action(hangAMirror(mirry, m1, sd1, s1), [], [iam(mirry), has(mirry, mirror(m1)), has(mirry, screwdriver(sd1)), has(mirry, screw(s1))], [mirrorHanging(m1), not(has(mirry, screw(s1)))]).*

- pArg$_2$: Arg$_2$ preconditions fulfilled by the initial state.
  - *believe(mirry, imply(do(mirry, giveResourceTo(mirry, chairy, glue(g1))), not(has(mirry, glue(g1))))); isgoal(mirry, has(mirry, glue(g1))); isgoal(mirry, mirrorHanging(m1)); prefer(mirry, mirrorHanging(m1), has(mirry, glue(g1)); cando(mirry, giveResourceTo(mirry, chairy, glue(g1))); cando(picty, giveResourceTo(picty, mirry, screwdriver(sd1))).*
- pHang: preconditions to hang the picture on the wall.
  - *has(picty, picture(p1)); has(picty, hammer(h1)).*
- pFix: *has(chairy, brokenChair(bc1)); cando(chairy, fixAChair).*
- B$_1$: *believe(mirry, imply(do(picty, giveResourceTo(picty, mirry, screwdriver(sd1))), mirrorHanging(m1))).*
- Arg$_1$: *appeal(picty, mirry, not(imply(do(mirry, giveResourceTo(mirry, picty, nail(n1))), not(mirrorHanging(m1)))), [imply([has(mirry, mirror(m1)), has(mirry, screwdriver(sd1)), has(mirry, screw(s1))], do(mirry, hangAMirror(mirry, m1, sd1, s1))), imply(do(mirry, hangAMirror(mirry, m1, sd1, s1)), mirrorHanging(m1)), action(hangAMirror(mirry, m1, sd1, s1), [], [iam(mirry), has(mirry, mirror(m1)), has(mirry, screwdriver(sd1)), has(mirry, screw(s1))], [mirrorHanging(m1), not(has(mirry, screw(s1)))])]).*
- Arg$_2$: *reward(picty, mirry, do(mirry, giveResourceTo(mirry, chairy, glue(g1))), [do(picty, giveResourceTo(picty, mirry, screwdriver(sd1)))]).*
- Do$_1$: *do(mirry, giveResourceTo(mirry, picty, nail(n1))).* Do$_2$: *do(mirry, giveResourceTo(mirry, chairy, glue(g1))).*
- Has$_1$: *has(picty, nail(n1)).* Has$_2$: *has(chairy, glue(g1)).* Has$_3$: *has(chairy, chair(bc1)).* Has$_4$: *has(picty, chair(bc1)).* Has$_5$: *has(mirry, nail(n1)).* Has$_6$: *has(mirry, glue(g1)).*

Two argumentation plans are integral parts of the general plan shown in Figure 7. The first argumentation plan is created by the appeal Arg$_1$. The goal of this argument is to persuade *Mirry* to give nail *n1*. This appeal indicates that there is another alternative to execute the action *hangAMirror*¸ whose preconditions do not include resource *n1*. In addition, argument Arg$_1$ not only persuades *Mirry* to accept *Picty's* request, but also modifies its mental state by adding a new belief B$_1$, which is needed to generate argument Arg$_2$.

Argumentation plan 2 is built to persuade agent *Mirry* to give the carpenter's glue *g1* to agent *Chairy*. That is achieved by rewarding *Mirry*: if *Mirry* gives *g1* to *Chairy*, *Picty* will give *sd1*. Resource *sd1* is needed by *Mirry* since it accepts argument Arg$_1$. Thus, agent *Picty* is not directly involved in these actions. However, these actions are vitally important for its goal achievement, since *Mirry* wants to hang the mirror on the wall and keep the glue, but *Chairy*, who needs the glue to repair the chair, has no resources to negotiate it. Therefore, if *Picty* considered only the conflicts in which it is involved, it will not be able to achieve its goals by itself. Additionally, it is worth noticing that *Mirry* will accept the reward because goal *has(mirry, glue(g1))* is less preferable than goal *mirrorHanging(m1)*.

As regards the fixing of the chair, the planning algorithm selects the unconditional action because it does not find facts that facilitate an argumentation plan construction. There are two reasons for this:

- There is not enough information to generate arguments that support the execution of the action.
- There is no evidence indicating that *Chairy* will refuse the action request.

For the argument Arg$_1$, belief *believe(mirry, imply(do(mirry, giveResourceTo(mirry, picty, nail(n1))), not(mirrorHanging(m1))))* indicates that *Mirry* will not give *n1* to *Picty* because it precludes it from fulfilling its goal. Also, there are similar reasons for argument Arg$_2$. The fact that makes a possible refusal evident is *believe(picty, imply(do(mirry, giveResourceTo(mirry, chairy, glue(g1))), not(has(mirry, glue(g1))))).*

Finally, after obtaining the resources needed, *Picty* will be able to execute the action *hangAPicture* and achieve its goal.

6.2.2. Solution with multiple alternatives

Up to this point, the negotiation scenario does not present alternatives to obtain the resources needed to achieve the agents' goals. However, it is feasible that several alternatives exist in a real scenario. These alternatives facilitate the achievement of the goals, but also make the decision-making process complex. Moreover, not all alternatives are viable. Therefore, any mistake in the decision-making process could lead to lack of time and resources and make the goals unattainable.

To show how our proposal works in such situations, we added new agents (*Adhy$_i$*) to the negotiation scenario. Each agent *Adhy$_i$* has carpenter's glue *g_i* (i > 1), and has one goal: to keep this resource. In this context, agent *Picty* must select the agent that it has to persuade in order to obtain the glue that *Chairy* needs by to fix the chair. Nevertheless, *Mirry* will continue being the unique agent that can be persuaded.

The evaluation of this new scenario was carried out in an incremental way. That is, we added agents *Adhy$_i$* one by one. We calculated five metrics in each situation, four about *Picty*'s performance and the fifth related to the performance of all agents. These metrics are the following:

- *T*: total time needed by *Picty* to fulfil the goal (*hangingPicture(p1)*).
- *#I*: total times that *Picty* initiated a negotiation.
- *#P*: number of times that the planning algorithm was executed since an initial stage. This happens when the agent execution starts or when the world changes after a failed plan execution and a new solution cannot be requested from the planning algorithm.
- *#Rp*: total times that the agent requested a new plan solution from the planning algorithm (replanning). After a failed plan execution, the agent must solicit a new plan solution. If the world does not change, the planning problem continues being valid and the algorithm will give a new solution.
- *#M*: total messages exchanged by all agents during the execution.

The main goal of this evaluation was to compare our proposal with the traditional negotiation process and planning. On the one hand, we evaluated the performance of agent *Picty* with the ability of building argumentation plans (*Picty_arg*) as shown in previous sections. On the other hand, we also evaluated the performance of the same agent without such ability (*Picty_sim*). The comparative results are shown in Table 2, where *#g_i* is the amount of resources *g_i* (carpenter's glue) included in the execution. Notice that *Mirry* always keeps initially resource *g_1*.

Table 2. Comparison of *Picty_arg* and *Picty_sim* performances.

As shown in Table 2, *Picty_sim*' s performance is better in the first two executions, where it fulfilled its goal in a shorter time, but required two executions of the planning algorithm. Figure 8.a shows the initial plan for *Picty_sim*. As there are no order restrictions between action *giveResourceTo(mirry, picty, nail(n1))* and *giveResourceTo(mirry, chairy, glue(g1))*, these actions can be executed in a parallel way. However, in *Picty_arg*'s plan (Figure 7), there is a relation between the argumentation plans that support these actions. This relation makes the second agreement (*g_1*)

impossible unless the argument that composes the first argumentation plan (n_1) is uttered. Therefore, if both actions are negotiated in parallel, the argument Arg_1 is not uttered before the negotiation of g_1, so the negotiation fails because *Picty* has no information to build the reward Arg_2. Thus, the negotiation for carpenter's glue fails, whereas the other is successful. After agreement, the action *giveResourceTo(mirry, picty, nail(n1))* is executed and the world is changed: now *Picty* has the nail. So, as the world has changed, the agent cannot replan, it must redefine the planning problem and plan again. Notice that the new planning does not include the negotiation of *n1* and the plan is executed successfully because belief B_1 was included in *Mirry*'s mental state of after the first execution. It is also worth noticing that, to be realistic, there is no mechanism to coordinate the moment in which the agents finish the planning stage and start the negotiations. That is, we assume that different agents can start the negotiation independently of the others.

Figure 8. Plans of *Picty_sim*.

In execution #g_2, *Picty_sim*'s performance continues being better than *Picty_arg*'s performance, but time *T* of the second execution is more than twice the time of the first one. This happens because the planning algorithm selects in a non-deterministic way which agent (*AgentS*) satisfies precondition *has(AgentS, glue(Glue))* of action *giveResourceTo(AgentS, chairy, glue(Glue))*. Figure 8.b shows *Picty_sim*'s failed plan. In this case, since there were two resources g_i, the planning algorithm selected agent *Adhy2* to instance *AgentS*. As *Adhy2* cannot be persuaded by *Picty_sim*, the plan failed and *Picty_sim* must request a new solution from the planning algorithm. The new solution is the alternative plan shown in Figure 8.a.

From the execution #g_3, *Picty_arg* obtained better results than *Picty_sim*. Figure 9.a compares the time taken by *Picty_arg* and *Picty_sim* to achieve their goals. Axis *X* represents the amount of agents that have carpenter's glue and axis *Y* represents the time taken by the agent. As can be seen, the time taken by *Picty_arg* increases almost linearly, because it decides with whom to negotiate the carpenter's glue in planning time, thus avoiding possible failures in the plan execution. In contrast, for *Picty_sim*, the time increases exponentially, because it does not take into account the negotiation within the planning process. Therefore, *Picty_sim* cannot determine in planning time with which agent it is better to negotiate. Thus, the time needed to achieve the goals increases exponentially because the agent has to execute the planning algorithm several times and requires several plan solutions until it finds the successful one.

Moreover, the number of messages uttered by the agents when *Picty_sim* is participating is meaningfully higher than when *Picty_arg* is participating. The reason for this difference lies in the different executions and failures of the plans, which lead to an excessive number of messages exchange. Figure 9.b shows the total messages uttered by the agents while *Picty_sim* and *Picty_arg* are participating in the negotiation. As that figures indicate, there is a reduction not only in the execution time of the argumentation plan generation, but also consequently, in the overload in the communication channel shared by the agents.

Figure 9. Comparative charts between *Picty_sim* and *Picty_arg* performance.

7. Discussion: argumentation plans to support decision making

As shown in the previous section, one of the main contributions of our approach is that it allows us to integrate the negotiation in general, and the argumentation in particular, within the planning stage. This integration allows the agent to take decisions in advance, in planning time, before the execution of the general plan is carried out.

Negotiation support systems have been widely studied [27, 22, 11, 41]. This kind of systems tries to assist negotiators during the different negotiation stages. Agent technology has also been integrated within these systems to model many decision-making negotiation tasks [47], especially since these agents are an excellent tool to assist users or to allow them to act on behalf of users [24, 25]. Our approach can be applied in both directions. That is, on the one hand, the agent can act on behalf of a user, taking into account his/her preferences, goals, etc. (for instance, as can be seen in [46]). In this case, the agent will negotiate autonomously. On the other hand, a personal agent can assist a user during the negotiation by using the information extracted from the integral plans. That is, a user operating a negotiation system can receive assistance to make decisions from a personal agent. Thus, although the user carries out the decision making process autonomously, the agent aims his/her in such process. In this sense, we distinguish among several decisions in which the information taken from the integral plans can be applied:

- To determine the order in which we carry out different negotiations (negotiation agenda).
- To decide with which opponent or which resource we must negotiate in case where there are several alternatives to choose from.
- To evaluate in advance the need to reach secondary agreements.

Finally, as the results have shown, making early decisions at this point allows the negotiator to save effort, time and resources.

8. Related work

Closer to our work in argumentation plans is that of discourse planning approaches, which use a planner for the construction of discursive dialogues for human-computer interaction (see, e.g. [37, 16]), and of natural language generation and AI planning (e.g. [14]). However, none of them is oriented to argumentation-based negotiation or conflict resolution or directed to reach an agreement. Moreover, the planners used in these approaches do not consider the agent's preferences for the action selection process.

On the other hand, it is worth differentiating our proposal from dialogues for deliberation (see, e.g. [45, 1]). In deliberation dialogues, agents discuss possibilities in order to agree on a joint course of action. These models work with argumentation-based dialogues and planning as in our work, but in an inverse way. In deliberation, the agents use the argumentation-based dialogue to jointly reach a plan, whereas we use planning to obtain an argumentation plan.

As we described above, we have based our work on the framework of Kraus et al. [23] (for instance, the actions to generate arguments as well as the policy to select arguments were motivated by that work), because we think it is more intuitive than other works, such as the framework of Ramchurn et al. [33]. However, as we showed in Section 2, the idea of modelling the argumentation process as a planning problem is general, and we can define the actions of the planner in another way.

As regards argument selection, several works take into account different factors: the argument strength [23], the trust in its opponents [33, 34], the utility expected [33], the data confidence [4], authority roles [42], among others. In this respect, our work presents a unified treatment of these factors by using preference of the planning algorithm.

Finally, with reference to integration negotiation-planning, it is worth mentioning the studies of Kraus et al. [23] and Sadri et al. [39]. In the first work, as we have detailed previously, the actions that the agent must negotiate arise from the planning, but the negotiation is not planned. In the second work, agents compare the cost of different alternative plans to present to their counterparts. The aim of this comparison is not to decide internally among different alternatives, but to compare proposals.

9. Conclusions and future work

We have presented an approach to integrate plans to argue during a negotiation process into the general planning. We explained how the argumentation process, which an agent develops during the negotiation, may be modelled as a planning problem in order to obtain an *argumentation plan*. We also described how the planner mechanisms can emulate the generation and selection of arguments. To this end, we have pointed out the importance of considering the agent's preferences about argument selection in the action selection mechanism implemented by the planning algorithm. Therefore, we proposed the use of a planning algorithm based on preferences as the key to argumentation plan construction.

Once we defined the construction of the argumentation plans, we presented a simple way to integrate it into the generation of general plans, which an agent must build in order to fulfil its goals. We showed that the negotiation and, in particular, the argumentation can be considered a key action in these plans.

Planning the argumentation is useful for several reasons. In our proposal, arguments are evaluated as a whole and not only individually, because the evaluation is done on argumentation plan instead of over each argument severally. This is important when the order in which the arguments are displayed influences the final result of the negotiation. In a similar way, the integration allows us to take into account the relations among the different negotiations, in particular the order restrictions among these (as shown in the Section 6). These order restrictions allow us to consider the topology of dependencies among the conflicts, which could not be considered if the different conflicts were treated independently.

Moreover, the agent can evaluate in advance the need to reach secondary agreements with another opponent, in order to achieve the final agreement. Thus, the planning stage comprises a more extensive vision of the problem.

Another important advantage of the integration negotiation-planning is the ability to decide in planning time which negotiation alternative is preferable. Since the negotiation context is composed of multiple agents, if the agent has two or more opponents with which it can negotiate a specific resource, it can predict which one it would be able to persuade more easily and at a lower cost. Similarly, the agent can decide which action or resource to negotiate if there are several alternatives that allow it to fulfil its goals. As shown by the experimental results, this fact improves the performance of the agent, especially when there are multiple action alternatives in the negotiation scenario.

Furthermore, the planning algorithm utilises the agent's preferences in order to select the best actions. This feature provides a useful versatility to the planning, because of the diversity of factors that influence the argumentation (such as the trust in the opponent) that can be modelled, in our proposal, as preferences about actions and goals.

Although building an integral general plan is more expensive than building a simple one, the computational complexity remains the same in both scenarios (for further details, see [17]). It is because the generation of argumentation plans is defined as a tradition planning problem as well.

A limitation of our proposal may involve the quality and amount of information that the agent needs in order to be able to build argumentation plans. However, we claim that the information needed to build argumentation plans is just the same information that the agent would need to negotiate traditionally. That is, no additional information is needed to build argumentation plans. If knowledge is partial and the agent cannot build argumentation plans, then the negotiation is conducted in the traditional way. In other words, in the worst case, the agent must negotiate traditionally, though the agent probably cannot generate good arguments if the information is insufficient. Moreover, we have shown how information uncertainty can be considered during the argument selection.

Representing the interaction between the agent and its opponents within the argumentation plan is one of the major difficulties that we can encounter. Even though we can predict the behaviour of the other agents on the basis of historic information, this prediction still remains probabilistic and prone to errors. Future work will concentrate on this point. We will extend this approach taking into account the possible responses of the opponents, such as attacks between arguments. These relations among arguments are defined in argumentation frameworks [12], and they are used in argument dialogues for negotiation [3], where two players interchange arguments (moves), one defending a position and the other making the counter-arguments or defeating. Further future research may also focus on building a negotiation agenda by taking advantage of the order relations among the different negotiation processes that arise within the general plan.

Notes

1. In this work, by *argumentation process* we mean the argumentation process carried out by an agent during a negotiation.
2. We do not use any specific formalism to represent the agent's mental state, but, for instance, BDI architecture might be used [36].
3. For the purpose of this paper, we have included acceptances of arguments as possible responses. However, we have not taken into account refusals, since it is not meaningful to generate arguments if we suppose that they will be rejected.
4. We only include the illocutions used in the planning problem definition. Other illocutions that are only used during the negotiation were omitted (i.e. request, propose, reject, withdraw).

References

[1] B. An, F. Douglis, F. Ye, Heuristics for negotiation schedules in multi-plan optimization, In: Proc. 7th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 551-558.

[2] L. Amgoud, Y. Dimopolous, P. Moraitis, A unified and general framework for argumentation-based negotiation, In: Proc. 6th Int. Joint Conf. on Autonomous Agents and Multi-Agents Systems, AAMAS'2007, Honolulu, Hawaii, 2007, pp. 1-8.

[3] L. Amgoud, S. Parsons, N. Maudet, Arguments, dialogue, and negotiation, In: Proc. 14th European Conference on Artificial Intelligence, ECAI'00, Berlin, Germany, 2000, pp. 338-342.

[4] L. Amgoud, H. Prade, Generation and evaluation de different types of arguments in negotiation, In: Proc. Int. Workshop on Non-monotonic Reasoning, Whistler BC, Canada, 2004, pp. 10-15.

[5] L. Amgoud, H. Prade, Handling threats, rewards and explanatory arguments in a unified setting, Int. Journal of Intelligent Systems, 20 (12) (2005), pp. 1195-1218.

[6] R. Ashri, I. Rahwan, M. Luck, Architectures for negotiating agents, In: V. Marik, J. Mueller, M. Pechoucek (Eds), Proc. Multi-Agent Systems and Applications III, Prague, Czech Republic, 2003, pp. 136-146.

[7] J. A. Baier, S. A. McIlraith, Planning with preferences, AI Magazine, 29 (4) (2008), pp. 25-36.

[8] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi-agent systems with a FIPA-compliant agent framework, Software - Practice and Experience, 31 (2001), pp. 103-128.

[9] L. Berdún and A. Amandi, Planning for intelligent agents. In Proc. of the 7º Argentine Symposium on Artificial Intelligence, Rosario, Argentina, 2005, pp 12-23.

[10] A. Casali, L. Godo and C. Sierra, g-BDI: a graded intensional agent model for practical reasoning. In Proc. 6th MDAI, Awaji Island, Japan, 2009, pp 5-20.

[11] D.K. Chiu, S. Cheung, P.C. Hung, S.Y. Chiu and A.K. Chung, Developing e-negotiation support with a meta-modeling approach in a web services environment, Decision Support Systems, 40 (2005), pp. 51–69.

[12] P. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence, 77 (2) (1995), pp. 321–357.

[13] S. Fatima, M. Wooldridge, N. Jennings, An agenda-based framework for multi-issue negotiation. Artificial Intelligence, 152 (2004), pp. 1-45.

[14] D. Field, A. Ramsay, Sarcasm, deception, and stating the obvious: planning dialogue without speech acts, Artificial Intelligence Reviews, 22 (2) (2004), pp. 149–171.

[15] R. E. Fikes, N. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence, 5 (2) (1971), pp. 189-208.

[16] R. Freedman, Using a reactive planner as the basis for a dialogue agent, In: Proc. of the 13th Int. Florida Artificial Intelligence Research Society Conference, Orlando, USA, 2000, pp. 203–208.

[17] M. Ghallab, D. Nau, P. Traverso, Automated planning, theory and practice, Morgan Kaufmann Publishers Inc., 2004.

[18] M. M. Geipel, G. Weiss, A Generic Framework for Argumentation-Based Negotiation, In: M. Klusch, K. V. Hindriks, M. P. Papazoglou, L. Sterling (Eds.), Proc. 11th International Workshop on Cooperative information Agents, XI LNAI, 4676. Springer-Verlag, Berlin, Heidelberg, 2007, pp. 209-223.

[19] R. H. Guttman, P. Maes, Agent-mediated integrative negotiation for retail electronic commerce. In: Proc. Workshop on Agent Mediated Electronic Trading, 1998, pp. 70-90.

[20] N. R. Jennings, S. Parsons, C. Sierra, P. Noriega, On argumentation-based negotiation, In: Proc. Int. Workshop on Multi-Agent Systems, Boston, USA, 1998, pp. 1–7.

[21] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge, Automated negotiation: prospects, methods and challenges, Group Decision and Negotiation, 10 (2) (2001), pp. 199–215.

[22] G. E. Kersten, S. J. Noronha, WWW-based negotiation support: design, implementation, and use, Decision Support Systems, 25 (2) (1999), pp. 135-154.

[23] S. Kraus, K. Sycara, A. Evenchik, Reaching agreements through argumentation: a logical model and implementation, Artificial. Intelligence 104 (1-2) (1998), pp. 1-69.

[24] P. Maes, Agents that reduce work and information overload. Commun. ACM, 37 (7) (1994), pp. 30-40.

[25] P. Maes, R. Guttman, A. Moukas, Agents that buy and sell. Commun. ACM, 42 (3) (1999), pp. 81–ff.

[26] A. Monteserin, A. Amandi, Learning argument selection preferences in argumentation-based negotiation In: Proc. IADIS Int. Conf. on Intelligent Systems and Agents 2010, Freiburg, Germany, 2010, pp. 27-34.

[27] S. Park, G. E. Bolton, L. Rothrock, J. Brosig, Towards an interdisciplinary perspective of training intervention for negotiations: Developing strategic negotiation support contents, Decision Support Systems, 49 (2) (2010), pp. 213-221.

[28] S. Parsons, N. R. Jennings, Negotiation through argumentation: a preliminary report, In: Proc. 2nd Int. Conf. on Multi-Agent Systems, Kyoto, Japan, 1996, pp. 267-274.

[29] S. Parsons, P. McBurney, Argumentation-based dialogues for agent co-ordination, Group Decision and Negotiation, 12 (5) (2003), pp. 415-439.

[30] S. Parsons, C. Sierra, N. R. Jennings, Agents that reason and negotiate by arguing, Journal of Logic and Computation, 8 (3) (1998), pp. 261-292.

[31] I. Rahwan, S. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, L. Sonenberg, Argumentation-based negotiation, The Knowledge Engineering Review, 18 (4) (2003), pp. 343-375.

[32] I. Rahwan, L. Sonenberg, P. McBurney, Bargaining and argument-based negotiation: some preliminary comparisons. In: I. Rahwan, P. Moraitis, C. Reed (Eds.), Argumentation in Multi-Agent Systems: Proc. 1st Inter. Workshop, ArgMAS'04: Expanded and Invited Contributions, LNAI, 3366, Springer-Verlag, Berlin, Germany, 2005, pp. 176-191.

[33] S. Ramchurn, N. R. Jennings, C. Sierra, Persuasive negotiation for autonomous agents: a rhetorical approach, In: C. Reed, F. Grasso, G. Carenini (Eds.), Proc. IJCAI Workshop on Computational Models of Natural Argument, Acapulco, Mexico, 2003, pp. 9–17.

[34] S. Ramchurn, C. Sierra, L. Godo, N. R. Jennings, A computational trust model for multi-agent interactions based on confidence and reputation, In: Proc. workshop on Deception, Fraud and Trust in Agent Societies, Melbourne, Australia, 2003, pp. 69–75.

[35] S. Ramchurn, C. Sierra, L. Godo, N. R. Jennings, Negotiating using rewards, In Proc. 5th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems, AAMAS'06, ACM, New York, NY, 2006, pp. 400-407.

[36] A. S. Rao, M. Georgeff, BDI agents: from theory to practice, In: Proc. 1st Inter. Conf. on Multi-Agent Systems, 1995, pp. 312–319.

[37] C. Reed, D. Long, M. Fox, An architecture for argumentative dialogue planning, In: Proc. Inter. Conf. on Formal and Applied Practical Reasoning, London, UK, 1996, pp. 555–566.

[38] S. Rueda, A. García, G. Simari, Argument-based negotiation among BDI agents. Computer Science & Technology, 2(7) (2002), pp. 1-8.

[39] F. Sadri, F. Toni, P. Torroni, Dialogues for negotiation: agent varieties and dialogue sequences. In: J. Meyer, M. Tambe (Eds.), Intelligent Agent Series VIII: Proc. 8th Inter. Workshop on Agent Theories, Architectures and Languages, Lecture Notes in Computer Science, 2333, Springer-Verlag, 2001, pp. 69–84.

[40] M. Schroeder, An efficient argumentation framework for negotiating autonomous agents, in Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, London, UK, 1999, pp. 140–149.

[41] M. Schoop, A. Jertila, T. List, Negoisst: a negotiation support system for electronic business-to-business negotiations in e-commerce, Data & Knowledge Engineering, 47 (3) (2003), pp. 371-401,

[42] C. Sierra, N. R. Jennings, P. Noriega, S. Parsons, A framework for argumentation-based negotiation, In: Proc. 4th Inter. Workshop on Agent Theories, Architectures and Languages, Rode Island, USA, 1998, pp. 177-192.

[43] S. Sohrabi, J. Baier, S. A. McIlraith, HTN Planning with preferences, In: Proc. 21th Inter. Joint Conf. on Artificial Intelligence (IJCAI), Pasadena, California, 2009, pp. 1790-1797.

[44] T. C. Son, E. Pontelli, Planning with preferences using logic programming. Theory Pract. Log. Program. 6 (5) (2006), pp. 559-607.

[45] Y. Tang, S. Parsons, Argumentation-based dialogues for deliberation, In: Proc. 4th Inter. Conf. on Autonomous Agents and Multi-Agent Systems, The Netherlands, 2005, pp. 552–559.

[46] J. Wainer, P. R. Ferreira Jr., E. R. Constantino, Scheduling meetings through multi-agent negotiations, Decision Support Systems, 44 (1) (2007), pp. 285-297.

[47] H. Wang, S. Liao, L. Liao, Modeling constraint-based negotiating agents, Decision Support Systems, 33 (2) (2002), pp. 201-217.

[48] D. Weld, An introduction to least commitment planning, AI Magazine, 15 (1994) pp. 27-61.

Appendix A: Additional actions for argument generation

- Action: *createPastPromiseAppeal(X, Y, Action)*
  Description: it is used to generate appeals to past promises that the opponent did not fulfil.
  Preconditions: *iam(X), isagent(Y), pastpromise(Y, X, do(Y, Action))*
  Effects: *appeal(X, Y, do(Y, Action), [pastpromise(Y, X, do(Y, Action))])*
- Action: *createSelfInterestAppeal(X, Y, Action, Goal)*
  Description: it allows the agent to generate appeals to self-interest where the proposed action implies a profit to *Y*.
  Preconditions: *iam(X), isagent(Y), isgoal(Y, Goal), believe(X, imply(Action, Goal)), cando(Y, Action)*
  Effects: *appeal(X, Y, do(Y, Action), [believe(X, imply(Action, Goal)), isgoal(Y, Goal)])*
- Action: *createPrevailingPracticeAppeal(X, Y, Action, Goal)*
  Description: it proposes an action execution using as justification historic information about a third agent. Agent *Y* believes that if it performs *Action*, it should not fulfil *Goal*, but agent *X* has historic evidence (*wasgoal()* and *did()*) that deny this belief.
  Preconditions: *iam(X), isagent(Y), isagent(Z), isgoal(Y, Goal), believe(Y, imply(Action, not Goal)), wasgoal(Z, Goal), did(Z, Action)*
  Effects: *appeal(X, Y, do(Y, Action), [wasgoal(Z, Goal), did(Z, Action)])*
- Action: *createCounterexampleJustificationAppeal(X, Y, Action, Goal)*
  Description: it is similar to the previous appeal, but the historic information is about *Y*.
  Preconditions: *iam(X), isagent(Y), believe(X, fulfilled(Y, Goal, Action))*
  Effects: *appeal(X, Y, believe(Y, imply(Action, Goal)), [fulfilled(Y, Goal, Action)])*
- Action: *createTransitiveJustificationAppeal(X, Y, A, B, C)*
  Description: it creates an appeal that shows relations that the opponent disclaims. So, it is possible to find the existing relations between actions and goals in order to be used by other actions.
  Preconditions: *iam(X), isagent(Y), believe(Y, imply(A, B)), believe(Y, imply(B, C))*
  Effects: *appeal(X, Y, believe(Y, imply(A, C)), [imply(A, B), imply(B, C)])*
- Action: *createTrivialJustificationAppeal(X, Y, Alpha)*
  Description: it is the simplest justification. *X* leads *Y* to believe a self belief.
  Preconditions: *iam(X), isagent(Y), believe(X, Alpha)*
  Effects: *appeal(X, Y, believe(Y, Alpha), [believe(X, Alpha)])*
- Action: *createRewardBoth(X, Y, ActionR, ActionP, Goal)*
  Description: *X* proposes to *Y* the execution of *ActionP* in exchange for execution of *ActionR*. *Goal* is shared by *X* and *Y*.
  Preconditions: *iam(X), isagent(Y), isgoal(X, Goal), isgoal(Y, Goal), believe(Y, imply(ActionR, Goal)), cando(Y, ActionP), cando(X, ActionR)*
  Effects: *reward(X, Y, do(Y, ActionP), [do(X, ActionR)])*

Agent: *Picty*
Facts about itself: *iam(picty).*
Goals: *isgoal(picty, pictureHanging(p1)).*
Beliefs:
– *believe(picty, imply([has(picty, hammer(h1)), has(picty, nail(n1)), has(picty, chair(bc1)), has(picty, picture(p1))], do(picty, hangAPicture(picty, p1, h1, n1, bc1)))).*
– *believe(picty, imply(do(picty, hangAPicture(picty, p1, h1, n1, bc1)), pictureHanging(p1))).*
– *believe(picty, imply([has(mirry, mirror(m1)), has(mirry, screwdriver(sd1)), has(mirry, screw(s1))], do(mirry, hangAMirror(mirry, m1, sd1, s1)))).*
– *believe(picty, imply(do(mirry, hangAMirror(mirry, m1, sd1, s1)), mirrorHanging(m1))).*
– *believe(mirry, imply([has(mirry, mirror(m1)), has(mirry, hammer(h1)), has(mirry, nail(n1))], do(mirry, hangAMirror(mirry, m1, h1, n1)))).*
– *believe(mirry, imply(do(mirry, hangAMirror(mirry, m1, h1, n1)), mirrorHanging(m1))).*
– *believe(picty, imply(do(picty, giveResourceTo(picty, mirry, hammer(h1))), not(pictureHanging(p1)))).*
– *believe(mirry, imply(do(mirry, giveResourceTo(mirry, picty, nail(n1))), not(mirrorHanging(m1)))).*
– *believe(chairy, imply(do(chairy, fixAChair(chairy, bc1, g1)), has(chairy, chair(bc1)))).*
– *believe(mirry, imply(do(mirry, giveResourceTo(mirry, chairy, glue(g1))), not(has(mirry, glue(g1))))).*
Facts about opponents:
– *isgoal(mirry, mirrorHanging(m1)).*
– *isgoal(chairy, has(chairy, chair(bc1))).*
– *isgoal(mirry, has(mirry, glue(g1))).*
– *prefer(mirry, mirrorHanging(m1), has(mirry, glue(g1))).*
Agent: *Mirry*
Facts about itself: *iam(mirry).*
Goals:
– *isgoal(mirry, mirrorHanging(m1)).*
– *isgoal(mirry, has(mirry, glue(g1))).*
– *prefer(mirry, mirrorHanging(m1), has(mirry, glue(g1))).*
Beliefs:
– *believe(mirry, imply([has(mirry, mirror(m1)), has(mirry, hammer(h1)), has(mirry, nail(n1))], do(mirry, hangAMirror(mirry, m1, h1, n1)))).*
– *believe(mirry, imply(do(mirry, hangAMirror(mirry, m1, h1, n1)), mirrorHanging(m1))).*
– *believe(mirry, imply(do(mirry, giveResourceTo(mirry, picty, nail(n1))), not(mirrorHanging(m1)))).*
– *believe(mirry, imply(do(mirry, giveResourceTo(mirry, chairy, glue(g1))), not(has(mirry, glue(g1))))).*
Facts about opponents:
– *isgoal(picty, pictureHanging(p1)).*
– *isgoal(chairy, has(chairy, chair(bc1))).*
Agent: *Chairy.*
Facts about itself: *iam(mirry).*
Goals: *isgoal(chairy, has(chairy, chair(bc1))).*
Beliefs:
– *believe(chairy, imply([has(chairy, brokenChair(bc1)), has(chairy, glue(g1))], do(chairy, fixAChair(chairy, bc1, g1)))).*
– *believe(chairy, imply(do(chairy, fixAChair(chairy, bc1, g1)), has(chairy, chair(bc1)))).*
Facts about opponents:
– *isgoal(picty, pictureHanging(p1)).*
– *isgoal(mirry, mirrorHanging(m1)).*
– *isgoal(mirry, has(mirry, glue(g1))).*
– *prefer(mirry, mirrorHanging(m1), has(mirry, glue(g1))).*

Table 1. *M*ental states of the agents *Picty*, *Mirry* and *Chairy*.

#g_i	***Picty_arg***					***Picty_sim***				
	T	***#I***	***#P***	***#Rp***	***#M***	***T***	***#I***	***#P***	***#Rp***	***#M***
1	01'04"	4	1	0	53	00'26"	7	2	0	95
2	01'18"	4	1	0	67	00'57"	10	2	1	123
3	01'25"	4	1	0	65	01'39"	16	3	2	165
4	01'38"	4	1	0	69	02'09"	19	3	3	169
5	02'10"	4	1	0	81	02'43"	19	2	4	168
6	02'36"	4	1	0	86	03'42"	22	2	5	181
7	02'58"	4	1	0	81	05'09"	25	2	6	197
8	03'23"	4	1	0	85	06'30"	28	2	7	211
9	04'04"	4	1	0	89	08'47"	31	2	8	221
10	04'53"	4	1	0	93	11'21"	34	2	9	231

Table 4. Comparison of *Picty_arg* and *Picty_sim* performances.

Figure 1. Graphical representation of an argumentation process..

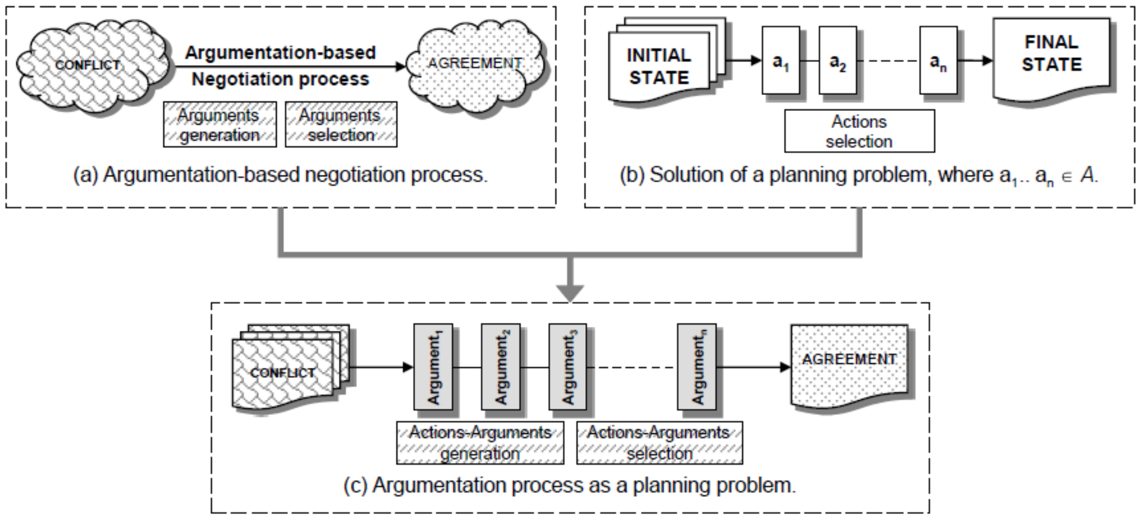

CONFLICT
Argumentation-based Negotiation process
AGREEMENT
Arguments generation
Arguments selection
(a) Argumentation-based negotiation process.
INITIAL STATE
a_1
a_2
a_n
FINAL STATE
Actions selection
(b) Solution of a planning problem, where $a_1 .. a_n \in A$.
CONFLICT
$\mathrm{Argument}_1$
$\mathrm{Argument}_2$
$\mathrm{Argument}_3$
$\mathrm{Argument}_n$
AGREEMENT
Actions-Arguments generation
Actions-Arguments selection
(c) Argumentation process as a planning problem.

Figure 2. From rules to actions for argument generation.

IF
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) Informal rule for argument generation based on the framework of Kraus et al., 1998.

```
action(NAME(PARAMETERS),
    [PRECONDITIONS],
    [EFFECTS]).
```

(b) General specification of an action in planning algorithms.

```
action(createCounterexampleAppeal(X, Y, ActionA, Goal),
    [i_am(X), is_agent(Y), is_goal(Y, Goal), did(Y,
      ActionB), believe(Y, imply(ActionB, not(Goal)))],
    [appeal(X, Y, do(Y, Action), [did(Y, ActionB),
      imply(ActionB, not(Goal))])]).
```

(c) Specification of a rule for argument generation as an action in a planning algorithm..

Figure 3. Argumentation plan.

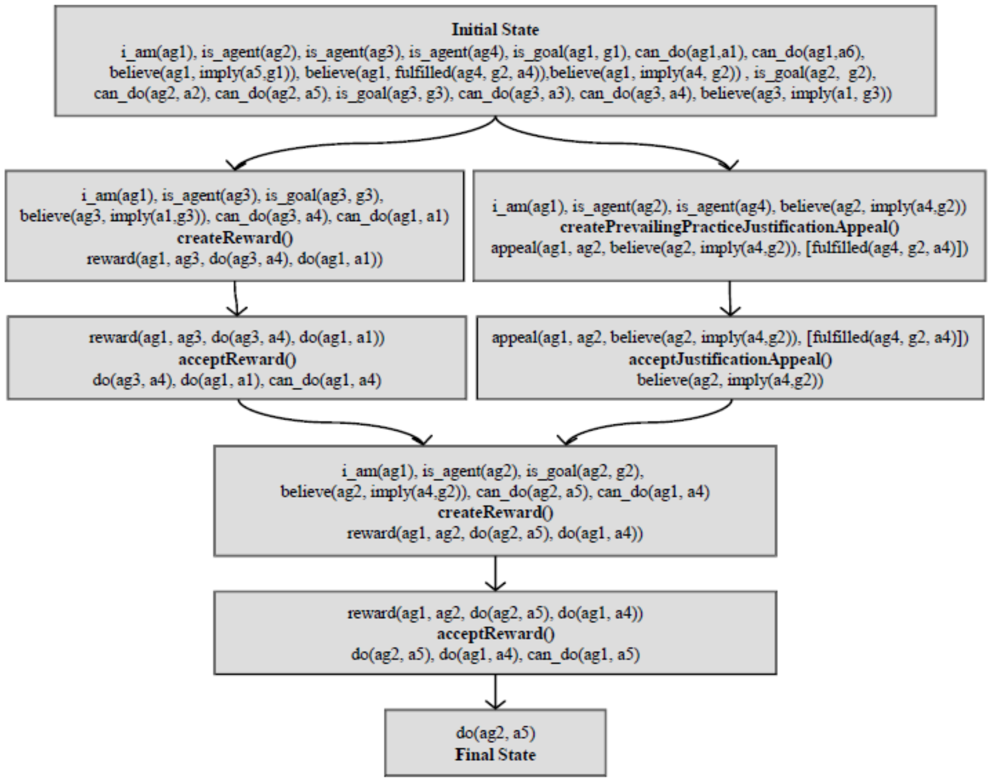

Initial State
i_am(ag1), is_agent(ag2), is_agent(ag3), is_agent(ag4), is_goal(ag1, g1), can_do(ag1,a1), can_do(ag1,a6),
believe(ag1, imply(a5,g1)), believe(ag1, fulfilled(ag4, g2, a4)),believe(ag1, imply(a4, g2)) , is_goal(ag2, g2),
can_do(ag2, a2), can_do(ag2, a5), is_goal(ag3, g3), can_do(ag3, a3), can_do(ag3, a4), believe(ag3, imply(a1, g3))
i_am(ag1), is_agent(ag3), is_goal(ag3, g3),
believe(ag3, imply(a1,g3)), can_do(ag3, a4), can_do(ag1, a1)
createReward()
reward(ag1, ag3, do(ag3, a4), do(ag1, a1))
i_am(ag1), is_agent(ag2), is_agent(ag4), believe(ag2, imply(a4,g2))
createPrevailingPracticeJustificationAppeal()
appeal(ag1, ag2, believe(ag2, imply(a4,g2)), [fulfilled(ag4, g2, a4)])
reward(ag1, ag3, do(ag3, a4), do(ag1, a1))
acceptReward()
do(ag3, a4), do(ag1, a1), can_do(ag1, a4)
appeal(ag1, ag2, believe(ag2, imply(a4,g2)), [fulfilled(ag4, g2, a4)])
acceptJustificationAppeal()
believe(ag2, imply(a4,g2))
i_am(ag1), is_agent(ag2), is_goal(ag2, g2),
believe(ag2, imply(a4,g2)), can_do(ag2, a5), can_do(ag1, a4)
createReward()
reward(ag1, ag2, do(ag2, a5), do(ag1, a4))
reward(ag1, ag2, do(ag2, a5), do(ag1, a4))
acceptReward()
do(ag2, a5), do(ag1, a4), can_do(ag1, a5)
do(ag2, a5)
Final State

Figure 4. Planning and negotiation.

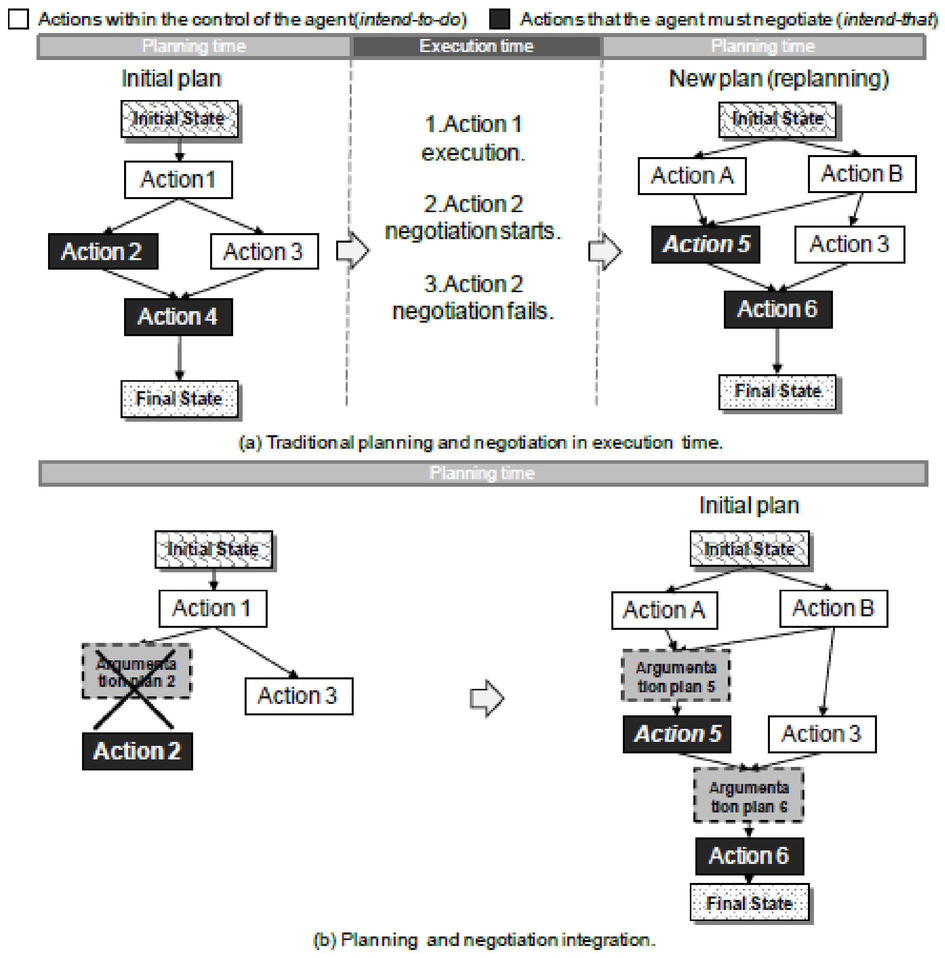


Figure 5. Figurative and real view of the integration.

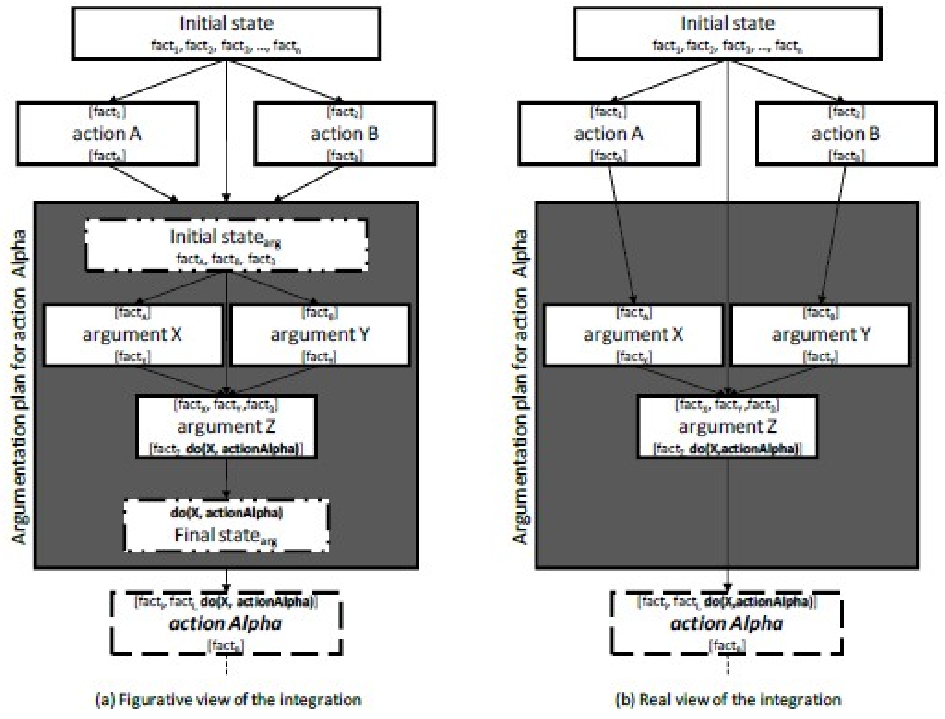


Figure 6. Algorithm to general plan construction.

```
context_change_flag  := TRUE;
WHILE ($\exists$ unachieved goals) DO
  IF (context_change_flag) THEN
     A := {Agent actions $\cup$ Argument actions};
     I := {Initial state};
     F := {Final state};
     P := {Action selection preferences};
     plan := plan_generation(A, I, F, P);
  ELSE
     plan := plan.replanning();
  END IF
  IF (plan <> NULL) THEN
     plan.execute();
  ELSE
     wait_for_context_change();
  END IF
  context_change_flag := check_change(A, I, F, P);
END WHILE
```

Figure 7. Picty's general plan.

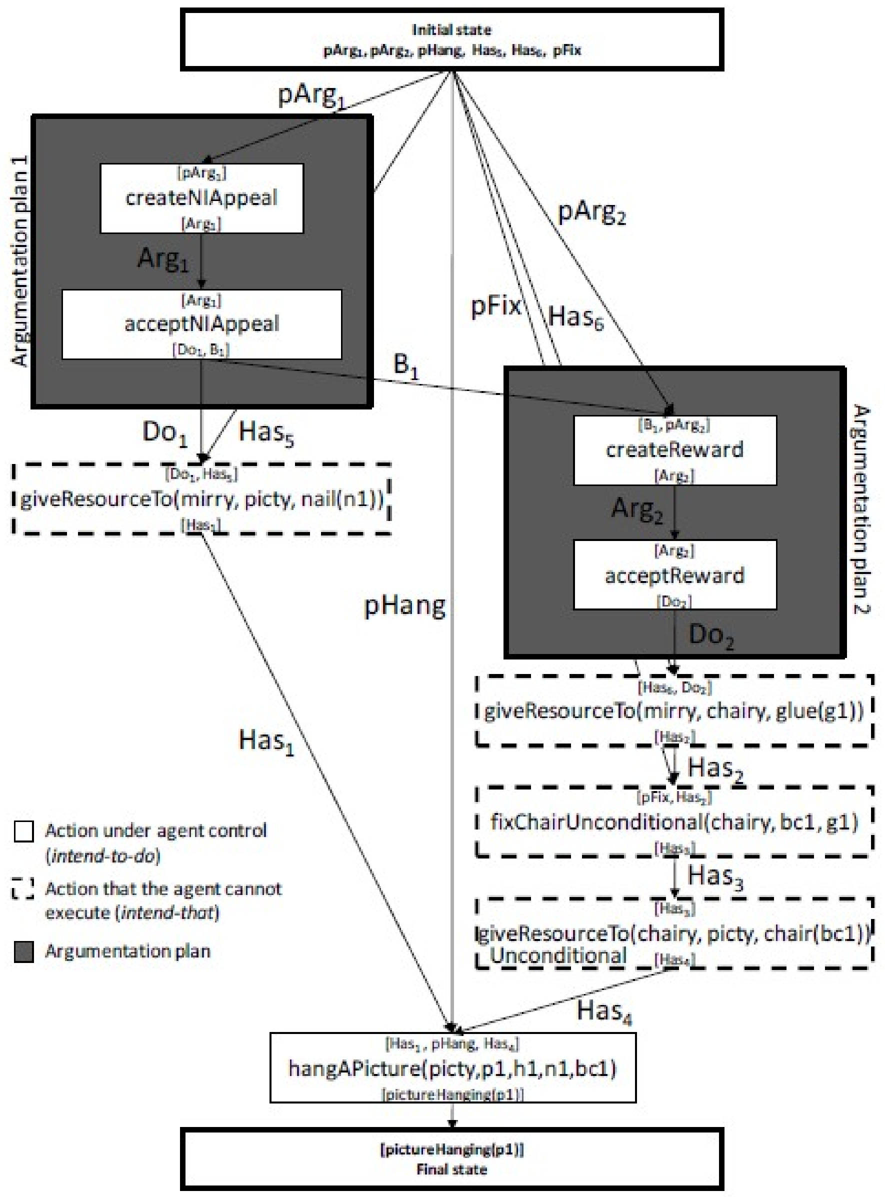
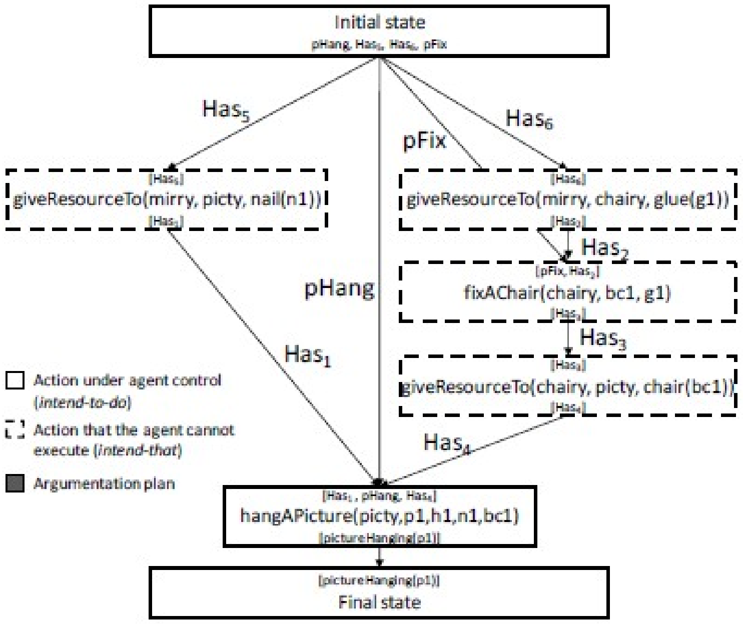
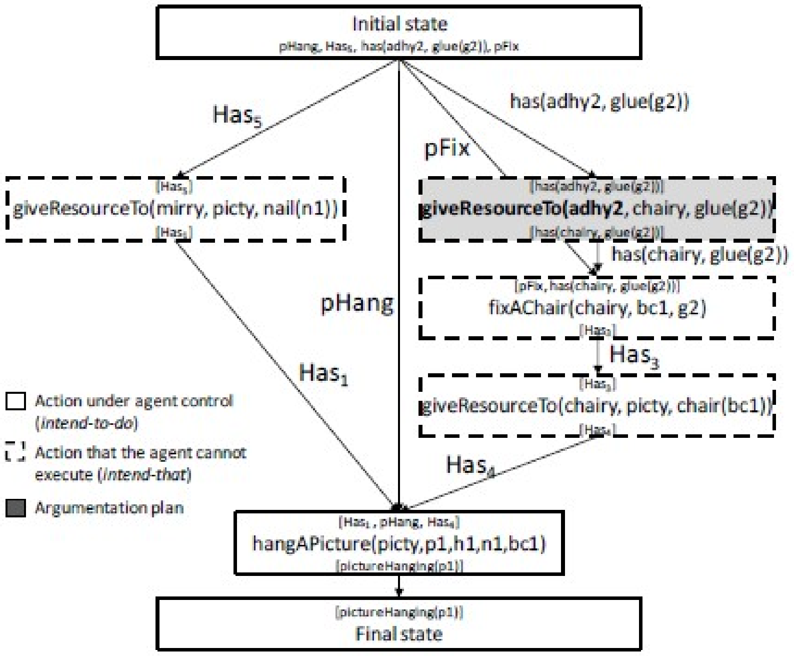

Initial state
pArg₁, pArg₂, pHang, Has₅, Has₆, pFix
pArg₁
Argumentation plan 1
[pArg₁] createNIAppeal [Arg₁]
Arg₁
[Arg₁] acceptNIAppeal [Do₁, B₁]
B₁
Do₁
Has₅
[Do₁, Has₅] giveResourceTo(mirry, picty, nail(n1)) [Has₁]
pArg₂
pFix
Has₆
Argumentation plan 2
[B₁, pArg₂] createReward [Arg₂]
Arg₂
[Arg₂] acceptReward [Do₂]
Do₂
pHang
[Has₆, Do₂] giveResourceTo(mirry, chairy, glue(g1)) [Has₂]
Has₂
[pFix, Has₂] fixChairUnconditional(chairy, bc1, g1) [Has₃]
Has₃
[Has₃] giveResourceTo(chairy, picty, chair(bc1)) Unconditional [Has₄]
Has₁
Has₄
Action under agent control (intend-to-do)
Action that the agent cannot execute (intend-that)
Argumentation plan
[Has₁, pHang, Has₄] hangAPicture(picty,p1,h1,n1,bc1) [pictureHanging(p1)]
[pictureHanging(p1)]
Final state

Figure 8. Plans of Picty_sim.

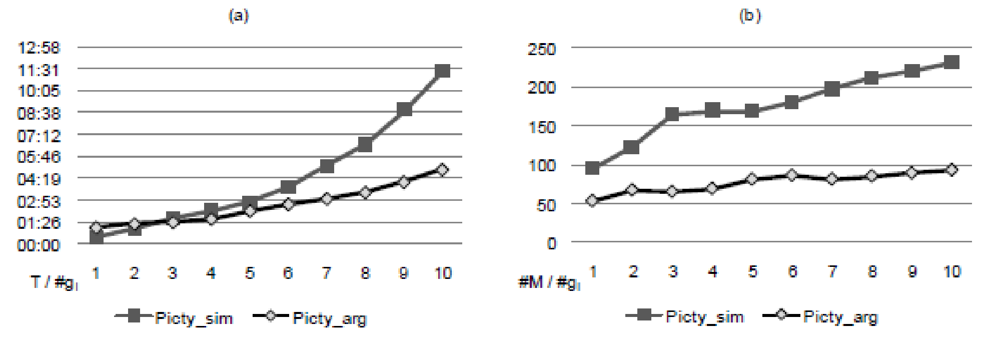

Initial state
pHang, Has5, Has6, pFix
Has5
pFix
Has6
[Has5]
giveResourceTo(mirry, picty, nail(n1))
[Has1]
[Has6]
giveResourceTo(mirry, chairy, glue(g1))
[Has2]
Has2
pHang
[pFix, Has2]
fixAChair(chairy, bc1, g1)
[Has3]
Has3
Has1
[Has3]
giveResourceTo(chairy, picty, chair(bc1))
[Has4]
Action under agent control (intend-to-do)
Action that the agent cannot execute (intend-that)
Argumentation plan
Has4
[Has1, pHang, Has4]
hangAPicture(picty,p1,h1,n1,bc1)
[pictureHanging(p1)]
[pictureHanging(p1)]
Final state

(a) Initial plan of *Picty_sim* with one resource *g*.


Initial state
pHang, Has5, has(adhy2, glue(g2)), pFix
has(adhy2, glue(g2))
Has5
pFix
[Has5]
giveResourceTo(mirry, picty, nail(n1))
[Has1]
[has(adhy2, glue(g2))]
giveResourceTo(adhy2, chairy, glue(g2))
[has(chairy, glue(g2))]
has(chairy, glue(g2))
[pFix, has(chairy, glue(g2))]
pHang
fixAChair(chairy, bc1, g2)
[Has3]
Has3
Has1
[Has3]
giveResourceTo(chairy, picty, chair(bc1))
[Has4]
Action under agent control (intend-to-do)
Action that the agent cannot execute (intend-that)
Argumentation plan
Has4
[Has1, pHang, Has4]
hangAPicture(picty,p1,h1,n1,bc1)
[pictureHanging(p1)]
[pictureHanging(p1)]
Final state

(b) Failed plan of *Picty_sim* with multiple resource *g*.

Figure 9. Comparative charts between Picty_sim and Picty_arg...



Author Bios

Ariel Monteserin is a teaching assistant in the Computer Science Department at Univ. Nac. del Centro de la Pcia. de Bs. As. (UNCPBA). He received his PhD in 2009. His main interests are negotiation and argumentation among intelligent agents.

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.