An Agent Architecture for Simultaneous Bilateral Negotiations*

Angela Fabregues and Carles Sierra

IIIA - Institut d'Investigació en Intel·ligència Artificial
CSIC - Spanish Scientific Research Council,
UAB, 08193 Bellaterra, Catalonia, Spain

Abstract. In this paper we introduce an agent architecture for joint action negotiation among several agents in complex environments and with negotiation time bounds. The architecture is based on a graded BDI model and on an information-based negotiation model. It is being tested using DipGame (`http://www.dipgame.org`).

1 Introduction

A lot of research on automated negotiations has been made in recent years but very little on negotiations with humans. Most of the papers on automated negotiation focus on bilateral negotiations where software agents compete using utility maximisation strategies. These strategies may work well when negotiating with software agents but not necessarily against humans that in many cases will not behave according to a constructivist rationality principle [2]. For instance, human agents' decisions depend a lot on e.g. the relationships with the other agents and on emotions. In this work we deal with simultaneous, repeated and bilateral negotiations in competitive environments. The other agents can be human or software. And they are able to co-operate performing joint plans to get a better outcome. Here, as we will see, *good* does not exclusively mean obtaining high utility but also to obtain other desired features.

We assume that agents are negotiating over large spaces of potential joint plans, in a limited amount of time and with limited resources. These are the usual settings that humans are faced with in their everyday life. Humans do not attempt to find the optimal solution before starting negotiating with others. People deal with uncertain environments and do not hesitate to negotiate plans that are not optimal but simply good enough. Usually, it is not possible to get the optimal one because of time constraints and because of the competitive simultaneous negotiation setting: if you don't close a deal quickly, you may not be able to close any deal at all.

The architecture proposed in this paper is specially designed for environments where several self-interested agents compete periodically performing actions in specific time instants. All the agents perform the actions at the same time. Actions change the state of the world but not all the actions are successfully executed, i.e. there is uncertainty on whether an action will be successful. Actions of all the agents are executed concurrently, therefore conflicts arise among actions and some of them may fail.

* Paper already published: [1].

Before selecting what actions to perform, agents interact with the aim of deciding joint plans, that is, to co-operate. Agents are assumed to be competitive but they are also interested in increasing their confidence on the other agents' next actions. This is because knowing what the other agents are going to do allows us to choose actions that avoid undesired conflicts. Getting from others this information and deciding joint plans is possible thanks to negotiation. In the environments we envisage an agent negotiates trying to convince the others to perform the set of actions that it desires them to perform. There would be few actions to execute but a lot of negotiation to decide them.

This paper presents ongoing work and there are not experimental results yet. We plan to test the architecture using the DipGame testbed (`http://www.dipgame.org`) against human and software players. DipGame provides an environment that is complex enough for testing our negotiation model.

In this work we first shortly introduce, as background material, the g-BDI and LOGIC (Section 2) models that we incorporate as components of our architecture. We then describe the architecture (Section 3), the action planning (Section 4), and the negotiation (Section 5) components. The paper ends with a discussion on future work (Section 6).

2 g-BDI and LOGIC

Taking a look into the literature we see that one of the most popular agent models is the BDI model [3–5]. BDI architectures have Believes, Desires and Intentions as their principal components. The BDI model has been used to cope with complex, dynamic and uncertain environments.

We use a graded BDI model, g-BDI [6], in order to describe the mental state of the agent. We want to build agents capable of interacting with humans. And human agents are really good working in uncertain environments. We argue that this graded BDI model is very useful when dealing with complex environments where the world is not perfectly observable as these we are considering here (e.g. the outcome of an action is uncertain).

The other model that we base our work on is the LOGIC negotiation model [7]. This work structures the information that we need in order to be able to negotiate with humans in repeated encounter scenarios. Information is organised along five dimensions: **Legitimacy**, that is, the relevant information that might be useful to the other agent in the negotiation, **Options**, that is, the deals that are acceptable, **Goals**, that is, the objectives of the negotiation, **Independence**, that is, the outside options[1] and **Commitments**, that is, previously signed agreements.

We use this model but not exactly as it is described in [7]. The LOGIC dimensions are assumed to be prepared before every single negotiation with an agent starts. Then, the negotiating agents exchange several proposals until a deal is accepted or someone withdraws. We do this differently. We do not select the negotiation partner beforehand. We maintain the information on the five dimensions updated on real time and negotiate with one or another agent depending only on the varying information content along the

[1] What can we do if the negotiation fails.

five dimensions. In this way, the partnership selection is intermingled with the selection of the next option to negotiate. We do so as our architecture needs to allow for multiple bilateral negotiations. We are also interested in the way that LOGIC proposes to evaluate the relationships between the different agents.

3 Architecture

In our architecture the environment, the world and the agents, are perceived via an interface module, see Figure 3. The messages that the agents send to us, the actions that they try to perform and the state of the world are observed through it. These perceptions are stored in a database (History) that updates the beliefs of the agents and provides data for decision making.

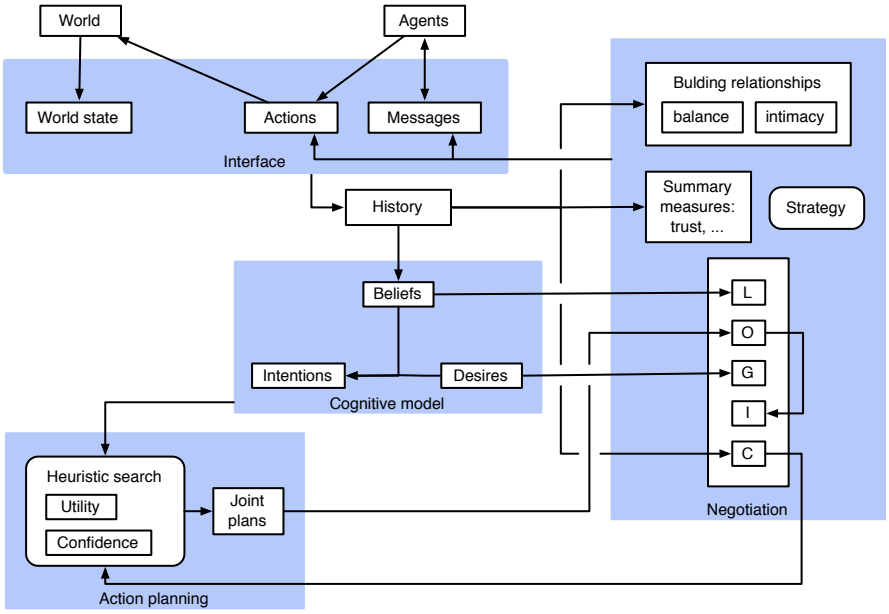

Fig. 1. Agent architecture. White boxes represent data or concepts, boxes with round corners represent processes and arrows represent data flows.

Beliefs are updated via an interpreter that is a mechanism capable of inferring facts based on the information that is stored in the history. These facts with their corresponding degree are the beliefs of the agent. The interpretation of the environment observations (i.e. the perception) depends mostly on the agent itself. We assume that an agent has a set of local rules to interpret the dialogical actions. The rules determine updates in the beliefs model of the agent. The schema of those rules is: State(σ) AND Message(μ) THEN Update($\mathcal{B}(\varphi)$) and the intended meaning of a rule is: from the current environment and internal states (σ) and a perceived message (μ) then update the beliefs model

$\mathcal{B}(\varphi)$. Every agent will then possess a theory consisting of a number of such rules that will update, for instance, the agent's model of the behaviour of other agents. Beliefs are one of the components of the agent's cognitive model together with desires and intentions as our architecture is based on a graded BDI model [6]. The agent's cognitive model provides data for the planning of actions and for the negotiation module. The behaviour of the agent depends on the dynamics of beliefs and desires.

In order to be able to decide what to do next the agent must look for action alternatives and build plans where, either the agent alone or with the help of others, the agent can get a good enough outcome in a particular world state. This action planning is done in our architecture with a heuristic search that is explained in section 4 and that provides a set of joint plans.

Any *joint* plan includes actions to be performed by other agents. Our certainty on the selection of that particular action by the other agent may be low. To increase it, agents negotiate proposing joint plans that may make the selection of those particular actions more attractive to the other agent. To select attractive proposals, the negotiation module organises the information into five different dimensions as described by the LOGIC negotiation model [7]. The *legitimacy* dimension retrieves from the beliefs the relevant information for the currently open negotiation. The *options* are the deals that can be built from the joint plans. The *goals* are the desires of the agents and the *independence* is a measure of the options that are currently available. Finally, the *commitments* are the previous promises to act still active. They are retrieved from the previous signed deals contained in the History. The negotiation strategy cares about building relationships for future negotiations and uses the model of the other agents behaviour to select the next message. See section 5 for further details.

This architecture has been built to be used in complex and very competitive environments. In those environments, agents are not able to wait until the optimal solution is found. They have to interact with other agents from the very beginning, and make proposals of deals on joint plans while they decide which plans are the best ones. Thus, the execution of an agent consists on several concurrent processes for: the interface (to receive messages, observe the performed actions and the world state and store them in the history), the interpreter that updates the agent model from the changes in the history, action planning and for the negotiation module (selecting proposals to make and actions to perform). The data in the boxes is constantly being updated by these processes. But the negotiation process gets idle regularly when a proposal is sent to another agent as we want to have only one proposal open at anytime. This simplifying restriction will be removed in future versions of the agent implementation. The idle time gets over when an answer from the appropriate agent is received or too much time has passed without any answer. See Algorithm 1 for a multithreaded version of the agent. The meaning of the different variables will become clear when reading the subsequent sections.

The algorithm consists on initially spawning all the processes except for the negotiation one. And then, repeat a sequence of: a number of negotiation rounds (modelled as function $negotiation()$) up to the time limit, a selection of actions from the set of agreed upon joint plans and their execution. The negotiation function starts waiting. The action planning execution needs some time to get enough plans to feed with options the negotiation process and time must be also given to other agents to analyse our proposals

Algorithm 1 function init(t_{max})

Require: **shared** P^t {current plans}
Require: **shared** $\delta = \bot$ {current proposed deal}
Require: **shared** $response = \bot$ {received response to δ}
Require: **shared** $patience = default$ {patience}
  spawn $pr_1 ... pr_n$ {processes associated to the architecture}
  while $\top$ **do**
    $t_{limit} \leftarrow t_{now} + t_{max}$
    while $t_{now} < t_{limit}$ **do**
      $negotiation()$
    end while
    $Actions \leftarrow selectActions(P^t)$
    $perform(Actions)$
  end while
  return

Algorithm 2 function negotiation()

  $waituntil(t_{now} = patience$ `or` $response \neq \bot)$
  if $\delta \neq \bot$ **then**
    if $response = \bot$ **then**
      $sendWithdraw(\delta)$
      $\delta \leftarrow \bot$
    else
      if $response = accept(\delta)$ **then**
        $Commitments \leftarrow Commitments \cup \{\delta\}$
        $response, \delta \leftarrow \bot$
      end if
    end if
  end if
  $Proposals' \leftarrow Proposals$
  $Proposals \leftarrow Proposals \setminus Proposals'$
  $Deals \leftarrow getProposalsToAccept(Proposals')$
  for $\delta' \in Deals$ **do**
    $sendAccept(\delta')$
  end for
  for $\delta' \in Proposals' \setminus Deals$ **do**
    $sendReject(\delta')$
  end for
  if $Deals = \emptyset$ **then**
    $\delta \leftarrow selectNextDeal(Plans)$
    if $\delta \neq \bot$ **then**
      $sendProposal(\delta)$
      $patience \leftarrow t_{now} + t_{max}^{\beta(\delta)}$
    end if
  end if
  return

after sending them. When new messages arrive, we check if it is a proposal. If it is, the message is stored in the set of proposals. This set is shared with the negotiating function that periodically checks this set to select a subset of proposals that can be jointly acceptable. The rest of proposals are rejected. If none of the proposals is good enough, a new deal is selected and the negotiation round is finished. Every proposal is stored in δ until an answer is received or a timeout fires. The negotiation function, Algorithm 2, waits for this answer. When it arrives, it stores it in the variable $response$ and the negotiation function resumes its execution by processing the answer.

Agents interact with the environment (performing actions) and also with other agents (sending and receiving messages). The *communication* module provides a language useful for formulating actions and also proposing deals. We assume that a common language is shared. We use the $\mathcal{L}_1$ language that is the first level of the language tower proposed in [8]. It allows agents to interact via exchanging, accepting and rejecting proposals, or to withdraw the negotiation, as defined in figures 2 and 3. In the subsequent sections we explain the components of the architecture and provide the equations for the functions that appear in the already introduced algorithms.

$\mathcal{L}_1$ ::= propose$(\alpha, \beta, deal)$ | accept$(\alpha, \beta, deal)$ |
reject$(\alpha, \beta, deal)$ | withdraw(α, β)
$deal$ ::= Commit$(\alpha, \beta, \varphi)^+$ | Agree(β, φ)
φ ::= $\underline{predicate}$ | Do$(\alpha, \underline{action})$ | $\varphi \wedge \varphi$ | $\neg\varphi$
β ::= α^+
α ::= $\underline{agent}$

Fig. 2. $\mathcal{L}_1$ language definition. Note that: $expression^+$ denotes a non-empty sequence of $expression$, non terminal symbols are written in italic, and undefined symbols (referring to terms in the ontology) appear in underlined italics.

4 Action Planning

The module of action planning, see Figure 3, provides a set of joint plans that can include actions to be performed by other agents. Those plans that involve actions for several agents require the collaboration among them and thus may trigger a negotiation process.

A basic plan is a tuple $\langle \alpha, a, t \rangle$ where α is an agent, a is an action and t is a time instant. The intuitive meaning is that agent α performs action a at time t. A plan is a set of such basic plans.The set of basic plans is denoted as B and the space of possible plans is $P = 2^B$. The empty set meaning inaction.

A plan is *unfeasible* if it has two or more incompatible actions. The incompatibility of actions depends on the application domain, e.g. in the Diplomacy game we cannot move the same unit to two different regions at the same time. Two feasible plans $p, p' \in P$ are incompatible if $p \cup p'$ is an unfeasible plan.

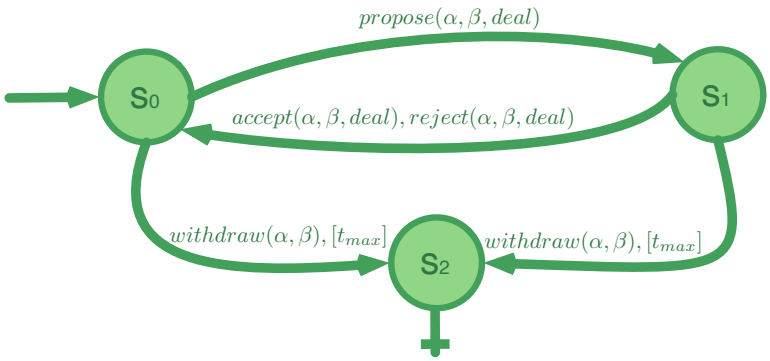

Fig. 3. $\mathcal{L}_1$ language protocol. Agents can offer, accept and reject proposals, or withdraw the negotiation. After some time t_{max} without communication the negotiation is assumed to finish. Note that the labels of the arrows are related to the sentences that can be formulated by the language defined in Figure 2.

We evaluate single plans based on their utility. We assume that there is a utility function $U : P \times W \rightarrow [0, 1]$ for plans and world states, and a confidence measuring the probability that a plan would be executed, $C : P \rightarrow [0, 1]$. Note that as some actions may depend on other agents we may be uncertain about them being actually performed.

Definition 1. *Given a plan $p \in P$ and a world state w, the* value *of plan p in world state w, $\mathcal{V}(p, w)$, is defined as the expected utility of the plan given the probability (confidence) of the plan being executed:*

$$\mathcal{V}(p, w) = E[\mathcal{U}(p, w)] = \mathcal{U}(p, w) \cdot \mathcal{C}(p) \tag{1}$$

The evaluation of the confidence and the utility depend a lot on the application domain and on the beliefs of the agent. We measure the confidence as the probability that we have on all agents involved in the plan performing the actions that the plan requires them to perform. That is, as our degree of belief on it (see [6] for details on the modelling of beliefs). If an agent thinks that it is not possible for a plan to be performed because the other agents are not going to do what we want to, then it will not take it into consideration and will not engage in negotiations over it, the agent will just look for another option.

The confidence level of a plan is usually in line with the trust on other agents and on the time horizon set by the latest basic plan in the plan. The more agents the plan depends upon, the lower the probability of success and the latter the lattest basic plan to be completed the more improbable the plan is as the probability of the environment changing (and thus making the actions unfeasible) in the meantime will be higher.

The utility of a plan is calculated based on the utility of the states of the world that we can get executing it and the probability, $\mathbb{P}$, on each of those states being reached.

Definition 2. *Given a plan $p \in P$ and a world state w, the* utility *of plan p in w, $U(p, w)$, is defined as the expected utility of the world states that we can reach performing p:*

$$\mathcal{U}(p, w) = \sum_{w_j \in W} \mathbb{P}(w_j|p, w) \times \mathcal{U}(w_j) \tag{2}$$

Both, the utility and the probability of reaching a world state ($\mathcal{U}(w_j)$ and $\mathbb{P}(w_j|p, w)$) depend on the application domain. In domains where the world is not perfectly observable and thus we are not sure about the current state nor about the state transition probabilities, we could use Partially Observable Markov Decision Processes (POMDPs). However, a wealth of data would be necessary to model it accurately. Even when the world is perfectly observable, we have to deal with uncertainty because we do not know what the other agents are going to do. And their actions will affect the world as well. The utility takes into account the uncertainty on the world. Contrarily, the confidence measures to what extend we are sure that the plan will be executed.

Definition 3. *Given a plan $p \in P$, the* confidence *of executing it, $\mathcal{C}(p)$, is defined as the degree of belief, r_b, on all its basic plans $b \in p$ being executed:*

$$\mathcal{C}(p) = \prod_{\substack{b \in p \\ \mathcal{B}(b, r_b)}} r_b \tag{3}$$

Before starting a negotiation process, our belief on a basic plan being executed can be low. An agent can negotiate to increase its confidence on it. This is done by signing deals that provide commitments on other agents actions. When an agent commits to the execution of a basic plan, this commitment is added to the set of current commitments, C and our belief on this basic plan being executed is the trust that we have on the agent [9]. But to decide whether to negotiate or not a deal on a basic plan, we can estimate the confidence that we would obtain after negotiating it.

Definition 4. *Given a basic plan $b = \langle \alpha, a, t \rangle$, the estimation of β's belief on its future execution $\mathcal{B}(b, r_b)$ if β engages in a negotiation, is defined as:*

$$r_b = \mathbb{P}(b|\neg\varphi) \times \mathbb{P}(\neg\varphi) + \mathcal{T}(\beta, \alpha, \varphi) \times \mathbb{P}(\varphi) \tag{4}$$

where $\varphi = Commit(\alpha, \beta, Do(\alpha, \langle a, t \rangle))$, $\mathcal{T}(\beta, \alpha, \varphi)$ is the trust that β has on α honouring the commitment, and $\mathbb{P}(\varphi)$ is the probability that α accepts the commitment.

As the search space for plans is exponential in the number of possible actions and the evaluation of an action may change a lot due to changes in the environment (world state) we have to work with good enough plans instead of looking for an optimal solution. In the kind of environments we are working on, we assume optimality is never achievable in practical terms. Therefore, the usual planning techniques do not provide good results. What we need is an action planning engine that would provide a set of feasible plans at any time taking into account the dynamics of our beliefs. It is important to have a rich set of solutions instead of just one as it makes the negotiation process more robust by providing several options for negotiation. To achieve this robustness,

we enforce by construction that the set of plans satisfy a trade-off between coherence and diversity. We want to have at any time as many good plans as possible but also we want coherence and diversity. Coherence means that the plans are somehow similar among them in order to leave several plans available as options when a particular subplan is successful negotiated, as incompatible options will be removed. Diversity means that plans are somehow dissimilar in order to leave several of them available as options when the negotiation on a particular subplan fails. We define the concept of similarity of two plans as the proportion of basic plans that they have in common.

Definition 5. *The similarity between two plans $p, q \in P$, is defined as the proportion of common basic plans:*

$$sim(p, q) = \frac{|p \cap q|}{|p \cup q|} \tag{5}$$

To define a degree of trade-off between coherence and diversity we use the Gini mean difference as a measure of dispersion that compares pairs of elements, in our case plans, of a set.

Definition 6. *Given the set of computed plans at time t, $P^t \subseteq P$, we define the* statistical dispersion *of P^t, $\Delta(P^t)$, as the Gini measure of the plans similarity.*

$$\Delta(P^t) = \frac{1}{|P^t|(|P^t| - 1)} \sum_{p,q \in P^t} sim(p, q) \tag{6}$$

Thus the higher the gini value, the higher the dispersion and the lower the gini the higher the coherence. The problem is that gini compares pairs of plans, not larger sets of them. Then it could happen that a set of plans with high dispersion could have a single basic plan appearing in all the plans of the set. This is not desirable because all the plans would depend on that single basic plan. This is a problem that could be solved applying a measure of dispersion not only to pairs but also to subsets of three, four or more plans. But the complexity of the resulting measure would be too high. Instead of this we use a measure of the popularity of the basic plans that we combine with the Gini dispersion measure. Then, under some threshold, the lower the maximum popularity of basic plans, the better.

Definition 7. *The* popularity *of a basic plan $b \in B$ within a set of plans P^t is the proportion of plans in P^t that contain it:*

$$pop(b, P^t) = \frac{|\{p \in P^t : b \in p\}|}{|P^t|} \tag{7}$$

The relevance of having a set of plans that is at the same time coherent and diverse will be better understood when introducing the concept of deal in the next section. In addition to looking for diversity of basic plans we could also look for diversity of negotiation partners (the agents involved in the basic plans), of actions or of time. Or whatever other dimension of a basic plan would like to define.

We use genetic algorithms (GA) to look for those plans. Genetic algorithms allow to efficiently explore large spaces of solutions. They produce successive populations of

solutions that are increasingly better adapted to the environment even when it is dynamically changing along time. For us, each single solution, a chromosome in the GA, represents a feasible plan. This population is updated generation after generation by the crossover, mutation and selection operators. Crossover and mutation must guarantee the feasibility of the generated plans.

We look for a set of plans being coherent and diverse, as explained before. We use an elitist selection over some individuals that will survive to the next generation. Elitism should be applied to the subset of the population that better satisfies a combination of factors: being coherent, diverse, having a minimum number of basic plans, and having substantial utility and confidence. To facilitate the coverage of the whole search space, also other individuals survive but with a probability of survival proportional to the value of their fitness. Finally, the rest of the members of the new population are generated with the crossover mechanism. A certain probability of mutation allows to explore plans containing basic plans that were not included in the initial population.

As explained before, the environment is constantly changing and thus also the beliefs of the agent. As the confidence and utility measures depend on it, the individual evaluation of the plans can change drastically and thus provoke an *apparently* incoherent behaviour of the agent. Like proposing to do the opposite that was proposed seconds ago. If we guarantee that a set of *coherent* plans survive to the next generation, the changes in the plans that are available for negotiation is not so drastic and therefore the behaviour of the agent will not be too erratic. On the other hand, a certain degree of diversity allows the search to explore the whole space of plans. This permits the agent to be prepared for any changes in the environment when they happen. Just in the same way as diversity acts in nature. Adjusting the level of coherence and diversity is thus crucial to have a good set of plans available at any time. The specific design including the codification of plans into chromosomes as well as the way to build the first population, and how to apply crossover and mutation depend, obviously, on the application domain.

5 Negotiation

Agents' plans of action usually contain basic plans with actions assigned to other agents. Those plans are considered joint plans. Plans and joint plans are formulated by the action planning module, see Figure 3, but the other agents involved in those plans do not know about them. Or, at least, do not know about our interest on them. Thus, the confidence that we have a priori on those plans may not be good at all. We would like that the other agents commit to the performance of the actions associated to the plans with the highest utility values because the commitments increase the confidence in their eventual execution and then our expected utility increases.

The first thing we need to do is to check which are our options for negotiation. Then, we evaluate them taking into account our goal in the negotiation, the previous commitments that we have and our independence. In the following we describe in more depth those dimensions, their usage in the model and how to formulate the negotiation options.

The options in a negotiation are the deals that we can accept [7]. A deal is in fact a plan. The signature of a deal commits the agents accepting the deal to perform the actions that the plan assigns to them. The action planning module provides us with a set of plans that could be used as deals. However, it is sometimes necessary to split those plans into several deals as they need to be negotiated separately with different agents. Thus, from the plans we produce deals that have actions assigned to our agent and just another agent. We denote the set of deals produced by a plan p as $deals(p)$, which is a set of plans: $deals(p) \subset 2^p$.

When a deal has been accepted, the negotiating agents commit to perform the actions in the basic plans that are assigned to them. Commitments being honoured up restrict the behaviour of agents. Achieving commitments from other agents is essential as their behaviour gets restricted and then the confidence for our plans increases.[2] At the same time, our commitments restrict our behaviour making us more predictable and thus vulnerable from being exploited. After signing a deal, some of the previous possible plans get incompatible with the current commitments (the set of commitments is in fact a plan). Unfeasible plans are removed from the set of options

A deal is *good* if it provides satisfaction to the agent. We thus evaluate a deal as the estimated satisfaction of the agent desires when adding this deal to the set of commitments, $\delta \cup C^t$. Agents want to get the best possible set of commitments, that is, the set that is compatible with the plans that have the highest utility. At the same time, the confidence in those plans has to be as high as possible. And also the set of available plans, i.e. independence, has to be kept as large as possible to allow for further improvement of the current set of commitments. Therefore, the satisfaction of an agent is defined as an aggregation operator over three criteria: utility, confidence and independence. This aggregation operator may change along time.

Definition 8. *Given a set of plans P^t and a set of commitments C^t, the* satisfaction *of an agent is defined as an aggregation of three different criteria: utility, confidence and independence.*

$$sat(P^t, C^t) = f^t(v_{\mathcal{U}}, v_{\mathcal{C}}, v_{\mathcal{I}}) \tag{8}$$

where: f^t is an aggregation operator (see for instance [10]), $v_{\mathcal{U}}$ is the OWA average utility of the plans in P^t, $v_{\mathcal{C}}$ is the average confidence of the plans in P^t and $v_{\mathcal{I}}$ is the percentage of plans in P^t compatible with C^t.

The coherence and diversity that we were looking for when searching for plans is crucial to be able to obtain an interesting set of deals. If the coherence of the plans is high, we will have options that do not restrict a lot our independence. When we sign a deal that is incompatible with a plan, we lose the opportunity to achieve it. And thus, the opportunity to use the deals that are only provided by those plans that are now incompatible. In this way, if we are not capable of convincing someone on accepting to perform one of the basic actions of our preferred plan, then we will still be able to use the already signed deals coming from this plan to satisfy other plans that are good as well.

The commitments are just the union of all the deals that are already accepted. Those deals can be jointly incompatible. This would mean that someone will deceit. Trust will

[2] As the success of the execution of our actions depend on the actions that other agents perform.

be useful to estimate who will deceit and to decide if we should honour or not a deal. Finally, legitimacy is not explained here because we focus on the exchange of deals and not other types of information. In future work we will use upper language levels in the tower [8] that allow for information exchange and then the legitimacy dimension will become important.

6 Discussion and Future Work

In this paper we have outlined the basic components of a negotiation architecture. It is tailored to build agents capable to interact with humans in competitive environments. One of the main characteristics of the environments we envisage is that agents and humans can occasionally co-operate through joint plans in order to get a better outcome. Those plans are possible thanks to the negotiation processes ending successfully by signing deals where agents commit to perform actions.

Our main contributions are: first, the interleaving of on the fly generation of plans and options for the negotiation and the negotiation process itself; second, the use of a notion of coherence and diversity to better explore huge search spaces that allow for a more compact process of negotiation without erratic moves. We also advance the state of the art in the modelling of graded BDI [6] by better integrating BDI reasoning and planning.

We are currently building a *DipGame agent* using this architecture and test it playing against humans. Our aim with these experiments is to empirically show that the architecture has the right components to be able to negotiate against humans in complex environments, multiple bilateral negotiations, repeated encounter scenarios, time fixed negotiation rounds, joint plans and uncertainty.

Acknowledgments

Research supported by the Agreement Technologies CONSOLIDER project under contract CSD2007-0022 and INGENIO 2010, by the Agreement Technologies COST Action, IC0801, and by the Generalitat de Catalunya under the grant 2009-SGR-1434.

References

1. Fabregues, A., Sierra, C.: An agent architecture for simultaneous bilateral negotiations. In: Proceedings of the 13è Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial (CCIA 2010). (2010)
2. Debenham, J., Sierra, C.: An agent supports constructivist and ecological rationality. In Baeza-Yates, R., Lang, J., Mitra, S., Simon, eds.: 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Milano, IEEE Computer Society, IEEE Computer Society (2009) 255–258
3. Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In M§ller, J., Rao, A., Singh, M., eds.: Intelligent Agents V: Agents Theories, Architectures, and Languages. Volume 1555 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1999) 1–10

4. Shoham, Y.: AGENT0: A simple agent language and its interpreter. In: Proceedings of the Ninth National Conference on Artificial Intelligence. Volume 2. (1991) 704–709
5. Broersen, J., Dastani, M., Hulstijn, J., van der Torre, L.: Goal generation in the BOID architecture. Cognitive Science Quarterly **2**(3-4) (2002) 428–447
6. Casali, A., Godo, L., Sierra, C.: A logical framework to represent and reason about graded preferences and intentions. In: Eleventh International Conference on Principles of Knowledge Representation and Reasoning, KR 2008, Sydney, The AAAI Press (2008) 27–37
7. Sierra, C., Debenham, J.: The logic negotiation model. In: Proceedings of Sixth International Joint Conference on Autonomous Agents and Multi-agent Systems. (2007) 1026–1033
8. Fabregues, A., Navarro, D., Serrano, A., Sierra, C.: Dipgame: A testbed for multiagent systems. In: AAMAS '10: Proceedings of the ninth international conference on autonomous agents and multiagent systems. (2010) 1619–1620
9. Debenham, J., Sierra, C.: Trust and honour in information-based agency. In: Fiftth International Joint Conference on Autonomous Agents and Multi Agent Systems, AAMAS 2006. (2006) 1225–1232
10. Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer (2007)