A Negotiation Framework for Negotiation of Coalition Policies

Mandis Beigi, Jorge Lobo, Keith Grueneberg, Seraphin Calo, and John Karat
IBM T. J. Watson Research Center
Hawthorne, USA

Abstract—There have been many proposed approaches to performing negotiation in terms of the negotiation procedure, the implementation of agreement, the interactions of software agents representing the different organizations, cooperation among agents, etc. However, one cannot determine a best single approach as it highly depends on the specific application and usage scenario, as well as the needs and goals of the participants. For instance, in some situations, reaching a near pareto-optimal solution is desirable even though it requires that an exhaustive search on all attributes must be performed. In other situations, time might be more valuable and therefore reaching an agreement in a timely manner might have a higher priority. In order to address many different types of negotiation goals and scenarios, there is a need for a flexible negotiation system that can incorporate various alternatives and that is easily extensible and configurable.

In this paper, we provide a generic negotiation system that can support many types of negotiation protocols. The proposed system acts as a third party that facilitates the negotiation process between multiple entities and allows them to choose a common negotiation goal and a desired negotiation protocol. We will provide a demonstration of the tool at the conference.

Keywords- Policy negotiation, policy conflicts, extensible, coalition

I. INTRODUCTION

Negotiation, in general, is the process of making proposals, trading options, offering concessions and obtaining mutual agreement [2, 3, 4, 8]. It takes place between two or more entities with an ultimate goal of satisfying all entities with mutual agreement. Humans engage in very complex negotiation processes [8] some of which have precise rules of interaction such as in auctions, but many negotiations create or modify rules in an ad hoc manner. Many times the success or failure of the negotiation depends on these rules and the strategies used by the negotiating entities.

We consider coalition environments in which a set of policies govern the operation of each of the partners and there is some level of interaction between the participants but we do not necessarily assume full cooperation. While planning for coalition operations, the coalition members need to suggest policies for the joint mission and determine the constraints that exist due to the presence of their own policies, and the interaction of these with the policies of other partners.

In order to expedite the planning phase, the coalition members need the capability to analyze the policies that will be in effect, and understand the impact of those policies on the joint mission. Each partner must analyze the policies suggested by their coalition partners together with their own organizational policies to see whether they permit a desired set of operations to be performed. The participants may have their own views on the mission priorities and may or may not be sharing their full set of policies with other coalition members. After such analyses are performed, each partner may try to negotiate a modification of policies with the other coalition participants to better suit their objectives. The negotiation can result in either a set of modified policies that all partners would agree to use for the joint mission or a disagreement and a voided mission.

The two types of capabilities thus needed by coalition members for a mission are: (a) policy analysis or evaluation capabilities, and (b) policy negotiation capabilities. With the rich environment of policy negotiation in mind we have developed a generic negotiation framework (i.e. not restricted to negotiating policies) and can be easily customized to support various negotiation models.

II. RELATED WORK

Most of the work in automatic negotiation is directed to negotiations between software agents (see, for example, [4, 10] and the references therein). Closest to our work is the framework introduced by Bartolini et. al. [13] for automatic negotiations. The main contribution of that work is a taxonomy of rules that can be used to capture a large class of negotiation mechanisms. The authors describe an implementation based on an agent architecture where each agent specializes in the evaluation of a specific class of rules (specified in description logic) to demonstrate how the rule taxonomy can be used. In our case, we use an open architecture where their agent-based implementation or implementations based on other types of formal descriptions such as the Law Governed Interactions model.
of Minsky et al [14] or more ad hoc mechanisms to describe negotiation protocols and strategies can be embedded. We need this generality because our purpose is to support humans in their negotiations.

Specifically for policy negotiation there is prior work on dealing with particular applications. Xue et al. [17] propose a negotiation mechanism for access control policies and determine conflicts using policy combining algorithms of XACML. Their work is limited to access policies and determine conflicts using policy combining operations. Ao et. al. [6] introduce a model of coalition where coalition policies are set a priori and the entities set their policies according to the global coalition policies. This model does not require negotiation and is useful for scenarios where coalition policies are already defined. However, in cases where the entities have never participated in a coalition, or they are formed in an ad-hoc manner, negotiation is needed to establish the coalition policies for the first time.

While there is considerable work in examining human-human negotiation [8], little has been done to develop technology to support complex human negotiation. This is partly because negotiation is a complex activity, and implementations have tended to focus on narrow slices of the topic. It would seem that in order to meet the requirements of negotiation, systems would need to provide the right level of flexibility.

III. NEGOTIATION FRAMEWORK

There are two basic components in a negotiation: a collection of negotiating entities and a set of issues that will be settled during the negotiation. To be more precise, an issue is an attribute in the negotiation for which the negotiating entities need to agree on a value. Thus, negotiating entities are entities that can assign values to attributes and accept or reject offers. Depending on the level of abstraction in which the negotiation takes place, the value of an attribute could be as simple as the price or the color of an object but it can also be a representation of a complex object such as the implementation of a process (e.g., making a direct deposit in a bank account every month) or a policy or regulation. Because human negotiation is a very complex process, the formal study of negotiation protocols and strategies has been proved to be difficult [14, 15, 16]. The research in negotiation strategies and protocols starts with the study of bilateral negotiations where only two entities are involved and a single attribute value needs to be chosen. But most practical situations require multilateral negotiations with multiple attributes [7]. Since our goal is to provide a general negotiation framework our platform assumes that multilateral multi-attribute negotiations will take place — single-attribute bilateral negotiations are a special case. In multi-attribute negotiations, the agents are faced with the problem of solving multiple issues (i.e., assigning values to multiple attributes) simultaneously. They need to decide on two things before negotiation begins: 1) the negotiation procedure and 2) the agreement implementation. These two form a negotiation framework of multi-attribute negotiation [5]. In addition to these two parameters there are five other parameters that might affect how a negotiation is carried out [3, 10]. These are: 3) the information model; 4) the interaction among the negotiating entities; 5) the model of cooperation; 6) the enforcement of the agreements; and, 7) the time frame for negotiation.

1. Negotiation procedure — There are three types of negotiation procedures: separate, simultaneous and sequential [11, 12]. Separate negotiation procedure means the agents negotiate each issue separately and independently but simultaneously. Simultaneous negotiation means the agents negotiate the issues one by one sequentially (i.e., issue by issue) without going back to renegotiate the already resolved issues. In sequential negotiation, ordering of the issues is important and must be decided by the agents. For example, they may decide to negotiate the most important issue first [9].

2. Agreement implementation — There are two ways to come to an agreement: sequential and simultaneous. Sequential implementation means the agreement on each issue is implemented once it is reached. Simultaneous implementation means agreements are implemented together when all issues are settled.

3. Information model — This dictates what is known to each agent. Therefore the information known to each agent may be complete or incomplete (i.e., partial knowledge). When there is complete information, the agents know both the state of the world and the preferences of all the other agents. However, incomplete information can affect the knowledge of the state and/or the agents’ preferences.

4. Agent interaction — There may be different types of interactions between the agents such as the agents making alternating offers versus one-shot offers. In alternating offer negotiations, each agent takes a turn.

5. Agent cooperation — There are two types of agents: cooperative agents and non-cooperative agents. Cooperative agents try to maximize their combined joint utilities while non-cooperative agents try to maximize their own utility regardless of the other agents’ utilities.

6. Enforcement — The negotiation protocol must state whether the agreement should be forced or not and if there are rewards or penalties for compliance or failure to comply with the agreements.

7. Negotiation time frame — The negotiation may have a finite timeframe or an infinite timeframe. When there is only a limited time the agents need to come to an agreement in a timely manner. In such scenarios time is costly.

Depending on the scenario and the negotiation participants’ choices, different options or variations of
each of the above methods may be relevant and used. We have designed a generic negotiation system that is extensible and allows the negotiating entities to choose different types of settings for many of these negotiation features.

IV. POLICY NEGOTIATION SYSTEM ARCHITECTURE

In our framework, we will consider an offer to be a set of attribute values. A minimal negotiation system will accept offers from multiple entities and decide, after consulting the negotiating entities, if there is agreement on an offer. In order to make such a decision, the system needs to be capable of performing one or a series of evaluations on the proposed offers. The types of evaluations may vary depending on the scenario and the objectives of the participants. For example, if the participants’ goal is to have a set of conflict-free policies, then the type of evaluation that the system needs to perform is conflict evaluation. When there are multiple attributes or issues that need to be negotiated, there will be different evaluation functions needed where each evaluates a different issue.

In order to make the negotiation system more useful for the participants, it should be able to suggest a set of valid offers to the users. The evaluation function may be capable of deciding whether an offer is acceptable.

Based on these intuitions and the different methods of negotiation mentioned in Section III, we have designed a negotiation system architecture with extensible modules as depicted in Figure 1. The system is a Java library that contains various interfaces which may be implemented in many different ways depending on a specific negotiation scenario. A system developer may provide many different implementations for the users to choose from, hence customizing the negotiation protocol before a negotiation session begins. Together with the negotiation system, there is a negotiation portal that exposes to the negotiating entities the different capabilities implemented in the negotiation system. The portal also provides access to application specific authoring tools for offers. The combination of both components forms the negotiation framework. We will discuss the negotiation portal in more detail in Section V.

The proposed system supports multiple concurrent negotiation sessions where each session has a set of participating entities and a negotiation goal, which determines whether the negotiation process successfully terminates. There is a negotiation session manager which manages the negotiation sessions and the sets of offers. Each session has a goal, a set of offer evaluation procedures, a turn taking protocol, a component to control the visibility of the offers, a negotiation procedure for resolving the issues, and a termination module. Next, we describe the different components of the system in more detail and relate the components to the seven parameters of a negotiation process introduced in Section III.

A. Negotiation Session Manager

The negotiation session manager manages all the concurrent sessions. It keeps track of the participating entities in each session and their submitted offers. It organizes the offers by the submission timestamp and the submitting entity. The negotiation session manager uses a single repository to store the all the submitted offers. Any authentication and access control that need to be implemented, including constraints on the number and types of participants in a negotiation session, will be part of this component. The negotiation session manager provides the following APIs:

- void openNegotiationSession(NegotiationSession negotiationSession, String organization);
- void joinNegotiationSession(String sessionID, String organization);
- void closeNegotiationSession(String sessionID);
- void acceptNegotiationOffer(String sessionID, String organization, INegotiationOffer offer);
- INegotiationOffer[] getNegotiationOffers(String sessionID, String organization);
- INegotiationOffer[] getNegotiationOffers(String sessionID);
- INegotiationOffer getLatestNegotiationOffer(String sessionID, String organization);
- INegotiationOffer getLatestNegotiationOffer(String sessionID);

B. Offer Evaluation

The system provides a set of built-in components which can perform evaluations on the offers submitted from each entity. Some examples of offer evaluation in the case of policy negotiation are conflict (among policies) evaluation, cost (of implementing/enforcing the policies) evaluation, and impact (on the system for deploying the new policies) evaluation. Once the evaluations are performed on the submitted offers, the results may be viewed by all or some of the negotiating entities in order to guide them in changing their offers to reach an agreement. The offer evaluation interface provides the following APIs:

- void performEvaluation(INegotiationOffer offer, String organization);
- I ofreceEvaluationResult getEvaluationResult();
- Boolean isAcceptable(INegotiationOffer offer, String organization);
- INegotiationOffer getSuggestedOffer(String organization);
- Boolean isEvaluationGoalAchieved();
- Float getUtility();
The performEvaluation() API is coordinated with the negotiation session manager component to make sure that the full offer is available for evaluation (each participant submitted an offer before the evaluation starts). There are also situations in which the evaluation requires multiple offers before the evaluation is made. The result of the evaluation may be any simple or complex value type which needs to implement the IofferEvaluationResult interface. The getEvaluationResult() API will be exposed through the negotiation portal in coordination with the offer visibility component. Showing the results of the evaluations to the entities may expose negotiation strategies of the owner of the evaluation procedure.

Different types of checks may be performed by each evaluation function to determine whether the offer being submitted matches certain criteria. For example, if 30 year loans are not allowed for commercial properties only loan offers with 5, 10 or 15 years will be accepted. The isAcceptable() API allows the user interface to know whether the offer is acceptable during each submission. The offer evaluation module may also use the utility function of the offer (described next) to determine whether an offer is acceptable. The getSuggestedOffer() API may be used to guide the user in determining an appropriate offer. A suggested offer may be enforced by the negotiation portal in which case the negotiation system becomes an arbitrator, otherwise, it will only present the suggested offer to the user in which case the system will act as a mediator [18]. The suggested offer may be optimized for a particular negotiating entity based on its own utility function in which case the entity will be considered non-cooperative. If the optimization is done using a global utility function, the entity will be considered cooperative. Our current implementation assumes a single utility function restricting the negotiation to be cooperative.

C. Negotiation Goal

The negotiation goal component provides an API to determine whether the negotiation goal has been achieved as well as its utility (i.e. how well the entities have achieved their goal) in order to decide whether the negotiation process can terminate successfully. The negotiating entities must all agree on the goal before negotiation begins. A possible goal evaluation mechanism is to ask directly each negotiating entity if the offer is acceptable. If all respond yes the goal has been achieved. Another example of a negotiation goal in the case of policy negotiation is to have a conflict-free set of policies which has a cost of deployment < $T.

Because of the multiple attributes, a negotiation goal may be the result of a combination of factors in the offer. We take advantage of the offer evaluation components and require the negotiating entities to define the negotiation goal based on linear combinations of offer evaluations. The following parametric formula is used to calculate the utility of an offer. The offer utility is a weighted sum of the evaluation utilities of all the different types of evaluations being used on the offer. The evaluation utility $E_{B,i}$ is a normalized value between 0 and 100. The weight reflects the importance factor of the corresponding evaluation type and ranges from 0 to 1.

$$U_B = \sum w_i E_{B,i}$$

where: $U_B$: total utility of offer B  
$w_i$: weight of each type of evaluation  
$E_{B,i}$: evaluation utility for $i^{th}$ evaluation type for offer B

$$w_i = P_k / \sum P_k$$

where: $P_k$ is an importance factor of the $k^{th}$ evaluation type. These are the parameters passed to the system to define the goal and need to be picked by the negotiating entities before the negotiation starts.

Figure 2 illustrates how the total utility of an offer is calculated. The goal is achieved when all the evaluations’ result values fall within a certain range. Evaluation $i$ is achieved if its result is within a certain range.

The goal interface provides the following APIs:
- Boolean isGoalAchieved();
- Float getUtility();

These APIs will be used by the negotiation portal to determine whether the entities have reached a successful termination of the negotiation process. The portal can also provide feedback to the entities in achieving the goal by displaying the total utility $U_B$. Due of the normalization this is a number from 0 to 100.

D. Negotiation Termination

The negotiation termination module performs the necessary actions when the negotiation entities leave the negotiation process. An entity may decide to drop out of negotiation for any reason before the negotiation terminates successfully. The system will need to notify all the entities when a participant decides to leave and in some cases this might cause the termination of the negotiation process. The negotiation termination interface provides the following APIs:
- void closeWithSuccess();
- void closeWithFailure();

E. Negotiation Procedure

The negotiation procedure module determines the order in which the different issues must be negotiated. This module provides the agenda for the negotiation process as proposed by Pershman [9]. The module may use the result of the evaluations to sort and determine which issue must be negotiated next. The negotiation portal may use this module to ensure that the users are following the agenda. The negotiation procedure interface provides the following APIs:
- String[] getNext();
The API will provide the list of attributes under negotiation in the current round and it will be empty when all the attributes have been covered. When there is no order (i.e. the procedure is simultaneous), the API will return NULL.

F. Turn Taking

In order to address the agent interaction method as discussed in section III 4, there must be a turn taking mechanism enforced by the system. The turn taking may follow a static order or it may be dynamically determined during or at the end of each round. One implementation might be round robin which makes sure each entity (i.e. organization) takes a turn in submitting their offers. Another implementation might be open turn which does not impose a turn-taking restriction and allows any entity to submit their offer at any time. The turn taking module uses the offer timestamps to decide which entity’s turn it is to submit their offer. More sophisticated turn taking algorithms may use the evaluation results of each entity’s offer submission to determine the order of the next submission round. This module addresses the agents’ interaction method. The turn taking interface provides the following APIs:

- Boolean isTurn(String organization);
- Boolean isFirstTime(String organization);

The negotiation portal will use these APIs to decide when and from whom to accept offers.

G. Offer Visibility

The offer visibility component may be used to determine whether a negotiating entity can see the offers from the other entities. An open offer implementation may allow every entity to be able to see all other offers at all times. A blind offer implementation (as in bids for a requisition negotiation) does not allow any entity to see the offers from other entities. This module partially defines the information model mechanism. The offer visibility interface provides the following API:

- Boolean isVisible(String organization1, String organization2);

The negotiation portal may use this API to decide whether it can show the offer from one entity to another entity at any point in time during the negotiation process. Note that offer evaluations and suggestions may take into consideration other offers if they are visible.

H. Offer Attribute Parameters

Each attribute of a negotiation offer has three parameters which may be set before the offer is submitted into the system. These are: ‘state’ and ‘private’ which are Booleans and ‘owner’ which is a string.

An attribute is set as a ‘state’ attribute if it is an attribute that describes part of the state of the world but is not part of the negotiation (i.e., it does not appear in offers).

The ‘private’ parameter is used to let the system know whether the attribute can be disclosed to the other negotiating entities. The value of the attribute is always known by its owner. For strategic or security reasons a negotiating entity may conceal some of their attributes. Non-private offers may be exchanged during negotiation while private offers may not. Currently, the attribute privacy is global and we do not support attribute level privacy per entity.

Finally, the ‘owner’ of an attribute can be either of the entities in the negotiation.

Table 1 summarizes which system components address which parameter in a negotiation framework.

<table>
<thead>
<tr>
<th>Negotiation Method Type</th>
<th>System Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negotiation Procedure</td>
<td>Negotiation Procedure</td>
</tr>
<tr>
<td>Agreement Implementation</td>
<td>Outside scope of Negotiation</td>
</tr>
<tr>
<td>Information Model</td>
<td>Offer Visibility, Offer Attribute Parameters</td>
</tr>
<tr>
<td>Agent Interaction</td>
<td>Turn Taking</td>
</tr>
<tr>
<td>Agent Cooperation</td>
<td>Offer Evaluation</td>
</tr>
<tr>
<td>Enforcement</td>
<td>Outside scope of Negotiation</td>
</tr>
<tr>
<td>Negotiation Time Frame</td>
<td>Offer Evaluation</td>
</tr>
</tbody>
</table>

V. BUILDING A NEGOTIATION PORTAL

In this section we present the negotiation portal we have built that uses our negotiation system to allow users belonging to different entities to negotiate on a set of mission policies. The negotiation portal consists of five components: 1) session management; 2) offer authoring; 3) offer submission; 4) offer evaluation; 5) and offer suggestion.

The Session Management module manages the different concurrent negotiation sessions and allows users to create new sessions or join the existing ones. It makes calls to the APIs provided by the Negotiation Session Manager and the Negotiation Termination components in the negotiation system.

The portal allows authors belonging to different entities to login and join a negotiation session. The first entity logging into the tool may start a new negotiation session and select the settings which are agreed to by all the entities before the negotiation begins.

The Offer Authoring component of the portal allows the users to write their offers by setting the values of the different attributes of the offer. This component extends the negotiation system’s general offer type to specific offer types such as policy sets, prices, etc. This part of the portal uses an application dependent authoring component to compose the offers. This component also uses the API provided by the Offer Visibility component of the negotiation system as well as the Offer Attribute Parameters to determine what to present to each user. The authoring tool also uses the API provided by the Negotiation Procedure component to guide the user through the different issues (i.e. attributes) that need to be resolved. Our policy authoring tool is template based and uses a fixed policy structure but has the look and feel of natural language.

The Offer Submission component of the portal uses the APIs provided by the Offer Evaluation, the Turn Taking, the Negotiation Goal, and the Negotiation Session.
Manager components of the negotiation system. When an offer is provided by a user to be submitted, first the isTurn() API of the Turn Taking component is called to verify that submitting entity is allowed to submit the offer, then the isAcceptable() API is called to check whether the offer is valid. An offer may then be accepted by the Negotiation Session Manager’s API, acceptNegotiationOffer(). After these steps, the system checks whether the goal has been reached by calling the Negotiation Goal’s isGoalAchieved() API. If the method returns with a TRUE value, the closeWithSuccess() API in the Negotiation Goal is called to perform the proper actions such as informing all the entities.

The Offer Evaluation component allows the users to see the results of the evaluation functions on the submitted offer. This component may be specifically designed to present the evaluation outcomes in a way that suits the needs of the users and the application. It uses the performEvaluation() and getEvaluationResult() APIs provided by the Offer Evaluation component of the negotiation system.

The Offer Suggestion component of the portal uses the getSuggestedOffer() API provided by the Offer Evaluation component of the negotiation system to provide the user with an offer based on the chosen strategy.

The portal uses a plugin configuration file to determine what implementations are available for negotiation goal, negotiation procedure, offer evaluation, turn taking, offer visibility, and negotiation termination. The plugins file is in XML format and lists all the different implementations that are available in the negotiation system. When a new session is created, the user is prompted for various settings to be used during the negotiation process. The system uses the plugins file to determine all the available implementations of the system modules.

VI. CONCLUSION AND FUTURE WORK

We have proposed an extensible negotiation system architecture that enables different entities to collaborate and negotiate their offers in reaching a mutual goal. The system consists of various components where each component is responsible for a part of the negotiation process. The negotiation protocol can be customized by allowing different implementations of the various components of the system.

There are many directions in which we can extend our negotiation framework to make it more flexible. Currently, our system uses the same set of evaluation functions for offers coming from all the entities and does not associate each entity with different types of evaluations. In this version we also assume that all the issues that need to be resolved are known before the negotiation process begins. We plan to address these limitations in the near future. We are also planning to perform a complete evaluation of the tool.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0002. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES