A Contract Decommitment Protocol for Automated Negotiation in Time Variant Environments

Federico Bergenti, Agostino Poggi and Matteo Somacher
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parco Area delle Scienze 181A, Parma, Italy
Tel. +39 0521 905712 – Fax +39 0521 905723
{Bergenti,Poggi,Somacher}@CE.UniPR.IT

Abstract

Negotiation is a fundamental mechanism in distributed multi-agent systems. Since negotiation is a time-spending process, in many scenarios agents have to take into account the passage of time and to react to uncertain events. The possibility to decommit from a contract is considered a powerful technique to manage this aspect. This paper considers interactions among self-interested and autonomous agent, each with their own utility function, and focuses on incomplete information. We define a negotiation model based on asynchronous message passing in which the negotiation doesn't end when an agreement is reached but when the consequences of the contract have happened – i.e. the action is done. In this model the agent utility functions are time dependent. We present an extension of the contract net protocol that implements the model.

1. Introduction

Negotiation has long been considered a fundamental key in distributed multi-agent systems for coordinating autonomous agents and managing their interactions [10]. These systems has no notion of global utility function and negotiation is a powerful mechanism to share information and to reach acceptable agreements among self-interested and goal oriented agents, each with their own utility function [4]. There are two correlated topics that has to be examined in designing automated negotiation for multi-agent systems: negotiation protocols, that define the set of rules that govern the interaction, and strategies that agents can follow during the protocol [3]. More complex and powerful is the protocol and more sophisticated can be the strategies.

Negotiation is a time-consuming process. Time has to be taken into account when the overhead needed to find an agreement might cause the negotiation to fail or when the variations of the negotiable objects are significant during the negotiation steps. If the agent utility function is time-dependent, the model has to provide the mechanisms to allow agents changing their strategies during the negotiation [4][1].

Negotiation time has impact on the solutions also when the environment realizes unexpected or uncertain events. If the contract causes future consequences, the events may not occur only during the negotiation but also after the agents have reached an agreement. In some case to decommit from a contract can be desirable for both agents [8]. The possibility to decommit from a contract has been studied with different approaches in distributed problem solving, in auction optimization and automated negotiation systems. The *Leveled Commitment Contracts* [6] are been proposed to deal with this problem. They specify in the contract the penalties that agents have to pay if they decommit from the contract.

The recent increased interest in automated negotiation research due to the growing up of the Internet applications and Electronic commerce fixes some constraints in the multi-agent systems design. These environments are very dynamic and heterogeneous, they often does not provide notion of global consistent knowledge or global goal and need interoperability among different agents. Although there is no universally best technique for deciding which negotiation mechanism to adopt [3], because different negotiation contexts need different solutions, unifying interaction negotiation protocol and negotiation languages is an important issue to make actually software agents interacting in the real world. To reach feasibility it is important that the negotiation model doesn't have a big impact on the agent design.

Our aim in this paper is to propose a general high-level interaction protocol and a negotiation model that can be followed by agents in distributed and heterogeneous multi-agent systems with incomplete information, eventually adding some constraints in respect of the different peculiarities of the environments.

In the following section we present the negotiation model based on asynchronous message passing. The basic idea is that such model could support different levels of reasoning and decision making in respect of the multiagent system needs. It focuses on the communication acts; reasoning about the strategy models is out of the scope of the paper. In section 3 we present a unified negotiation protocol that allows agents to reach an agreement when it is feasible. It takes into account the possibility that uncertain events occur and it provides to agents the mechanism to change their strategies during the negotiation. We show how some simpler yet powerful negotiation protocols - i.e. English and Dutch auctions and Contract Net - can be derived from that general protocol fixing some constraints. In section 4 is presented the FIPA [2] implementation of such protocol that uses the communicative acts specified by the FIPA ACL language to support the interaction between agents.

2. The negotiation model

We define a negotiation model for the distributed multi-agent systems with a finite set of agents *A* that are self-interested and autonomous, each with their own utility function. The information about the agent strategies and knowledge is incomplete, so no notion of global utility function is present in the system. Agents interact with asynchronous message passing following an interaction negotiation protocol that establishes the negotiation rules. The agents, during the negotiation, try to reach an agreement on a finite set $X = \{x_1(t), x_2(t), \ldots, x_n(t)\}$ of negotiation values that are time dependent in the more general case. We call *negotiables* these negotiation values and we express them through an integer number value. Negotiables are general entities and they can represent goods, tasks to be performed or resources to be allocated.

We suppose that each agent is able to calculate a utility function $U(\bar{x}(t)):X\rightarrow\Re$, based on its local information, to evaluate if a negotiable can be agreed. In the model no assumption are made about the linearity of the utility functions and the negotiables are the only observable elements. The agent A_i considers the agreement feasible for a negotiable $\bar{x}(t_0)$ if $U(\bar{x}(t_0)).>T_i$, where $T_i\in\Re$ is the threshold level that the agent A_i fixes before starting the negotiation (in a more complex model also T_i could be time dependent). We call *acceptable negotiable space* for the agent *i* the bounded subspace $S_i\subseteq X$ of negotiables satisfying the A_i utility function. The goal of the negotiation process is to find a vector $\bar{x}(t_0)$ satisfying all the utility functions of the agents involved. If the negotiation process between two agent A_1 and A_2 succeeds, the final vector $\bar{x}_f(t_0)\in S_f= S_1\cap S_2$.

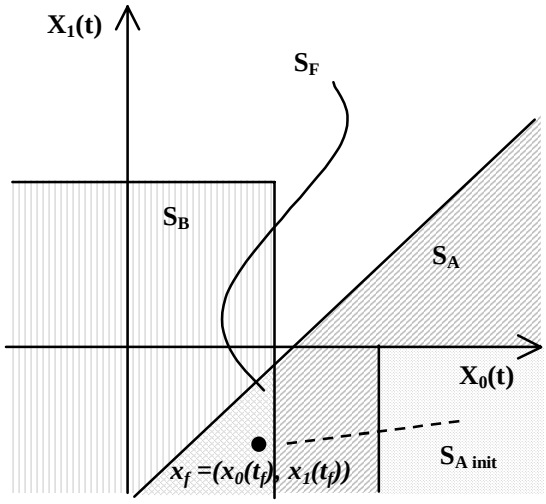

Figure 1. Example of a successful negotiation process.

The negotiation process starts with an agent proposing an acceptable negotiable space S_i (or a negotiable $\bar{x}(t)$) to another one. During the negotiation the agents exchange, within the proposals, some negotiable spaces or negotiables until the agreement is reached on a negotiable value $\bar{x}(t_0)$. The history of the negotiation process is represented by the sequence of spaces and vectors exchanged among the agents. The basic idea is to build an observable environment that can be used by the agents to manage strategies and decision-making techniques.

The negotiation process can involve only a subset of the full negotiable space *X* and the dimension of the negotiables could change during the negotiation. This possibility increases the complexity of the model and requires that agents are able to consider this in their strategies.

Figure 1 illustrates negotiation process in which two agents reach an agreement. Firstly the agent A_A propose to the agent A_B a space S_{Ainit}; after some iteration one of the two agents propose the agreement on the negotiable x_f that is in both *acceptable negotiable spaces* so can be accepted.

The model described above makes no assumption about agent beliefs and strategies, so we don't approach analysis of equilibrium or pareto optimality.

3. The interaction protocol

We assume we have a finite collection of agents that play the role of *initiator* $I = \{i_1, i_2, \ldots, i_n\}$ of the protocol and a finite collection of agents that play the role of *responder* $R = \{r_1, r_2, \ldots, r_n\}$. The protocol is symmetric. It is important to provide a symmetric negotiation

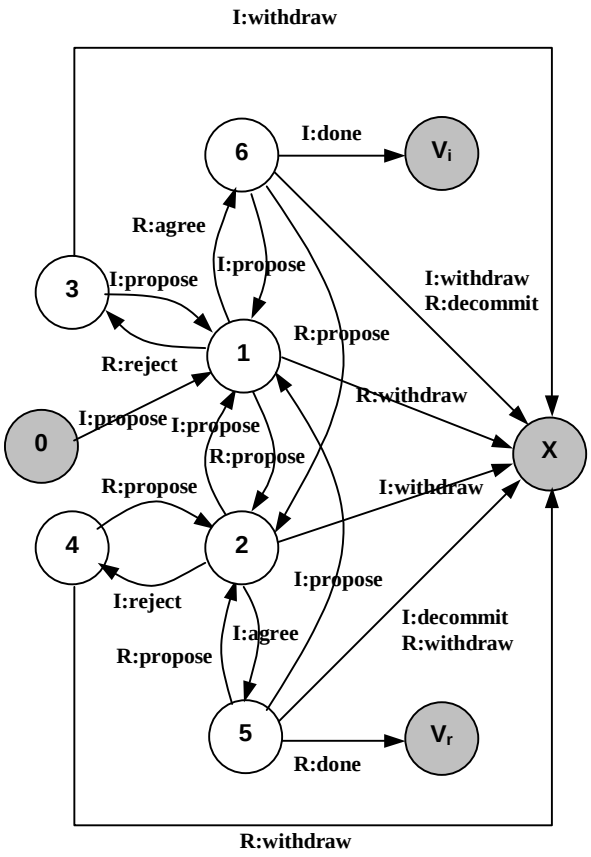

Figure 2. The negotiation interaction protocol

mechanism [5][10] where there are no special agents that have a different responsibility in the negotiation process. When the agent society needs a coordinator or a broker the responsibility assignment needs to be resolved before the negotiation starts [10]. This is the case of auctions that need some added rules and establishes that the initiator informs the other agents when negotiation ends.

The initiator agent is interested in starting the negotiating and it can request or offer a good. The negotiable objects [3] used in the protocol can be both negotiables and negotiable spaces. During the bargaining agents can iterate proposals exchanging objects negotiation until one agent accepts a proposal containing a negotiable $\overline{x}\ (t_0)$. The protocol fixes implicitly some negotiation rules, i.e., the order in which messages can be sent, some others has to be chosen by agents before starting the negotiation process adding the constraints imposed by the environment. We describe the interactions between two agents but the protocol can be used also, without modifications, in one to many (i.e., auctions) or many to many negotiation processes. Figure 2 shows the state diagram that describes the protocol.

We consider that the negotiation process doesn't end when the agents reach an agreement but when the consequences of the contract occur, i.e., the action requested is done. This because the time interval from the agreement instant to the action can be significant and sometimes much longer than what needed by the previous interactions. In the meanwhile, unexpected events can arise invalidating the reasons for the agreement, moreover the agent utility function, which is time-dependent, could fall under the threshold level. In that case we don't want to restart the negotiation because we would loose the information obtained from the process. Actually, if the action has not already been done the agents could take advantages in decommit from the contract [6][8]. We don't analyze here the impact of the decommitment in the agent society because it is closely related to the strategies and dependent on the reference scenario. So in the more general case we don't add penalties to the agent that decommits from the contract. If needed, penalties can be defined in the set o rules established between agents before the negotiation.

The protocol starts when the initiator agent issues a *propose* to the responder agent moving from state 0 to state 1. The initial proposal defines an initial negotiable object and which agent has to do the action specified in the contract, so it could express the intention to offer or to request a good (service). The responder agent has four possibilities:

- it can accept the proposal (going to state 6);
- it can *reject* the contract waiting for another *propose* (going to state 3);
- it can refuse to negotiate withdrawing the proposal (the protocol finishes with a non-success state).
- it can reply with a *propose* (going to state 2) refining the negotiable object;

The protocol is symmetric and in the state 2 also the initiator has four possibilities after receiving the proposal of the responder:

- it can accept the proposal (going to state 5);
- it can *reject* the contract waiting for another *propose* (going to state 4);
- it can reply with a *propose* (going to state 1) refining the negotiable object;
- it can refuse to negotiate withdrawing the proposal (the protocol finishes with a non-success state).

Now the negotiation process can take in several iterations, between state 1 and state 2, to find the negotiable that can be accepted by the agents. This is not the case of a cooperative environment where agents try to exchange the more information they can but it is important in a competitive environment where the negotiation goal for each agent is to obtain the best contract maximizing their utility function that often is proportional reverse to the others. If only one agent is responsible to modify the negotiable objects the following two cases can arise: if the

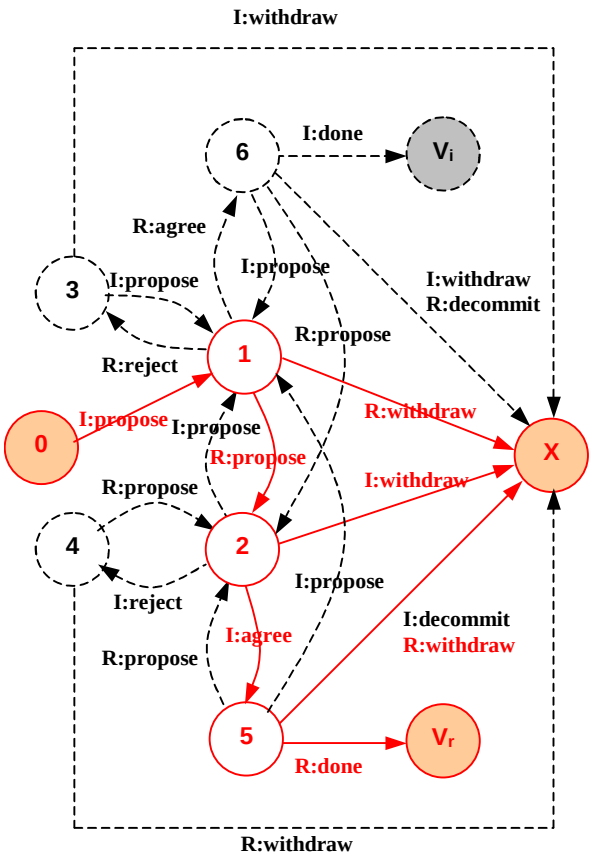

Figure 3. The Contract Net protocol derived from the general interaction protocol

agent that proposes to do something (or to provide some service) is the initiator, the iteration moves between state 1 and state 3, otherwise the iteration moves between the state 2 and state 4. Actually state 1 is the state in which the responder agent can accept the proposal and state 2 is the state in which the initiator can accept the responder proposal. Every time an agent rejects a proposal, the other can stop the negotiation protocol withdrawing to continue, so from states 3 and 4 is allowed the transaction to the final unsuccessful state.

Let's consider that the iteration ends and the agents reached an agreement. The protocol goes to the state 5 or to the state 6 with one agent that answers *agree* to the current proposal.

If no unexpected event occurs the agent that took charge of doing the action will inform the other that the action is done and the protocols ends in the V_i state if the initiator did the action (from state 6) or in the V_r state if the responder did the action (from state 5). If something goes wrong and the agents cannot provide what they promised because some unrecoverable error occur in the system, the protocol ends with a *withdraw* message although the agreement was reached.

Before the action is done both the agents can decommit from the contract. This simply cancels the previous agreement. If agents are no more interested in the negotiation they send a *decommit* message to the other agent ending the protocol. The decommitment can also used to refine the negotiable and continuing the bargaining even though a previous agreement was reached. The transactions from state 5 to state 1 and from state 6 to state 2 are an implicit decommit of the agents after they agreed a proposal. The transactions from state 6 to state 1 and from state 5 to state 2 are an implicit decommit of the agents that had their proposal accepted. In both cases the agents send to the other agent a further proposal because they intend that it would be better for both.

For deepening this aspect let's consider the following example: two agents reached an agreement about a service that will be provided at a time in the future. In the meanwhile a third agent offers the same service that is more advantageous for the client. The possibility to decommit allows the three agents to continue the negotiation without restarting the process from the beginning. Of course penalties can be considered if the environment requires them.

An interaction protocol has to guarantee success in a reasonable time and at reasonable computational cost. Without defining some specific negotiation rules the protocol can lead to negotiations that never terminates. Furthermore if the environment is not time stationary this risk could be more evident. When the agent utility functions don't are time dependent, if the offer made by an agent is less preferable to that agent than the previous ones and the agent withdraws when its utility function falls under a threshold, we can consider that such protocols guarantees success [9]. What is needed is to provide rules and constraints for such more complex environment that lead to the same assertions. Since the rules are closely related to the scenario, in a general protocol we have not to define them. However we could generally impose that the sequence of proposals during the negotiation process has to be monotonic to prevent infinite loops in the bargaining.

3.1 The Contract Net protocol

The Smith's Contract Net [7] is a fundamental high-level communication protocol for a Distributed Problem Solving and Distributed Artificial Intelligence systems. It enables the distribution of tasks among the nodes (agents) that operate in the system. Smith assigns two different roles to the contractors:

- the *manager*, responsible for monitoring the execution of the task;

– the *contractor*, responsible for the actual execution of the task.

Figure 3 shows how the Contract Net can be implemented adding some constraints to the general interaction protocol we propose, i.e., forbidding some transactions. The forbidden transactions are shown with dashed lines and the dashed circles are states that cannot be reached during the negotiation. The states and the transactions allowed are shown in red.

The initiator agent plays the role of manager and the responder plays the role of contractor. The initial task announcement message corresponds to the initial proposal from state 0 to state 1. If the contractor makes the bid we have the *propose* message that leads to the state 2 otherwise the protocol ends with the contractor that refuses to negotiate. The bid could be accepted or rejected by the manager. In the second case the protocol goes to the state 5 where the initiator agent wait for knowing if the task has been done.

3.2 The auctions

The auctions are non-symmetric and one-to-many negotiation processes. Forbidding some transactions is also possible to derive an auction protocol from the general interaction one. In this case the auctioneer plays the role of the initiator agent. The agents have to decide when the auctioneer can or has to inform the others that the auction ended.

The auction starts when the auctioneer calls for bids sending to the agents involved in the bargaining a propose message, moving from state 0 to state 1. The responder agents makes the bid moving to the state 2, then the auctioneer can accept (the protocol goes to the state 5) or reject the bid (the protocol goes to the state 4).

In the auctions, the auctioneer explicitly stops the protocol and according the type of action this can happen into different states. Let's consider two typical auction styles, the English and the Dutch:

– according to English style a low initial price rises gradually until a client declares its intention to buy;
– in the Dutch style the auctioneer starts with a price much higher than the real market value of the goods, then she lowers the price gradually until one of the clients accepts the suggested price.

In the English style the auctioneer inform that the auction ended when no bids are announced for the current proposal: the transaction occurs from state 1 to the final state. In the Dutch style the auctioneer stops the auction because nobody accepts the proposed price (from the state 1) or when the good is sold (the protocol is in the state 5 where the bidder accepted the price of the auctioneer).

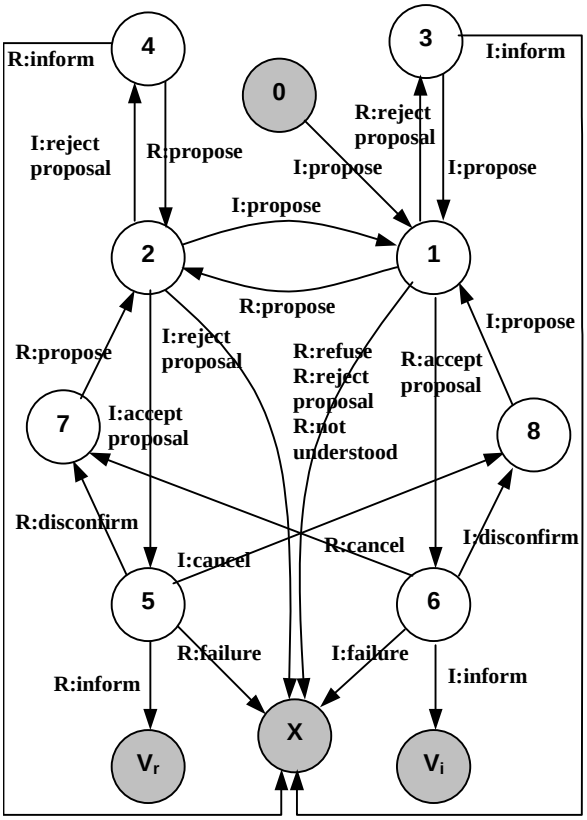

Figure 4. The FIPA mapping of the protocol

4. FIPA implementation

FIPA, the Foundation for Intelligent Physical Agents, tries to actively support agent sociality recognising that a way to ease the modelling of interactions between agents is to exploit interaction protocols [2]. FIPA provides four generic interaction protocols to support negotiation between agents, based on the FIPA ACL specifications: the FIPA contract net, the FIPA iterated contract net, the FIPA auction English and the FIPA auction Dutch.

We provide a FIPA mapping of the general negotiation protocol presented in this paper. Figure 4 shows the state diagram representation of this version. As also shown in the previous session, the auctions and the contract net can be derived from this protocol.

First of all we can notice that the FIPA version of the protocol has two more states. This is due to the formal semantic of the FIPA performatives. Actually these state (state 7 and state 8) are transient states and when the protocol goes by them the agents send two consequent ACL performatives. This because when an agent wants to decommit for doing another proposal it must invalidate the rational effect of the *accept_proposal* by sending a *disconfirm* communicative act to the other agent, then it

can send another *propose*. The *disconfirm* message is necessary to reject the contract because a new propose would not cancel the manager intendments about the previous proposal. Similarly if an agent wants to decommit after she accepting a proposal for refining the negotiable object it has to *cancel* the effect of the previous message before sending the *propose*.

When the agents want to stop the negotiation process because their offers continue to be rejected, they send an *inform* and the protocol goes from state 3 or 4 to the unsuccessful state.

This protocols differs from the FIPA negotiation ones because it begins with a *propose* message instead of a *callforproposal*. Actually a *callforproposal* is not necessary when the initiator agent offers a service (or to do an action). In the other cases the *propose* from state 0 to state 1 takes the place of the *callforproposal*. This allows to send one less message and to finish the protocol with 3 messages instead of 4 when no iteration is needed and the agreement is reached. The sequence of messages in that case would be: *propose*, *accept_proposal*, *inform(done)*.

5. Conclusions

In this paper we presented a negotiation model and a general interaction negotiation protocol based on asynchronous message passing that can be used in non-stationary environments. Agents can decommit from the contract to manage unexpected events. This allows agents having time dependent utility functions. We showed how auctions and some version of the Contract Net could be provided simply adding constraints to the general protocol. This is useful because it provides a skeleton adaptable to different scenarios and the agents could build strategies using mathematical instruments.

During the negotiation process the agents exchanges set of vectors representing the negotiable values. What an external observer would see is a space or a point moving on the negotiable value space until the agreement is reached or the bargaining ends without solutions.

We also provided a FIPA mapping of the general interaction negotiation protocol that can be easily employed by FIPA compliant agents.

Extensions of this research include analyzing and providing BDI agent strategies that were out of the scope of this work. It has also to be studied how, when and under which constraints the protocol guarantees success.

6. Acknowledgement

This project is partially supported by the grant CNR Applied Research Project 5% “Multimedialità”.

7. References

[1] J. Dix, S. Kraus and V. S. Subrahmanian “Temporal Agents Programs”. In Artificial Intelligence 127(1), pages 87-135, 2001.

[2] Foundation for Intelligent Physical Agents. FIPA ’99 Specification, Part 2: Agent Communication. Available at: http://www.fipa.org

[3] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra and M. Wooldridge “Automated Negotiation: Prospects, Mehods and Challenges”. In International Journal of Group Decision and Negotiation. 10(2), pages 199-215, 2001.

[4] S. Kraus, J. Wilkenfeld, G. Zlotkin “Multiagent Negotiation Under Time Constraints” Artificial Intelligence journal, 1995.

[5] J. F. Nash “The Bargaining Problem”. Econometrica 21, pages 155-162, 1950.

[6] T. Sandholm and V. Lesser “Leveled Commitment Contracts and Strategic Breach”. In Games and Economic Behaviour 35, pages 212-270, 2001.

[7] R. G. Smith “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver”. IEEE Transactions on Computers, C-29(12), pages 1104-1113, 1980.

[8] W. Walsh and M. Wellman “Efficiency and Equilibrium in Task Allocation Economies with Hierarchical Dependencies”. IJCAI, Stockholm, 1999.

[9] M. Wooldridge and S. Parsons “Languages for Negotiation”. ECAI, Berlin, 2000.

[10] G. Zlotkin and J. S. Rosenschein “Mechanisms for Automated Negotiation in State Oriented Domains”. In Journal of Artificial Intelligence Research 5, pages 163-238, 1996.