DR-NEGOTIATE – A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies

Thomas Skylogiannis[1] Grigoris Antoniou[2]
1 Department of Computer Science, University of Crete, Greece
dogjohn@csd.uoc.gr
2 Institute of Computer Science, FORTH, Greece
antoniou@ics.forth.gr

Nick Bassiliades[3] Guido Governatori[4]
3 Department of Informatics, Aristotle University of Thessaloniki, Greece
nbassili@csd.auth.gr
4 School of ITEE, University of Queensland, Australia
guido@itee.uq.edu.au

Abstract

This paper reports on a system for automated agent negotiation. It uses the JADE agent framework, and its major distinctive feature is the use of declarative negotiation strategies. The negotiation strategies are expressed in a declarative rules language, defeasible logic and are applied using the implemented defeasible reasoning system DR-DEVICE. The choice of defeasible logic is justified. The overall system architecture is described, and a particular negotiation case is presented in detail.

1. Introduction

In the last few years, there has been a great interest in electronic commerce potential. As the number of transactions carried out through the Internet increases, the interest for partial or full automation of these transactions increases as well [1]. This automation is achieved by the use of software agents' technology. One basic stage of e-commerce procedure that can be automated is the negotiation stage [2].

The focus of our work is on the automated negotiation aspect of e-commerce. As stated in [18], *automated negotiation* is the process by which two or more agents communicate and try to come to a mutually acceptable agreement on some matter. The basic dimensions of automated negotiation are negotiation protocols and negotiation strategies. *Negotiation protocol* is a set of rules which govern the interaction and a *negotiation strategy* is a decision making model, which participants employ in order to achieve their goal in line with the negotiation protocol.

As far as negotiation strategies are concerned, there are three possible approaches to design then according to [18]: Game theoretic, which models negotiation as a game, heuristic which employ a set of tactics and some rules for selecting a tactic, and argumentation-based which introduces performatives such as threats, promises etc. We take the heuristic approach and propose to use defeasible logic, a declarative language based on rules and priorities, as a formal framework to model protocols and strategies for automated negotiation; this is the major distinctive feature of our implemented system. Our work builds upon the previous theoretic work of [8].

At this point we must justify the choice of defeasible logic among various schemes for representing strategies and protocols. Firstly it is *formal*, that is, its semantics and syntax are properly defined. This means that both humans and computers can interpret them the same way, and *explanations* of the agent's behavior can be provided. Another characteristic of deafeasible logic is that it is *conceptual* meaning that it offers a good level of abstraction. So anyone can focus only on protocol or strategy design, being indifferent to the implementation. Defeasible logic is also *comprehensible* and *expressive* as well. The latter is very important because enables us to describe a wide range of protocols and strategies.

The paper is organised as follows. Section 2 provides a short discussion of the use of defeasible logics. Section

3 presents the system architecture, while sections 4-6 illustrate the functionality and use of the system based on a concrete case. Section 7 presents related work and finally, planned future work is described in 8.

2. Choice of Formalism

2.1 On Defeasible Logics

Defeasible reasoning is a simple rule-based approach to reasoning with incomplete and inconsistent information. It can represent facts, rules, and priorities among rules. This reasoning family comprises defeasible logics [14, 16] and Courteous Logic Programs [15], and has the following characteristics:

- They are rule-based, without disjunction
- Classical negation is used in the heads and bodies of rules, but negation-as-failure is not necessarily used in the object language (it can easily be simulated, if necessary [17])
- Rules may support conflicting conclusions
- The logics are skeptical in the sense that conflicting rules do not fire. Thus consistency is preserved
- Priorities on rules may be used to resolve some conflicts among rules
- The logics take a pragmatic view and have low computational complexity

Generally speaking, defeasible logics have two kinds of rules: *strict rules* which behave like standard, classical rules (once their premises are satisfied they fire) and *defeasible rules*, which may not fire even when their premises are satisfied, because they are blocked by other rules. More complex logics have a further kind of rules, so-called defeaters.

2.2 Why Defeasible Logics for Negotiation Strategies

Applying a negotiation strategy in a particular context is an intensive decision-making process. While most aspects of negotiation strategies could be fully captured in classical logic programming (which has a formal semantics and has proven to be a powerful tool for building decision-making systems), this would put a burden on the developers of strategies, since logic programming is a generic paradigm and offers nothing specific to strategy specification (such as argumentation, defeasibility, hypothetical reasoning, preferences, etc.). Accordingly, we propose to use a logic programming language based on non-monotonic reasoning. Among the many members of the family of non-monotonic logics, we choose defeasible logic [16] for the following reasons:

- A negotiation can be thought of as a dialogue between parties concerning the resolution of a dispute. This suggests that argumentation based reasoning formalisms are suitable to characterise it. In [19], it was shown that defeasible logic can be characterised by an argumentation semantics, thus the formal semantics of defeasible logic is in line with the argumentative nature of negotiations.
- Given the close connection between derivations in Defeasible Logic and arguments, strategies expressed in Defeasible Logic are explainable.
- Defeasible logic is a sceptical formalism, meaning that it does not support contradictory conclusions. Instead it seeks to resolve conflicts. In cases where there is some support for concluding A but also support for concluding ¬A, the logic does not conclude either of them (thus the name "sceptical"). If the support for A has priority over the support for ¬A then A would be concluded. We believe that non-sceptical reasoning is unappropriate for modeling decision-making processes such as negotiations, since it is quite useless to deduce both that a decision should be taken, and that it should not be taken.
- Defeasible logic integrates the concept of priorities between rules, thereby supporting a direct way of modeling preferences, without having to attach a metric to them, as it is the case of approaches based on utility functions [20].
- Regarding strategy specification, most of the current systems adopt a quantitative approach based on utility functions. Very often, it is not easy to find the right utility functions for a given set of negotiation issues, especially in situations where one needs to express preferences without attaching a metric to them. Moreover, utility functions are mostly used to determine preferences that can otherwise be expressed in a more comprehensible and suitable way through defeasible rules and priorities among these rules. For this reason, we believe that defeasible logic is more suitable than, or at least complementary to, strategy specification approaches purely based on utility functions.
- Defeasible logic has a linear complexity, and existing implementations are able to deal with non trivial theories consisting of over 100,000 rules [21], offering thus an executable and scalable system.

3. Implemented Agent Architecture

The *Agent Framework* we used was JADE [6, 7]. JADE is an open-source middleware for the development of distributed multi-agent applications based on the peer-to-peer communication architecture. JADE is java-based and compliant with the FIPA specification. It provides libraries for agent communication and interaction, based on FIPA standards. It also provides tools for agent lifecycle management, inspection of exchanged messages and debugging. JADE provided us with the *Agent Infrastructure* we desired. In our case, there is no *Agent System Architecture* as we only have implemented a demonstrator and not a complete multi-agent negotiation system.

The *Agent Architecture* we implemented was primarily based on the architecture proposed by Dumas et al. [8]. Software agents consists of four components: (a) A memory which contains past decisions and interactions (Knowledge Base), (b) A communication module which handles incoming and outcoming messages (JADE platform), (c) A reasoning module (DR-DEVICE Inference Engine) (d) A control module for the coordination of the above components (script in Java).

For the reasoning module of the agent we used DR-DEVICE [12]. DR-DEVICE is a defeasible reasoning system. Its user interface is compatible with RuleML, the main standardization effort for rules on the semantic web and is based on a CLIPS-based implementation of deductive rules.

The architecture of the negotiating agent is depicted in Fig.1. When the agent is notified of an external event, such as an incoming message (step 1), the control module initially retrieves a fact template from the local storage unit (step 2) and consequently, the negotiation parameters from the memory (step 3). The template is an empty placeholder in line with DR-DEVICE system syntax. When the template is filled with the negotiation parameters, is then regarded as "the facts". The control module updates the knowledge base with the new facts (step 4) and then activates DR-DEVICE (step 5). DR-DEVICE in turn retrieves from the knowledge base the facts, along with the strategy (step 6) and starts the inferencing process. After the inferencing has been completed, the knowledge base is updated with the results (step 7). The control module queries the knowledge base for the result (step 8) and after a short processing; an appropriate message is posted to the communication module.

4. The Negotiation Protocol: An Example

As we have already stressed, a basic condition for the automation of the negotiation process among intelligent agents, is the existence of a negotiation protocol, which encodes the allowed sequences of actions, or in other words the rules of the game. Our first thought was to use a well-defined protocol for 1-1 automated negotiation. Although FIPA provides a plethora of standardized protocols, such as FIPA brokering, FIPA English auction, FIPA Contract net etc., we found that there is no standard interaction protocol, when it comes to 1-1 automated negotiation. As a result, we implemented a negotiation protocol proposed in [9].

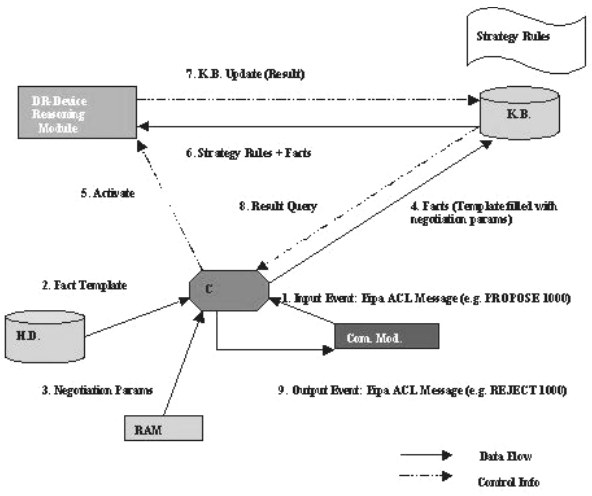

Strategy Rules
7. K.B. Update (Result)
DR-Device Reasoning Module
K.B.
6. Strategy Rules + Facts
5. Activate
8. Result Query
4. Facts (Template filled with negotiation params)
2. Fact Template
C
1. Input Event: Fipa ACL Message (e.g. PROPOSE 1000)
H.D.
Com. Mod.
3. Negotiation Params
9. Output Event: Fipa ACL Message (e.g. REJECT 1000)
RAM
Data Flow
Control Info

Fig.1. Architecture of Defeasible Logic Based Negotiating Agent

This protocol is a finite state machine that must be hard-coded to all agents, participating into the negotiation. Bartolini et al. [10], say that most multi-agents systems today use a single negotiation protocols, which is usually a finite state machine, hard-coded to all the agents, leading to an inflexible environment, which can accept only agents designed for it. To overcome this inflexibility they propose a generic interaction protocol which can be parameterized with different negotiation rules and give different negotiation mechanisms. The rules can be exchanged among agents that are able to inform their peers, which protocol they wish to use. However, the focus of our work is not to protocol design and we believe the protocol we use is a good solution for our demonstrator.

As we have already said, our protocol is a finite state machine with discrete states and transition. The protocol is depicted in Fig. 2. S0 to S6 represent the states of a negotiation and E is the final state in which there is an agreement, or a failure of agreement between the participants. Send and Recv predicates represent the interactions which cause state transitions. To clarify the function of the protocol we give an example. If the sequence of transitions is the following: S0→

S1→S2→S6→E, that means that the agent initially sends a call for proposal message (CFP) to the other negotiating agent (S0→ S1), then he receives a propose message (S1→S2) and after the evaluation he decides to send an accept message (S2→S6). Lastly he receives an accept message and the negotiation terminates successfully (S6→E). We make the convention that the participant that plays the role of the buyer starts the negotiation by posting a CFP message. So, while the protocol can be used as it is by a buyer, it needs a small modification for a seller. Particularly instead of the transition S0→ S1 there should be a transition S0→ S2 with label "Recv CFP".

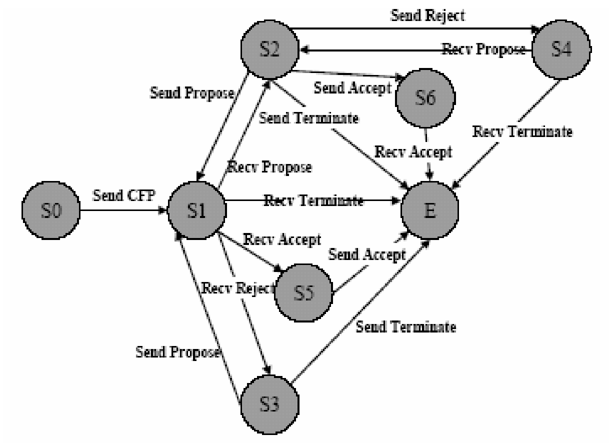


Fig.2. 1-1 Negotiation Protocol

5. The Negotiation Strategy: An Example

The strategy of a potential buyer or seller during a negotiation scenario is very critical for the outcome of the encounter. Every strategy is indeed designed in line with a particular protocol. As we have already seen, there is a plethora of strategies classified according to different criteria. We based the strategies we used on the work of Tsang et al. [11]. They define the simple constrained bargaining game between one buyer and one seller. Some of the most important assumptions are that the seller is constraint by the cost and the number of days within which he has to sell, while the buyer is constrained by his utility and the number of days within which he has to buy. In addition, neither the buyer nor the seller has information about the constraints of the other. The players make alternative bids with the seller to bid first and they can bid only once per day. An agreement is reached when both buyer and seller bid for the same price. Finally, according to the assumptions, if a deal cannot be made before a player runs out of time the negotiation terminates. Tsang et al. propose a number of different strategies both for buyer and sellers. For our buyer we adopted the simple buyer strategy, whose characteristics are described in Table 1. We have made the following changes for the strategy of the buyer: Firstly, buyer and seller are not obliged to make only one offer per day but one offer per negotiation step. Negotiation step is handled by the protocol and increases each time a player (buyer or seller) has made an offer and subsequently has received a counteroffer or another message. So, we speak about time to buy (TTB) and time to sell (TTS), measured in negotiation steps. Secondly, except for the offer-acceptance criterion we have added a check during the offer submission to avoid results which are against the benefit of the player. Thirdly, we incorporated into the strategy parameters relevant to the protocol like the state of the negotiation and the step of the negotiation .Lastly, an agreement is reached when both buyer and seller send an "agree" message.

Strategy Name	**First Bid Algorith m**	**Offer-Acceptance Criterion**	**Last Day BiDDING**	**General Bid Algorith m**
Simple Buyer	Utility/DTB	Counteroffer +Minimum Profit< Utility	As usual	Bid half way between previous bid price and utility

Table1. Buyer's Strategy Characteristics

For the buyer, participating into the negotiation, we used the modified strategy of Tsang et al. and we expressed it in defeasible logic. For the seller we used a strategy hard-coded in java to demonstrate that agents with diferrent architecture can interact without any problems. Seller's strategy is quite similar with that of buyer, except for the general bidding strategy. Seller decreases his offer by a fixed amount while buyer increases his offer in a linear fashion .

We express the buyer's strategy in defeasible logic (see Fig.3). The predicates we use are the following:

- Step(s): The step of the negotiation. When a buyer or seller sends a message and then receives another one the step is increased by one
- Counteroffer(c): The offer which a buyer or seller receives from the opponent
- Min_profit(mp): The minimum profit the buyer seeks after buying the product
- Utility(u): The utility of the buyer if he buys the product
- Ttb(ttb): The time (negotiation steps) the buyer has at his disposal in order to buy the product
- State(st): The current state of the negotiation according to the protocol. The possible states are:

1(S1)→The buyer has already sent a CFP or a PROPOSE message
2(S2)→ The buyer has already received a PROPOSE message
3(S3)→ The buyer has already received a REJECT message
…

- First_bid(fb): The initial bid of the buyer
- Previous_bid(prb): The previous bid of the buyer

R1: State(st), Counteroffer(c), Min_profit(mp),Utility(u), st=2, c+mp ≤ u/2 ⇒ ACCEPT_PROPOSAL
R2: State(st), Counteroffer(c), Min_profit(mp),Utility(u), st=2, c+mp > u ⇒ ~ACCEPT_PROPOSAL
R3: State(st), st=5 ⇒ ACCEPT_PROPOSAL
R4: Step(s), Counteroffer(c), Min_profit(mp),Utility(u), First_Bid(fb), State(st), s=0 ,st=2,
u/2 < c+mp ≤ u, bid=fb ⇒PROPOSE(bid)
R5: Step(s), Ttb(ttb), State(st), Counteroffer(c), Min_profit(mp),tility(u),First_Bid(fb), Previous_bid(prb), 0<s≤ttb-1, st=2, u/2 < c+mp ≤ u, prb=0, bid=fb ⇒PROPOSE(bid)
R6: Step(s), Ttb(ttb), State(st), Counteroffer(c), Min_profit(mp),Utility(u), Previous_bid(prb),
0<s<ttb-1,st=2, u/2 < c+mp ≤ u, prb!=0, bid=(u-prb)/2+prb ⇒PRELIM_PROPOSE(bid)
R7: Step(s), Ttb(ttb), State(st), Previous_bid(prb), Utility(u), 0<s<ttb, st=3,
bid=(u-prb)/2+prb⇒PRELIM_PROPOSE(bid)
R8: Min_profit(mp),Utility(u), PRELIM_PROPOSE(bid) ⇒ PROPOSE(bid)
R9: Min_profit(mp),Utility(u),PRELIM_PROPOSE(bid), bid>u-mp ⇒ ~PROPOSE(bid)
R10: Min_profit(mp),Utility(u),PRELIM_PROPOSE(bid) ,bid>u-mp,new_bid=utility-min_profit⇒ PROPOSE(new_bid)

R9>R8

Fig. 3. Buyer's Strategy in Defeasible Logic

Rules R1, R2, and R3 define the conditions for the acceptance or rejection of a proposal. More specific Rule R1 states that if the current state of the negotiation is S2 (agent has received a "propose" message) and if opponent's offer plus the minimum profit is less or equal to half the utility, the counteroffer is accepted in all cases.

R2 describes the case in which opponent's offer plus the minimum profit is greater than his utility and the counteroffer is rejected. Finally R3 defines that if the current state of the negotiation is S5 (agent has received an "accept" message) he also sends an "accept" message. There are two levels for the bidding policy. Bidding of first step and general bidding policy.

R4 states that if the negotiation is at state S2 and at the first step, the utility divided by the ttb is offered. According to R5 if the current state of the negotiation is S2 (Agent has received a "propose" message) *but* it has not made one, it offers the utility divided by the ttb.

R6 defines that if the current state of the negotiation is S2 (agent has received a "propose" message) *and* it has made an offer in the previous step, it increases linearly its offer, which derives from the type:

$$bid = \frac{utility - previous\_bid}{2} + previous\_bid$$

R7 describes that if the current state of the negotiation is S3 (agent has received a "reject" message) it also offers the above bid.

R8 defines that if R7 or R6 is true then the computed amount for the bid is to be offered. However, R9 checks if the bid to be offered is lower than the utility minus the minimum profit and if it is not, R10 is fired. Rule R9 is an additional check that ensures that the offered amount of money for the product, is not against the benefit of the buyer.

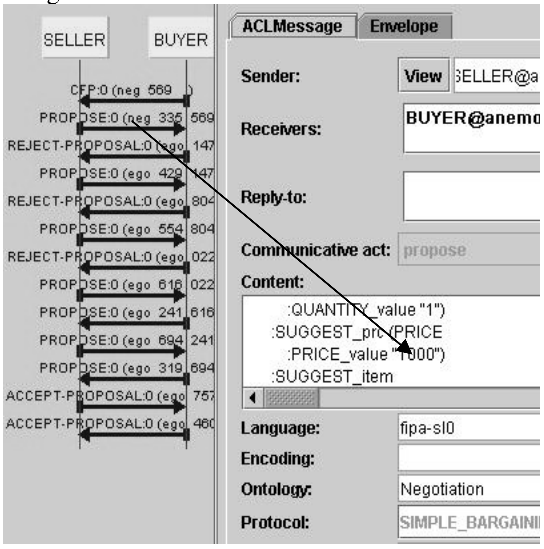
6. Negotiation Trace

At this section we demonstrate the operation of the system and we examine a negotiation trace between a buyer agent and a seller agent. JADE platform provides a special-purpose agent which is called sniffer. Sniffer can monitor the exchanged messages of two or more agents in the agent platform. The specific parameters of the negotiation are given in the next table. We examine the trace from the point of view of the buyer. The parameters of the negotiation are summarized in Table 2.

BUYER	SELLER
Ttb=5	Tts=10
Minimum Profit = 100	Minimum Profit = 100
Utility = 1000	Maximum Profit = 800
	Bid decrement = 40
	Cost = 200

Table 2. Negotiation Trace Parameters

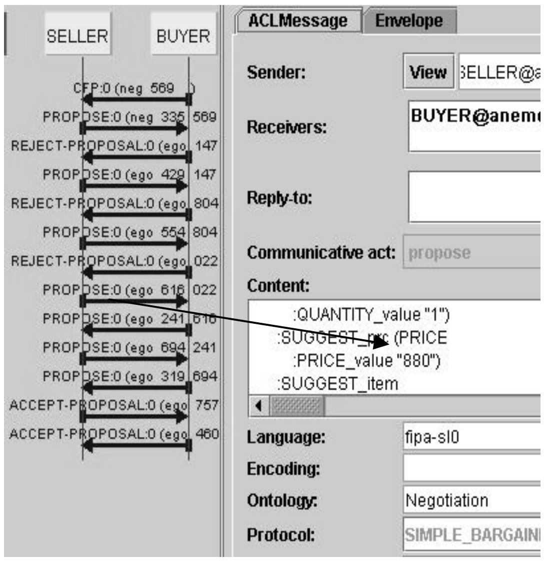
As we can see in Fig. 4 , buyer initially issues a "call for proposal message" (CFP) and seller responds with a "propose". The amount proposed by the seller is 1000. According to the condition of R2 of buyer's strategy, as long as the relation c+mp > u is true, the buyer keeps rejecting the offer.


SELLER BUYER
CFP:0 (neg 569)
PROPOSE:0 (neg 335 569
REJECT-PROPOSAL:0 (ego 147
PROPOSE:0 (ego 429 147
REJECT-PROPOSAL:0 (ego 804
PROPOSE:0 (ego 554 804
REJECT-PROPOSAL:0 (ego 022
PROPOSE:0 (ego 616 022
PROPOSE:0 (ego 241 616
PROPOSE:0 (ego 694 241
PROPOSE:0 (ego 319 694
ACCEPT-PROPOSAL:0 (ego 757
ACCEPT-PROPOSAL:0 (ego 460

ACLMessage Envelope
Sender: View SELLER@a
Receivers: BUYER@anemo
Reply-to:
Communicative act: propose
Content:
:QUANTITY_value "1")
:SUGGEST_prc (PRICE
:PRICE_value "1000")
:SUGGEST_item
Language: fipa-sl0
Encoding:
Ontology: Negotiation
Protocol: SIMPLE_BARGAINI

Fig.4. 1-1 Negotiation Trace a

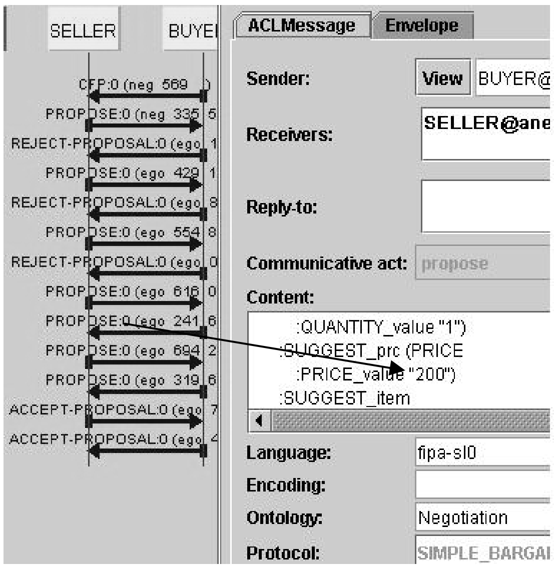
As buyer has rejected seller's offer, at the next step of the negotiation, seller decreases its offered amount by 40 (bid decrement) and waits for the response of the buyer . As buyer regards (according to his strategy) that the amount of 960 is too high, continues to reject the offer, without issuing a counteroffer. At this point we must say that although the protocol allows both for a counteroffer or a rejection of a proposal, the decision lies to the agent and is expressed through the strategy. Seller decreases his offer by another 40, offering 920. Buyer still rejects seller's offer and the latter subsequently offers 880 (Fig.5).


SELLER BUYER
CFP:0 (neg 569)
PROPOSE:0 (neg 335 569
REJECT-PROPOSAL:0 (ego 147
PROPOSE:0 (ego 429 147
REJECT-PROPOSAL:0 (ego 804
PROPOSE:0 (ego 554 804
REJECT-PROPOSAL:0 (ego 022
PROPOSE:0 (ego 616 022
PROPOSE:0 (ego 241 616
PROPOSE:0 (ego 694 241
PROPOSE:0 (ego 319 694
ACCEPT-PROPOSAL:0 (ego 757
ACCEPT-PROPOSAL:0 (ego 460

ACLMessage Envelope
Sender: View SELLER@a
Receivers: BUYER@anem
Reply-to:
Communicative act: propose
Content:
:QUANTITY_value "1")
:SUGGEST_prc (PRICE
:PRICE_value "880")
:SUGGEST_item
Language: fipa-sl0
Encoding:
Ontology: Negotiation
Protocol: SIMPLE_BARGAINI

Fig.5. 1-1 Negotiation Trace b

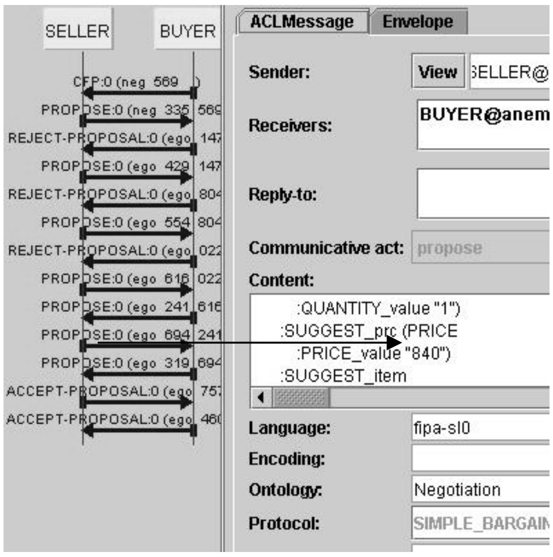
As the relation 880+100>1000 (c+mp>u) is now false, R5 fires and buyer buyer offers his initial offer which is the amount 1000/5 (u/TTb). This is depicted in Fig.6.


SELLER BUYEI
CFP:0 (neg 569)
PROPOSE:0 (neg 335 5
REJECT-PROPOSAL:0 (ego 1
PROPOSE:0 (ego 429 1
REJECT-PROPOSAL:0 (ego 8
PROPOSE:0 (ego 554 8
REJECT-PROPOSAL:0 (ego 0
PROPOSE:0 (ego 616 0
PROPOSE:0 (ego 241 6
PROPOSE:0 (ego 694 2
PROPOSE:0 (ego 319 6
ACCEPT-PROPOSAL:0 (ego 7
ACCEPT-PROPOSAL:0 (ego 4

ACLMessage Envelope
Sender: View BUYER@
Receivers: SELLER@ane
Reply-to:
Communicative act: propose
Content:
:QUANTITY_value "1")
:SUGGEST_prc (PRICE
:PRICE_value "200")
:SUGGEST_item
Language: fipa-sl0
Encoding:
Ontology: Negotiation
Protocol: SIMPLE_BARGAI

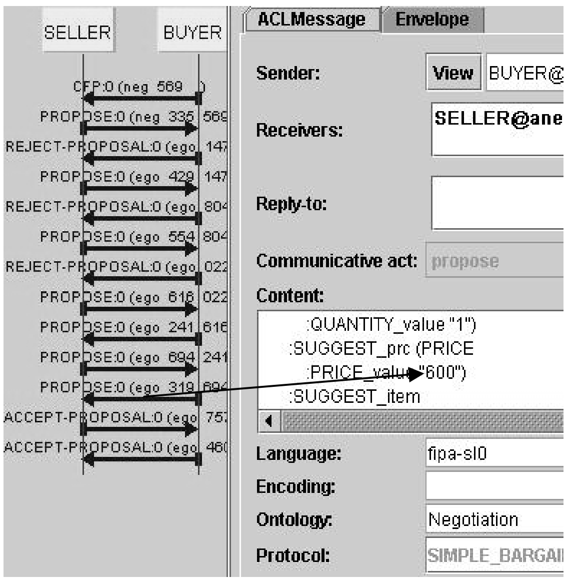
Fig.6. 1-1 Negotiation Trace c

At he next step, seller offers 840 and waits the buyer for its response (Fig. 7). R6 now fires and buyer offers the amount 600 (Fig. 8). Seller in turn issues an accept message and the negotiation terminates. At this point we must notice that if seller's last message was not "accept-proposal", buyer's control module would issue a "cancel" as the ttb would exceed 5.


SELLER BUYER
CFP:0 (neg 569)
PROPOSE:0 (neg 335 569
REJECT-PROPOSAL:0 (ego 147
PROPOSE:0 (ego 429 147
REJECT-PROPOSAL:0 (ego 804
PROPOSE:0 (ego 554 804
REJECT-PROPOSAL:0 (ego 022
PROPOSE:0 (ego 616 022
PROPOSE:0 (ego 241 616
PROPOSE:0 (ego 694 241
PROPOSE:0 (ego 319 694
ACCEPT-PROPOSAL:0 (ego 757
ACCEPT-PROPOSAL:0 (ego 460

ACLMessage Envelope
Sender: View SELLER@
Receivers: BUYER@anem
Reply-to:
Communicative act: propose
Content:
:QUANTITY_value "1")
:SUGGEST_prc (PRICE
:PRICE_value "840")
:SUGGEST_item
Language: fipa-sl0
Encoding:
Ontology: Negotiation
Protocol: SIMPLE_BARGAIN

Fig.7. 1-1 Negotiation Trace d



```
SELLER        BUYER
CFP:0 (neg 569 )
PROPOSE:0 (neg 335 569
REJECT-PROPOSAL:0 (ego 147
PROPOSE:0 (ego 429 147
REJECT-PROPOSAL:0 (ego 804
PROPOSE:0 (ego 554 804
REJECT-PROPOSAL:0 (ego 022
PROPOSE:0 (ego 616 022
PROPOSE:0 (ego 241 616
PROPOSE:0 (ego 694 241
PROPOSE:0 (ego 319 694
ACCEPT-PROPOSAL:0 (ego 757
ACCEPT-PROPOSAL:0 (ego 460
```

```
Sender: BUYER@
Receivers: SELLER@ane
Reply-to:
Communicative act: propose
Content:
      :QUANTITY_value "1")
   :SUGGEST_prc (PRICE
      :PRICE_value "600")
   :SUGGEST_item
Language: fipa-sl0
Encoding:
Ontology: Negotiation
Protocol: SIMPLE_BARGAI
```

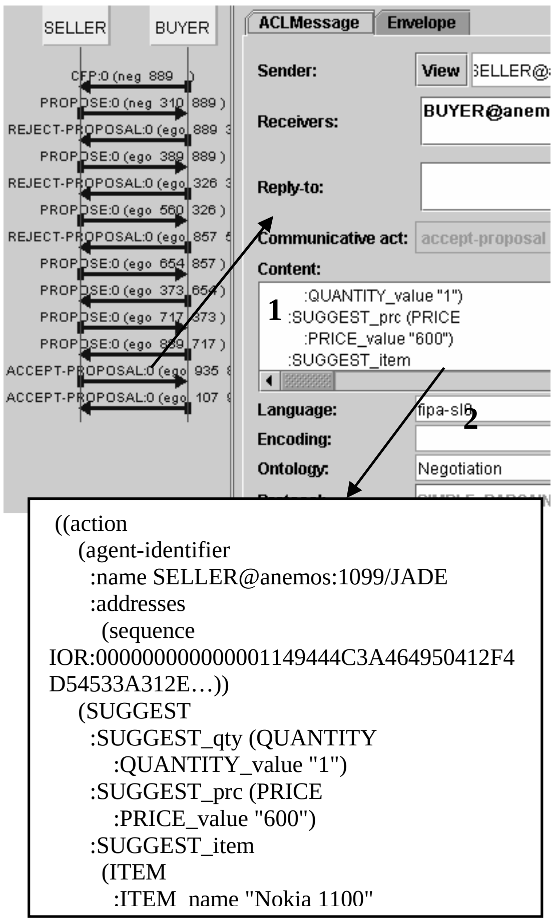
Fig.8. 1-1 Negotiation Trace e

Now lets analyze the structure of exhangesd messages. FIPA ACL messages [4] are built up of three layers of languages [13]. a) Elements of the world are defined in an ontology. b) An agent's intention to describe or alter the world is expressed by a communicative act or speech-act such as INFORM and c) statements about the world are expressed by means of a Content Language. In order for agents to be able to reason about the effects of their communication ,ACL messages should be inserted into proper Agent Interaction Protocol [5] which describe allowed sequences of actions among agents.

The traces were taken with the help of sniffer agent, provided by JADE platform. At the left-hand side one can see all the interactions among the buyer and the seller agent, We analyze the interaction which is indicated by the arrow 12 directed from the seller to buyer. The ACL which corresponds to interaction 12 of Fig.9, is indicated by arrow no.1. The *communicative act* (or speech-act) of this ACL message is "accept-proposal". The *ontology*, which both buyer and seller share is called "Negotiation" and the used *interaction* protocol is called "Simple-Bargaining". The used content language was FIPA SL0 and the the content of the message is indicated by arow no.2. The negotiation is about a black NOKIA 1100 mobile phone, which finally seller accepts to sell to buyer for 600 money units.

7. Related Work

Many efforts in the area of automated negotiation have focused on applying game theory techniques, to design an optimal strategy for a given protocol [22].



```
SELLER        BUYER
CFP:0 (neg 889 )
PROPOSE:0 (neg 310 889 )
REJECT-PROPOSAL:0 (ego 889 3
PROPOSE:0 (ego 389 889 )
REJECT-PROPOSAL:0 (ego 326 3
PROPOSE:0 (ego 560 326 )
REJECT-PROPOSAL:0 (ego 857 5
PROPOSE:0 (ego 654 857 )
PROPOSE:0 (ego 373 654 )
PROPOSE:0 (ego 717 373 )
PROPOSE:0 (ego 889 717 )
ACCEPT-PROPOSAL:0 (ego 935 8
ACCEPT-PROPOSAL:0 (ego 107 9
```

```
Sender: SELLER@
Receivers: BUYER@anem
Reply-to:
Communicative act: accept-proposal
Content:
      :QUANTITY_value "1")
 1 :SUGGEST_prc (PRICE
      :PRICE_value "600")
   :SUGGEST_item
Language: fipa-sl0
Encoding:          2
Ontology: Negotiation
```

```
((action
   (agent-identifier
     :name SELLER@anemos:1099/JADE
     :addresses
       (sequence
IOR:000000000000001149444C3A464950412F4
D54533A312E…))
   (SUGGEST
     :SUGGEST_qty (QUANTITY
        :QUANTITY_value "1")
     :SUGGEST_prc (PRICE
        :PRICE_value "600")
     :SUGGEST_item
       (ITEM
        :ITEM_name "Nokia 1100"
```

Fig.9. 1-1 Negotiation Trace f and Message Content in FIPA SL

Although this approach yields interesting results under simplifying assumptions (e.g. known valuations, risk-neutral attitudes, computationally unbounded agents), it is difficult or impossible to apply them in some realistic situations.

Reeves et al. [23] use Courteous Logic Programming (CLP) to express knowledge about user preferences, constraints, and negotiation structures. The authors do not address the issue of specifying bidding strategies, but rather that of determining the set of auctions and other negotiations that need to be undertaken in order to transform a contract template into an executable contract. Interestingly, Defeasible Logic (DL) is more expressive than CLP, in the sense that it fully supports stratified theories [17].

More recently, another auction management server called *eAuctionHouse* has been released [24]. It supports combinatorial auctions and mobile agents that can issue bids on behalf of a user. However, the user is not allowed to specify his own bidding strategy: he has to choose between a set of predefined ones. The same problem has the project Kasbah [25] which is an "older system".

8. Conclusions and Future Work

In this paper we described the design and implementation of a system for automated agent negotiation, based on declarative strategies. Such negotiations can be expected to play a key role on the semantic web. Planned future work includes: a) Designing graphical user interfaces, b) Comparing our approach of executing the declarative negotiation strategies to the alternative of translating the strategies into Java (in which case the declarative strategies are used in the specification/analysis phase, but not in the execution phase) and finally c) Applying our system in electronic marketplaces and in legal negotiation platforms.

References

[1]. Pattie Maes, Robert H.Guttman and Alexandros G. Moukas (1999). "Agents That Buy and Sell". *Communications of the ACM Vol. 4 March 1999.*

[2]. Minghua He, Nicholas R. Jennings, and Ho-Fung Leung (2003). "On Agent-Mediated Electronic Commerce". *IEEE transactions on knowledge and data engineering Vol. 15, No 4 July/August 2003.*

[3]. Agent Construction Tools. http://www.agentbuilder.com/AgentTools/index.html

[4]. Yannis Labrou, Tim Finin, Yun Peng (1999). "Agent Communication Languages: The current Landscape". *IEEE Intelligent Systems March/April 1999.*

[5]. FIPA Interaction Protocols Specification. http://www.fipa.org/repository/ips.php3

[6]. JADE Project. http://jade.cselt.it/

[7]. F.Bellifemine, G Caire, A.Poggi, G. Rimassa (2003). " JADE A White Paper" *Telecom Italia EXP magazine Vol 3, No 3 September 2003*

[8]. Marlon Dumas, Guido Governatori, Arthur H.M ter Hofstede, Phillipa Oaks (2002). "A Formal Approach to Negotiating Agents Development". *Elsevier Science – Electronic Commerce Research and Applications Vol. 1,Issue 2 Summer 2002 pp. 193-207*

[9]. Stanley Y. W. Su, Chunbo Huang and Joachim Hammer (2000). "A Replicable Web-based Negotiation Server for E-Commerce". *In proceedings of the 33rd Hawaii International Conference on System Sciences.*

[10]. Claudio Bartolini, Chris Preist and Nicholas R. Jennings (2002). "A Generic Software Framework for Automated Negotiation". *In proceedings of the first International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) Bologna Italy 2002*

[11]. Edward Tsang and Tim Gosling (2002). "Simple Constrained Bargaining Game". *In proceedings of the Distributed Constrained Satisfaction Workshop-First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) Bologna Italy 2002*

[12]. Nick Bassiliades, Grigoris Antoniou, Ioannis Vlahavas (2004). "A Defeasible Logic Reasoner for the Semantic Web" . *Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)*, G. Antoniou, H. Boley (Ed.), Springer-Verlag, Hiroshima, Japan, 8 Nov. 2004

[13]. Chris van Aart, Ruurd Pels, Giovanni Caire, Federico Bergenti (2002). "Creating and Using Ontologies in Agent Communication". *Telecom Italia EXP magazine Vol 2, No 3 September 2002.*

[14]. D. Nute (1994). Defeasible logic. In *Handbook of logic in artificial intelligence and logic programming (vol. 3): nonmonotonic reasoning and uncertain reasoning*. Oxford University Press

[15]. B. N. Grosof (1997). "Prioritized conflict handing for logic programs". In *Proc. of the 1997 International Symposium on Logic Programming*, 197-211

[16]. G. Antoniou, D. Billington, G. Governatori and M.J. Maher (2001). "Representation results for defeasible logic". ACM Transactions on Computational Logic 2, 2 (2001): 255 - 287

[17]. G. Antoniou, M. J. Maher and D. Billington (2000). Defeasible Logic versus Logic Programming without Negation as Failure. *Journal of Logic Programming* 41,1

[18]. N.R.Jennings, S.Parsons, C.Sierra, P.Faratin (2000). "*Automated Negotiation* ". *In proceedings of 5th Int. Conf. on the Practical Application of Intelligent Agents and Multi- Agent Systems (PAAM-2000)*

[19]. G. Governatori and M.J. Maher (2000)."An argumentation-theoretic characterization of defeasible logic". In *Proc. 14th European Conference on Artificial Intelligence*

[20]. H. Raiffa (1982). "*The Art and Science of Negotiation*". Harvard University Press

[21]. M.J. Maher, A. Rock, G. Antoniou, D. Billington and T. Miller (2001). "Efficient defeasible reasoning systems". *International Journal of Tools with Artificial Intelligence* 10(4), 483-501

[22]. J.S. Rosenschein and Gilad Zlotkin. Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers. MIT Press, Readings MA, USA, 1994.

[23]. Daniel M. Reeves, Michael P. Wellman, Benjamin N.Grosof, and Hoi Y. Chan (2000). "Automated negotiation from declarative contract descriptions". In proceedings of 17th *National Conference on Artificial Intelligence, Workshop on Knowledge-Based Electronic Markets(KBEM),* Austin, Texas, July 30–31 2000.

[24]. Tuomas SandHolm (2002). eMediator: "A Next Generation Electronic Commerce Server". Journal of Computational Intelligence Vol.18, No. 4 2002

[25]. A. Chavez, D. Dreilinger, R. Guttman, and P. Maes (1997). "A real-life experiment in creating an agent marketplace". In Proc. of the 2nd Int. Conference on The Practical Applications of Agents and Multi-Agent Technology (PAAM), London, UK, 1997.