GROUP DECISION AND NEGOTIATIONS 2007
VOLUME II
MT. TREMBLANT-MONTREAL, CANADA

An Argumentation-Driven Model for Autonomous and Secure Negotiation

Jamal Bentahar[1], Jihad Labban[1], Bernard Moulin[2]

[1] Concordia University, Concordia Institute for Information Systems Engineering, Canada
bentahar@ciise.concordia.ca
j_labban@encs.concordia.ca

[2] Laval University, Department of Computer Science and Software Engineering, Canada
bernard.moulin@ift.ulaval.ca

Abstract.
The purpose of this paper is to propose a formal description of an argumentation-based negotiation protocol between autonomous agents. This protocol is designed to be secure and computationally efficient. The security is ensured by allowing agents to reason about trust. The computational efficiency is achieved by specifying the protocol as a set of simple logical rules that agents can easily combine. These latter are specified as a set of computational dialogue games about which agents can reason. The protocol converges by checking the termination conditions. The paper addresses also the implementation issues of our protocol using an agent-oriented platform equipped with logical programming.

Keywords: Agent communication, negotiation, argumentation, trust and reputation

1 Introduction

Software autonomous agents are a promising technology for developing a new generation of flexible and intelligent applications. In fact, agents are associated with a powerful set of metaphors and techniques for designing, implementing, and verifying complex, distributed systems such as electronic trading and distributed business process [24]. Although there is little consensus about the definition of a software agent, it is generally held that agents are autonomous pieces of software, able to take initiative in order to satisfy some goals [27].

In this area of agent-based computing, it is extensively recognized that communication between software agents is one of the major topics of research [1,2,5,9,11]. All the applications of multi-agent systems [18,24,27,28] ranging from digital libraries through cooperative engineering to electronic commerce, have one thing in common: the agents operating in these systems have to communicate. These systems consist of multiple autonomous agents that interact with each other in order to solve some problems. If a problem is particularly complex, large, or unpredictable, the only way it can

reasonably be addressed is to develop a number of functionally specific and modular components (agents) which are able to solve a particular problem aspect [23]. This decomposition allows each agent to use the most appropriate paradigm to solve its particular problem.

Negotiation is one important type of interaction that is gaining increasing prominence in the agent community, whereby agents with conflicting interests, but a desire to cooperate, try to come to a mutually acceptable agreement on the division of scarce resources [8,14,16,20]. Resources can be commodities, services, time, money, etc. Resources are scarce in the sense that competing claims over them cannot be fully satisfied simultaneously. The problem of resource negotiation via communication in a distributed setting is core to a number of applications, particularly the emerging semantic grid computing-based applications such as e-science and e-business. To allow agents to autonomously negotiate with each other, we propose to equip them with argumentation and logical reasoning capabilities.

Argumentation can be defined as a process for the interaction of different arguments for and against some conclusion [6,10,15]. Argumentation has been researched extensively in the last decade, especially for inference, decision support, dialogue, and negotiation. Agents can use argumentation in their communication in order to justify their negotiation stances and influence other agents' negotiation stances. An important branch of argumentation is formal dialectics [1,2,3,6,10,14]. In its most abstract form, a dialectical model is a set of arguments and a binary relation representing the attack-relation (and indirectly the defence relation) between the arguments in a dialogical process. Consequently, dialectical models are relevant for automated negotiation, in which agents should persuade each other. In this paper, we propose to use dialectical argumentation to assist agents to reach a decision and convince each other. A single agent may use such an argumentation to perform its reasoning because it needs to make decisions in highly dynamic environments, considering interacting preferences and utilities. Also, this argumentation can help multiple agents to interact rationally, by giving and receiving reasons for conclusions and decisions, within an enriching dialectical process that aims at reaching mutually agreeable joint decisions. During negotiation, agents can establish a common knowledge of each other's commitments and find compromises, and persuade one another to make commitments.

Several computational frameworks for argumentative inferences have been developed in the literature [1,6,8,14,15,19]. However, only few proposals have considered the implementation and application issues of argumentation-based negotiation. Another challenging problem for automated negotiation that has not been deeply addressed is the security engineering. The problem of securing negotiation systems in a distributed setting is of great utility for a number of applications such as e-commerce, e-business and Web services-based applications.

The objective of this paper is to address the specification and security issues of agent-based negotiation. We propose a new computational model for efficient and secure negotiation using an argumentation-driven framework and reputation-based approach. The idea is to be able to participate in flexible negotiations and to trust interacting agents within a multi-agent system. This is because in order to share resources and allow mutual access, involved agents in e-infrastructures need to establish a framework of trust that establishes what they each expect of the other. Such a framework must allow one entity to assume that a second entity will behave exactly as the first entity expects. Current approaches to security fail to adequately address the e-computing challenges. These approaches are mostly centralized on mechanisms such as digital certificates, and thus are particularly vulnerable to attacks. This is because if some authorities who are trusted implicitly are compromised, then there is no other check in the system. By contrast, in the decentralized approach we propose in this paper and where the principals maintain trust in each other for more reasons than a single certificate, any "invaders" can cause limited harm before being detected.

Paper Overview The rest of the paper is organized as follows. In Section 2, we introduce the conceptual framework of the agent communication mechanism. We discuss the theoretical considerations, agent architecture, and argumentation framework. In Section 3, we address the formal specification of our dialogue game-based negotiation protocol using persuasive argumentation. We

present the protocol form, the specification of each dialogue game, and the protocol termination and dynamics. In Section 4, we address the security issue by describing an efficient trust model. In Section 5, we describe the implementation of a prototype allowing us to illustrate how the specification of dialogue games is implemented. Finally, in Section 6, we draw some conclusions and we identify some directions for future work.

2 Conceptual Framework

2.1 Theoretical Consideration

Agent communication is related to several disciplines: philosophy of language, social psychology, artificial intelligence, logic, mathematics, etc. In this domain, in order to be able to negotiate, solve conflicts of interest, cooperate, find proofs, agents need not only exchange single messages, but also take part in conversations with other agents. A conversation is defined as a coherent sequence of utterances. The term "coherent" means that the information conveyed by an utterance is related to the information conveyed by the other utterances in a conversation. For example, if p is the information conveyed by an utterance, the information conveyed by the next one can be the acceptance, refusal, challenge, attack, etc. of p. Indeed, if agents communicate by exchanging isolated messages, the resulting communication is extremely poor and agents cannot participate in complex interactions such as negotiations, which are formed by a sequence of utterances.

To consider the conversational aspect of agent communication, we use action logic to specify the communicative acts. In addition, to capture the formal semantics of such communicative acts, our approach is based on the notion of "social commitments" [2,7,12]. A social commitment SC is an engagement made by an agent (the debtor), that some fact is true or that something will be done. This commitment is directed to a set of agents (creditors). A commitment is an obligation in the sense that the debtor must respect and behave in accordance with this commitment. Social commitments are a powerful representation to model multi-agent interactions. Commitments provide a basis for a normative framework that makes it possible to model agents' communicative behaviours. This framework has the advantage of being expressive because all speech act types can be represented by commitments [5]. Commitment-based protocols enable the content of agent interactions to be represented and reasoned about [12].

In order to model the dynamics of conversations, we interpret a speech act SA as an action performed on a commitment or on its content. A speech act is an abstract act that an agent, the speaker, performs when producing an utterance U and addressing it to another agent, the addressee. In the negotiation dialogue game protocol that we specify in Section 3, the actions that an agent can perform on a commitment are: $Act \in \{Create, Withdraw\}$. Create mean that making an offer, and Withdraw means that withdrawing it. The actions that an agent can perform on commitment content are: $Act\text{-}content \in \{Accept, Refuse, Challenge, Defend, Attack, Justify\}$. In our framework, a speech act is interpreted either as an action applied to a commitment when the speaker is the debtor, or as an action applied to its content when the speaker is the debtor or the creditor [2]. Formally, a speech act can be defined as follows:

Definition 1 (Speech Act)

$$SA(Ag_1, Ag_2, U) \triangleq Act(Ag_1, SC(Ag_1, Ag_2, p)) \; \& \mid Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$$

where $i, j \in \{1, 2\}$ *and* $(k = i \text{ or } k = j)$, p is the commitment content.

The definiendum $SA(Ag_1, Ag_2, U)$ is defined by the definiens $Act(Ag_1, SC(Ag_1, Ag_2, p))$ as an action performed by the debtor Ag_1 on its commitment. The definiendum is defined by the definiens $Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$ as an action performed by an agent Ag_k (the debtor or the creditor) on the commitment content.

Example 1

Let us consider the following utterances:

U1: *Quebec is the capital of Canada.*
U2: *No, the capital of Canada is Ottawa.*

The utterance *U1* leads the debtor to create a commitment whose content is "*Quebec is the capital of Canada*". On the other hand, the utterance *U2* highlights a creditor's action on this content that is in this case a refusal.
Formally:

$SA(Ag_1, Ag_2, U1) \triangleq$

$Create(Ag_1, SC(Ag_1, Ag_2, (Capital(Canada, Québec))))$

$SA(Ag_2, Ag_1, U2) \triangleq$

$Refuse\text{-}content(Ag_2, SC(Ag_1, Ag_2, (Capital(Canada, Quebec))))$

$\& \; Create(Ag_2, SC(Ag_2, Ag_1, (Capital(Canada, Ottawa))))$

2.2 Architecture

The conceptual framework architecture we propose is characterized by capturing simultaneously 1) the mental aspect of agents taking part in the conversation (beliefs, desires, goals…), 2) the social aspect reflecting the context in which these agents communicate and the social commitments and norms, and 3) the reasoning aspect which is essential to be able to take part in coherent conversations. The reasoning part is based upon an argumentation system enabling agents to justify the facts on which they are committed and to justify their actions on commitments. The combination of these three aspects is necessary because producing social commitments (i.e. public utterances) reflects the agents' mental states on which agents should reason before committing in a conversation and during its unfolding.

The communication model consists of three layers: the conversation layer, the argumentative layer, and the cognitive layer. This stratification in layers is supported by the abstraction levels. The conversation layer is directly observable and highlights speech acts the agents perform. These acts are not performed in an isolated way, but within a particular conversation. The argumentative layer is used to correctly manage the social commitments and arguments that are related to the conversation. Finally, the cognitive layer is used to take into account the agents' private mental states, the social relations, and other elements that agents use to be able to communicate. In this paper we focus on the second layer.

In order to allow negotiating agents to use suitably the communication model, this latter must be compatible with the agent architecture. Thus, we propose a negotiating agent model consisting of a mental model, a social model, and a reasoning model (Figure 1). The mental model includes beliefs, desires, goals, etc. The social model captures the social concepts such as conventions, roles, commitments, etc. Commitments that agents make public by communication are related to the mental states via the reasoning model.

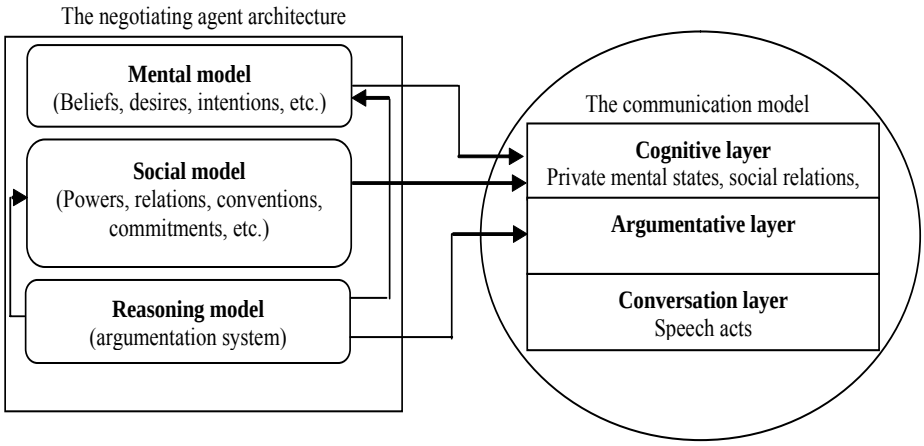


Figure 1. ***The conceptual framework***

2.3 Argumentation Framework

The agent's reasoning model is specified using an argumentation system. Such a system essentially includes a logical language L, a definition of the argument concept, and a definition of the attack relation between arguments. The use of a logical language enables negotiating agents to use a logic-based reasoning in order to effectively reason about arguments in terms of inferring and justifying conclusions, and attacking and defending arguments. Hereafter we define the concepts that will be used in the negotiation. Here Γ indicates a possibly inconsistent knowledge base with no deductive closure. $\vdash$ stands for classical inference and $\equiv$ for logical equivalence.

Definition 2 (Argument) An argument is a pair (H, h) where h is a formula of L and H a sub-set of Γ such that : i) H is consistent, ii) $H \vdash h$ and iii) H is minimal, so no subset of H satisfying both i and ii exists. H is called the support of the argument and h its conclusion. We use the notation: $H = Support(Ag, h)$ to indicate that agent Ag has a support H for the conclusion h.

Example 2

Let $\Gamma = \{a, a \to b, c \to \neg b, c\}$. Then, $(\{a, a \to b\}, b)$ and $(\{a \to b\}, \neg a \vee b)$ are two arguments.

Definition 3 (Attack Relation) Attack is a binary relation between two arguments. Let (H_1, h_1) and (H_2, h_2) be two arguments, (H_1, h_1) attacks (H_2, h_2) is denoted: $(H_1, h_1) \nRightarrow (H_2, h_2)$. $(H_1, h_1) \nRightarrow (H_2, h_2)$ iff $H_2 \equiv \neg h_1$.

Example 3

Let $\Gamma = \{a, a \to b, c \to \neg b, c, \neg b \to \neg d, \neg c\}$. Then, the argument $(\{a, a \to b\}, b)$ attacks the argument $(\{c, c \to \neg b, \neg b \to \neg d\}, \neg d)$. Also, the argument $(\{\neg c\}, \neg c)$ attacks the argument $(\{c, c \to \neg b, \neg b \to \neg d\}, \neg d)$.

In fact, before committing to some fact h being true (i.e. before making an offer by creating a commitment whose content is h), the speaker agent should use its argumentation system to build an argument (H, h). On the other side, the addressee agent must use its own argumentation system to

select the answer it will give (i.e. to decide about the appropriate manipulation of the content of an existing commitment). For example, an agent Ag_1 accepts the commitment content h proposed by another agent if Ag_1 has an argument for h. If Ag_1 has an argument neither for h, nor for $\neg h$, then it challenges h.

In our framework, we distinguish between arguments that an agent has (private arguments) and arguments that this agent used in its conversation (public arguments). Thus, we use the notation: $S = Create\_Support(Ag, SC(Ag_1, Ag_2, p))$ to indicate the set of commitments S created by agent Ag to support the content of $SC(Ag_1, Ag_2, p)$. This support relation is transitive i.e.:

$(SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_1))$
$\wedge\ SC(Ag_1, Ag_2, p_1) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0)))$
$\Rightarrow SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0))$

Surely, an argumentation system is essential to help agents to justify their negotiation stances and influence other agents' negotiation stances. However, reasoning on other social attitudes should be taken into account in order to explain the agents' decisions. In our approach, agents can reason about trust and use trustworthiness to decide, in some cases, about the acceptance of arguments. This trust-based reasoning is essential for securing negotiation settings.

3 Logical Rules for Negotiation Protocol

3.1 Computational Dialogue Games

Agent communication protocols specify the rules of interaction governing a dialogue between autonomous agents in a multi-agent system. These protocols are patterns of behaviour that restrict the range of allowed follow-up utterances at any stage during a dialogue. Unlike protocols used in distributed systems, agent communication protocols must take into account the fact that artificial agents are autonomous and proactive. These protocols must be flexible enough and must also be specified using a more expressive formalism than traditional formalisms such as finite state machines. Indeed, logic-based protocols seem an interesting way [3,11].

Computational dialogue games [3,8,17] aim at offering more flexible protocols. This is achieved by combining different games to construct complete and more complex protocols. Dialogue games are declarative specifications that govern communication between autonomous agents. They are interactions between players, in which each player moves by performing utterances according to a pre-defined set of roles. In this paper, we propose to formalize these games as a set of logical rules about which agents can reason in order to decide which game to play and how to combine games. Indeed, protocols specified using finite state machines or Petri nets are not flexible in the sense that agents must respect the whole protocol from the beginning to the end. For this reason, we propose to specify these protocols by small computational dialogue games that can be considered as conversation policies that can be logically put together. Formally, we define a computational dialogue game as follows:

Definition 4 (Computational Dialogue Game) Let Action Ag_1 and Action Ag_2 be two communicative actions performed by Ag_1 and Ag_2 respectively, and let *Cond* be a formula from the logical language L. A computational dialogue game is a logical rule indicating that if Ag_1 performs Action Ag_1, and that *Cond* is satisfied, then Ag_2 will perform Action Ag_2 afterwards. This rule is expressed as follows:

$$Action\_Ag_1 \xrightarrow{Cond} Action\_Ag_2$$

Cond is expressed in terms of the possibility of generating an argument from an agent's argumentation system.

3.2 Negotiation Dialogue Games

Our negotiation protocol is specified as a set of computational dialogue games. In accordance with our commitment-based approach, the game moves are considered as actions that agents apply to commitments and to their contents (see **Definition 1**). Because we suppose that we have always two agents Ag_1 and Ag_2, a SC whose content is p will be denoted in the rest of this paper $SC(p)$. We use the notation: $p \frown Arg\_Sys(Ag_1)$ to denote the fact that a propositional formula p can be generated from the argumentation system of Ag_1 denoted $Arg\_Sys(Ag_1)$. The formula $\neg(p \frown Arg\_Sys(Ag_1))$ indicates the fact that p cannot be generated from Ag_1's argumentation system. A propositional formula p can be generated from an agent's argumentation system, if this agent can find an argument supporting p. To simplify the formalism, we use the notation $Act'(Ag, SC(p))$ to indicate the action that agent Ag performs on the commitment $SC(p)$ or on its content (Act'$\in$ {Create, Withdraw, Accept, Challenge, Refuse}). For the actions related to the argumentation relations, we write $Act\text{-}Arg(Ag, [SC(q)], SC(p))$. This notation indicates that Ag defends (resp. attacks or justifies) the content of $SC(p)$ by the content of $SC(q)$ ($Act\text{-}Arg \in \{Defend, Attack, Justify\}$). In a general way, we use the notation $Act'(Ag, S)$ to indicate the action that Ag performs on the set of commitments S or on the contents of these commitments, and the notation $Act\text{-}Arg(Ag, [S], SC(p))$ to indicate the argumentation-related action that Ag performs on the content of $SC(p)$ using the contents of S as support. We also introduce the notation $Act\text{-}Arg(Ag, [S], S')$ to indicate that Ag performs an argumentation-related action on the contents of a set of commitments S' using the contents of S as supports.

We distinguish two types of dialogue games: entry game and chaining games. The entry game allows the two agents to open the negotiation. The chaining games make it possible to combine dialogue games during the negotiation. The negotiation terminates when the exit conditions are satisfied (Figure 2).

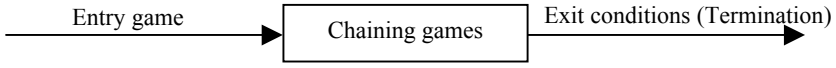
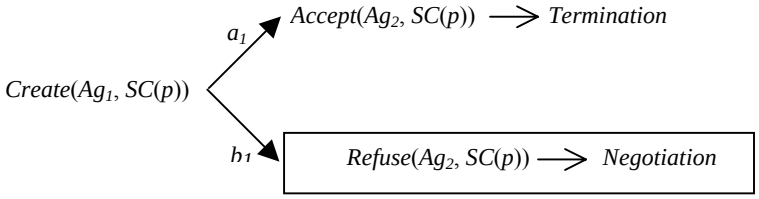


Figure 2. ***The general form of the protocol***

A Entry Game

The entry dialogue game describing the entry conditions in our negotiation protocol about an offer represented by a propositional formula p is described as follows (Specification 1):

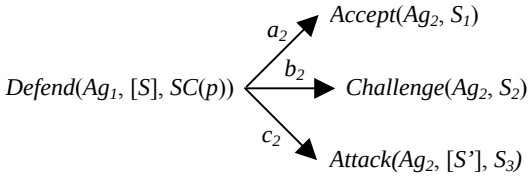


where a_1 and b_1 are two conditions specified as follows:
$a_1 = p \frown Arg\_Sys(Ag_2)$
$b_1 = \neg p \frown Arg\_Sys(Ag_2)$

If Ag_2 has an argument for p, then it accepts the offer p (the content of $SC(p)$) and the conversation terminates as soon as it begins (Condition a_1). The negotiation starts when Ag_2 refuses the Ag_1's offer p because it has an argument against p (condition b_1).

B Defense Game

Once the two agents opened the negotiation, the initiator must defend its point of view in order to persuade the addressee. Consequently, a defence game should be played. Our protocol is specified in such a way that the negotiation dynamics starts by playing a defence game. We call this type of negotiation persuasive negotiation. This game is specified as follows (Specification 2):



where:

$S = \{SC(p_i) / i = 0,...,n\}$, p_i are propositional formulas.

$\bigcup_{i=1}^{3} S_i = S \bigcup SC(p)$, $S_i \cap S_j = \varnothing$, $i, j = 1,...,3 \,\&\, i \neq j$

By definition, *Defend*(Ag_1, [S], $SC(p)$) means that Ag_1 creates S in order to defend the content of $SC(p)$. Formally:

$$Defend(Ag_1, [S], SC(p)) \triangleq (Create(Ag_1, S) \wedge S = Create\_Support(Ag_1, SC(p)))$$

We consider this definition as an assertional description of the *Defend* action. We propose similar definitions for *Attack* and *Justify* actions which are not presented in this paper.

This specification indicates that according to the three conditions (a_2, b_2 and c_2), Ag_2 can accept a subset S_1 of S, challenge a subset S_2 and attack a third subset S_3. Sets S_i and S_j are mutually disjoint because Ag_2 cannot, for example, both accept and challenge the same commitment content. Accept, Challenge and Attack a set of commitment contents are defined as follows:

$$Accept(Ag_2, S_1) \triangleq (\forall i, SC(p_i) \in S_1 \Rightarrow Accept(Ag_2, SC(p_i)))$$
$$Challenge(Ag_2, S_2) \triangleq (\forall i, SC(p_i) \in S_2 \Rightarrow Challenge(Ag_2, SC(p_i)))$$
$$Attack(Ag_2, [S'], S_3) \triangleq \forall i, SC(p_i) \in S_3 \Rightarrow \exists S'_j \subseteq S', Attack(Ag_2, [S'_j], SC(p_i))$$

where: $S'_j = S'$. This indication means that any element of S' is used to attack one or more elements of S_3.

The conditions a_2, b_2 and c_2 are specified as follows:

$$a_2 = \forall i, SC(p_i) \in S_1 \Rightarrow p_i \cap Arg\_Sys(Ag_2)$$
$$b_2 = \forall i, SC(p_i) \in S_2 \Rightarrow (\neg(p_i \cap Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \cap Arg\_Sys(Ag_2)))$$
$$c_2 = \forall i, SC(p_i) \in S_3 \Rightarrow \exists S'_j \subseteq S', Content(S'_j) = Support(Ag_2, \neg p_i)$$

where $Content(S'_j)$ indicates the set of contents of the commitments S'_j.

C Challenge Game

The challenge game is specified as follows (Specification 3):

$$Challenge(Ag_1, SC(p)) \xrightarrow{a_3} Justify(Ag_2, [S], SC(p))$$

where the condition a_3 is specified as follows:

$a_3 = (Content(S) = Support(Ag_2, p))$

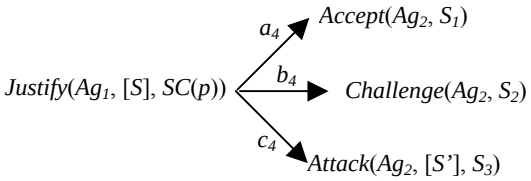
In this game, the condition a_3 is always true. The reason is that in accordance with the commitment semantics, an agent must always be able to defend the commitment it created [5].

D Justification Game

For this game we distinguish two cases:

Case1. SC(p) ∉ S

In this case, Ag_1 justifies the content of its commitment $SC(p)$ by creating a set of commitments S. As for the *Defend* action, Ag_2 can accept, challenge and/or attack a subset of S. The specification of this game is as follows (Specification 4):



where:

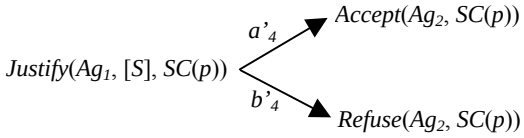
$S = \{SC(p_i) / i = 0,...,n\}$, p_i are propositional formulas.

$\bigcup_{i=1}^{3} S_i = S \bigcup SC(p)$, $S_i \bigcap S_j = \varnothing$, $i, j = 1,...,3 \;\&\; i \neq j$

$a_4 = a_2$, $b_4 = b_2$, $c_4 = c_2$

Case2. {SC(p)} = S

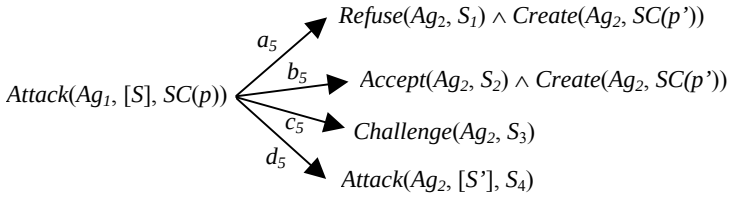
In this case, the justification game has the following specification (Specification 5):



Ag_1 justifies the content of its commitment $SC(p)$ by itself (i.e. by p). This means that p is part of Ag_1's knowledge. Only two moves are possible for Ag_2: 1) accept the content of $SC(p)$ if Ag_1 is a trustworthy agent for Ag_2 (a'_4), 2) if not, refuse this content (b'_4). Ag_2 cannot attack this content because it does not have an argument against p. The reason is that Ag_1 plays a justification game because Ag_2 played a challenge game.

E Attack Game

The attack game is specified as follows (Specification 6):



where:
$S=\{SC(p_i)/i=0,...,n\}$, p_i are propositional formulas.
$\bigcup_{i=1}^{4} S_i = S \bigcup SC(p)$, $Card(S_1) = 1$, $S_i \bigcap S_j = \varnothing$, $i, j = 1,...,4 \,\&\, i \neq j$

The conditions a_5, b_5, c_5 and d_5 are specified as follows:
$a_5 = \exists i, SC(p_i) \in Create\_Support(Ag_2, SC(\neg q))$
where S1 = $\{SC(q)\}$
$b_5 = \forall i, SC(p_i) \in S_2 \Rightarrow p_i \frown Arg\_Sys(Ag_2)$
$c_5 = \forall i, SC(p_i) \in S_3 \Rightarrow (\neg(p_i \frown Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \frown Arg\_Sys(Ag_2)))$
$d_5 = \forall i, SC(p_i) \in S_4 \Rightarrow \exists S'_j \subseteq S'$,
$Content(S'j) = Support(Ag_2, \neg p_i) \wedge \nexists k, SC(p_k) \in$ Create_Support$(Ag_2, SC(\neg p_i))$

Ag_2 refuses Ag_1's argument if Ag_2 already attacked this argument. In other words, Ag_2 refuses Ag_1's argument if Ag_2 cannot attack this argument since it already attacked it, and it cannot accept it or challenge it since it has an argument against this argument. We have only one element in S_1 because we consider a refusal move as an exit condition. The acceptance and the challenge actions of this game are the same as the acceptance and the challenge actions of the defence game. In the case of refusal and acceptance, Ag_2 can make a counteroffer *p'* by creating a new commitment. In this situation, Ag_1 will play an entry game by accepting or refusing the counteroffer. Finally, Ag_2 attacks Ag_1's argument if Ag_2 has an argument against Ag_1's argument, and if Ag_2 did not attack Ag_1's argument before. In d_5, the universal quantifier means that Ag_2 attacks all Ag_1's arguments for which it has an against-argument. The reason is that Ag_2 must act on all commitments created by Ag_1. The temporal aspect (the past) of a_5 and d_5 is implicitly integrated in $Create\_Support(Ag_2, SC(\neg q))$ and Create_Support$(Ag_2, SC(\neg p_i))$.

F Termination
The protocol terminates either by a final acceptance or by a refusal. A final acceptance means that the two agents agree on a consensus. Because it is prohibited for the two agents to play the same move during the negotiation, the termination of the protocol is ensured.

3.3 Protocol Dynamics

The persuasive negotiation dynamics is described by the chaining of a finite set of dialogue games: *entry game*, *acceptance move*, *refusal move*, *defence*, *challenge*, *attack* and *justification games*. These games can be combined in a sequential and parallel way (Figure 3).

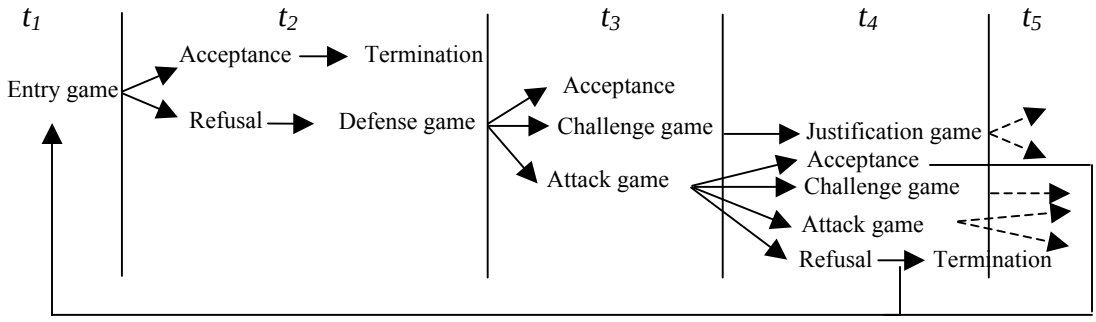


Figure 3. ***The persuasion dialogue dynamics***

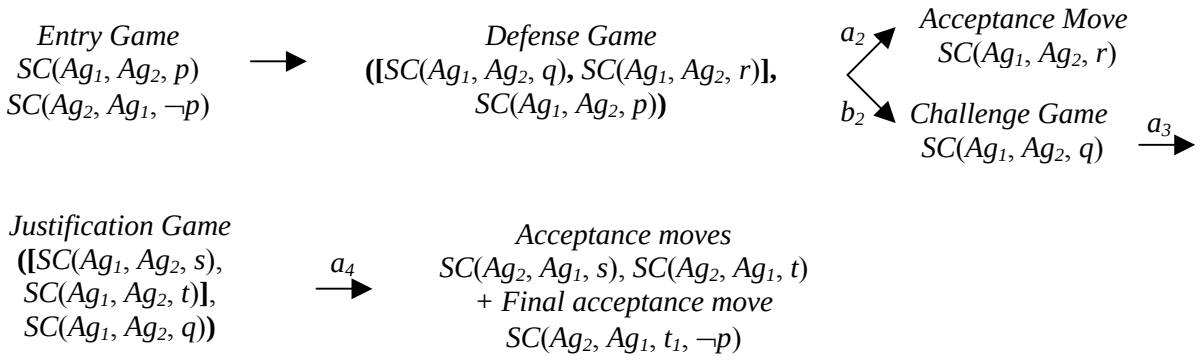
After Ag_1's defence game at moment t_2, Ag_2 can, at moment t_3, accept a part of the arguments presented by Ag_1, challenge another part, and/or attack a third part. These games are played in parallel. At moment t_4, Ag_1 answers the challenge game by playing a justification game and answers the attack game by playing an acceptance move, a challenge game, another attack game, and/or a refusal move. After the acceptance and the refusal, the player can propose a counteroffer, after which a renegotiation will start. The persuasive negotiation dynamics continues until the exit conditions become satisfied (final acceptance or a refusal). From our specifications, it follows that our protocol plays the role of the dialectical proof theory of the argumentation system.

Example 4

Let us consider the following dialogue to illustrate the dialogue games and protocol dynamics presented in this section:

Ag_1: Newspapers can publish information I (p)*.*
Ag_2: I don't agree with you.
Ag_1: They can publish information I because it is not private (q)*, and any public information can be published* (r)*.*
Ag_2: Why is information I public?
Ag_1: Because it concerns a Minister (s)*, and information concerning a Minister is public* (t)*.*

The letters on the left of the utterances are the propositional formulae that represent the propositional contents. Agent Ag_1's KB contains: ($[q, r], p$), ($[s, t], q$) and ($[u], u$). Agent Ag_2's KB contains: ($[\neg p], \neg p$). The combination of the dialogue games that allows us to describe the persuasion dialogue dynamics is as follows:



4 Security Consideration

In the previous section, we described the formal specification of the autonomous negotiation protocol. In this section, we address the second aspect: trust-based security. In recent years, several models of trust have been developed in the context of multi-agent systems [13,21,22,29]. However, these models are not designed to trust argumentation-based negotiating agents. Their formulations do not take into account the elements we use in our negotiation approach (accepted and refused arguments, satisfied and violated commitments). In addition, these models have some limitations regarding the inaccuracy of the collected information from other agents. In this section we present our argumentation and probabilistic-based model to trust negotiating agents that overcome some limitations of these models.

4.1 Formulation

Let A be the set of agents. We define an agent's trustworthiness in a distributed setting as a probability function as follows:

$$TRUST : A \times A \times D \rightarrow [0,1]$$

This function associates to each agent a probability measure representing its trustworthiness in the domain D according to another agent. Let X be a random variable representing an agent's trustworthiness. To evaluate the trustworthiness of an agent Ag_b, an agent Ag_a uses the records of its interactions with Ag_b. Equation 1 indicates how to calculate this trustworthiness as a probability measure (number of successful outcomes / total number of possible outcomes).

$$TRUST(Ag_b)_{Ag_a} = \frac{Nb\_Arg(Ag_b)_{Ag_a} + Nb\_C(Ag_b)_{Ag_a}}{T\_Nb\_Arg(Ag_b)_{Ag_a} + T\_Nb\_C(Ag_b)_{Ag_a}} \quad (1)$$

$TRUST(Ag_b)_{Ag_a}$ indicates the trustworthiness of Ag_b according to Ag_a's point of view.

$Nb\_Arg(Ag_b)_{Ag_a}$ is the number of Ag_b s' arguments that are accepted by Ag_a.

$Nb\_C(Ag_b)_{Ag_a}$ is the number of satisfied commitments made by Ag_b towards Ag_a.

$T\_Nb\_Arg(Ag_b)_{Ag_a}$ is the total number of Ag_b s' arguments towards Ag_a.

$T\_Nb\_C(Ag_b)_{Ag_a}$ is the total number of commitments made by Ag_b towards Ag_a.

All these commitments and arguments are related to the domain D. The basic idea is that the trust degree of an agent can be induced according to how much information acquired from him has been accepted as belief in the past. Because all the factors of Equation 1 are related to the past, this information number is finite.

Trustworthiness is a dynamic characteristic that changes according to the interactions taking place between Ag_a and Ag_b. This supposes that Ag_a knows Ag_b. If not, or if the number of interactions is not sufficient to determine this trustworthiness, the consultation of other agents becomes necessary.

As proposed in [29], each agent has two kinds of beliefs when evaluating the trustworthiness of another agent: local beliefs and total beliefs. Local beliefs are based on the direct interactions between agents. Total beliefs are based on the combination of the different testimonies of other agents that we call witnesses. In our model, local beliefs are given by Equation 1. Total beliefs require studying how different probability measures offered by witnesses can be combined. We deal with this aspect in the following section.

4.2 Estimating Agent's Trustworthiness

Let us suppose that an agent Ag_a wants to evaluate the trustworthiness of an agent Ag_b with who he never (or not enough) interacted before. This agent must ask agents he knows to be trustworthy (we call these agents confidence agents). To determine whether an agent is confident or not, a trustworthiness threshold w must be fixed. Thus, Ag_b will be considered trustworthy by Ag_a iff $TRUST(Ag_b)_{Ag_a}$ is higher or equal to w. Ag_a attributes a trustworthiness measure to each confidence agent Ag_i. When he is consulted by Ag_a, each confidence agent Ag_i provides a

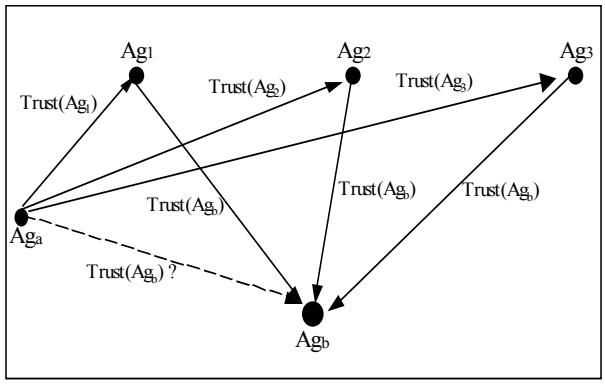


Figure 4. ***Problem of measuring Ag_b's trustworthiness by Ag_a***

trustworthiness value for Ag_b if Ag_i knows Ag_b. Confidence agents use their local beliefs to assess this value (Equation 1). Thus, the problem consists in evaluating Ag_b's trustworthiness using the trustworthiness values transmitted by confidence agents. Figure 4 illustrates this issue.

We notice that this problem cannot be formulated as a problem of conditional probability. Consequently, it is not possible to use Bayes' theorem or total probability theorem. The reason is that events in our problem are not mutually exclusive, whereas this condition is necessary for these two theorems. Probability values offered by confidence agents are not mutually exclusive since they are provided simultaneously.

To solve this problem, we must investigate the distribution of the random variable X representing the trustworthiness of Ag_b. Since X takes only two values: 0 (the agent is not trustworthy) or 1 (the agent is trustworthy), variable X follows a Bernoulli distribution ß(1, p). According to this distribution, we have Equation 2:

$$E(X) = p \tag{2}$$

Where $E(X)$ is the expectation of the random variable X and p is the probability that the agent is trustworthy. Thus, p is the probability that we seek. Therefore, it is enough to evaluate the expectation $E(X)$ to find $TRUST(Ag_b)_{Ag_a}$. however, this expectation is a theoretical mean that we must estimate. To this end, we can use the Central Limit Theorem (CLT) and the law of large numbers. The CLT states that whenever a random sample of size n (X_1,…X_n) is taken from any distribution with mean μ, then the sample mean $(X_1 + \ldots + X_n)/n$ will be approximately normally distributed with mean μ. As an application of this theorem, the arithmetic mean (average) $(X_1+\ldots+ X_n)/n$ approaches a normal distribution of mean μ, the expectation and standard deviation $\sigma/\sqrt{n}$ .Generally, and according to the law of large numbers, the expectation can be estimated by the weighted arithmetic mean.

Our random variable X is the weighted average of n independent random variables X_i that correspond to Ag_b's trustworthiness according to the point of view of confidence agents Ag_i. These

random variables follow the same distribution: the Bernoulli distribution. They are also independent because the probability that Ag_b is trustworthy according to an agent Ag_t is independent of the probability that this agent (Ag_b) is trustworthy according to another agent Ag_r. Consequently, the random variable X follows a normal distribution whose average is the weighted average of the expectations of the independent random variables X_i. The estimation of expectation $E(X)$ can be given by Equation 3.

$$M_0 = \frac{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} TRUST(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a}} \tag{3}$$

The value M_0 represents an estimation of $TRUST(Ag_b)_{Ag_a}$. Equation 3 does not take into account the number of interactions between confidence agents and Ag_b. This number is an important factor because it makes it possible to promote information coming from agents knowing more Ag_b. In addition, an other factor might be used to reflect the timely relevance of transmitted information. This is because the agent's environment is dynamic and may change quickly. The idea is to promote recent information and to deal with out-of-date information with less emphasis. Equation 4 gives us an estimation of $TRUST(Ag_b)_{Ag_a}$ if we take into account these factors and we suppose that all confidence agents have the same trustworthiness.

$$M_1 = \frac{\sum_{i=1}^{n} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b} TRUST(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}} \tag{4}$$

The factor $N(Ag_i)_{Ag_b}$ indicates the number of interactions between a confidence agent Ag_i and Ag_b. This number can be identified by the total number of Ag_b's commitments and arguments. The factor $TR(Ag_i)_{Ag_b}$ represents the timely relevance coefficient of the information transmitted by Ag_i about Ag_b 's trust (TR denotes timely relevance). In our model, we assess this factor by using the function defined in Equation 5. We call this function: the timely relevance function.

$$TR(\Delta t_{Ag_i}^{Ag_b}) = e^{-\lambda \ln(\Delta t_{Ag_i}^{Ag_b})} \tag{5}$$

Δt is the time difference between the current time and the time at which Ag_i updates its information about Ag_b's trust. λ is an application-dependant coefficient. The intuition behind this formula is to use a function decreasing with the time difference. Consequently, the more recent the information is, the higher is the timely relevance coefficient. The function ln is used for computational reasons when dealing with large numbers. Intuitively, the function used in Equation 5 reflects the reliability of the transmitted information. Indeed, this function is similar to the well known reliability function for systems engineering ($R(t) = e^{-\lambda t}$).

The combination of Equation 3 and Equation 4 gives us a good estimation of $TRUST(Ag_b)_{Ag_a}$ (Equation 6) that takes into account the four most important factors: (1) the trustworthiness of confidence agents according to the point of view of Ag_a; (2) the Ag_b's trustworthiness according to the point of view of confidence agents; (3) the number of interactions between confidence agents and Ag_b; and (4) the timely relevance of information transmitted by confidence agents. This number is an important factor because it makes it possible to highlight information coming from agents knowing more Ag_b.

$$M_2 = \frac{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b} TRUST(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}} \quad (6)$$

This Equation shows how trust can be obtained by merging the trustworthiness values transmitted by some mediators. This merging method takes into account the proportional relevance of each trustworthiness value, rather than treating them equally.

According to Equation 6, we have:

$$\forall i, TRUST(Ag_b)_{Ag_i} < w$$

$$\Rightarrow M < w.\frac{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}}{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}}$$

$$\Rightarrow M < w$$

5 Prototyping

In this section we describe a prototype implementation of the different dialogue games using the *Jack*^{TM} platform [25]. We chose this language for three main reasons:
1- It is an agent-oriented language offering a framework for multi-agent system development. This framework can support different agent models.
2- It is built on top of and fully integrated with the Java programming language. It includes all components of Java and it offers specific extensions to implement agents' behaviours.
3- It supports logical variables and cursors. These features are particularly helpful when querying the state of an agent's beliefs. Their semantics is mid-way between logic programming languages with the addition of type checking Java style and embedded SQL.

Our system consists of two types of agents: negotiating agents and trust model agents. These agents are implemented as *Jack*^{TM} agents, i.e. they inherit from the basic class *Jack*^{TM} Agent. Negotiating agents are agents that take part in the persuasive negotiation. Trust model agents are agents that can inform an agent about the trustworthiness of another agent.

According to the specification of the justification game, an agent Ag_2 can play an acceptance or a refusal move according to whether it considers that its interlocutor Ag_1 is trustworthy or not. If Ag_1 is unknown for Ag_2, Ag_2 can ask agents that it considers trustworthy for it to offer a trustworthiness assessment of Ag_1. From the received answers, Ag_2 can build a trustworthiness graph and measure the trustworthiness of Ag_1. This trustworthiness model is described in detail in [3].

To be able to take part in a persuasion dialogue, agents must possess knowledge bases that contain arguments. In our system, these knowledge bases are implemented as *Jack*^{TM} *beliefsets*. Beliefsets are used to maintain an agent's beliefs about the world. These beliefs are represented in a first order logic and tuple-based relational model. The logical consistency of the beliefs contained in a beliefset is automatically maintained. The advantage of using beliefsets over normal Java data structures is that beliefsets have been specifically designed to work within the agent-oriented paradigm.

Our knowledge bases (KBs) contain two types of information: arguments and beliefs. Arguments have the form ([*Support*], *Conclusion*), where Support is a set of propositional formulas and Conclusion is a propositional formula. Beliefs have the form ([*Belief*], *Belief*) i.e. Support and Conclusion are identical. The meaning of the propositional formulas (i.e. the ontology) is recorded in a beliefset whose access is shared between the two agents.

To open a dialogue game, an agent uses its argumentation system. The argumentation system allows this agent to seek in its knowledge base an argument for a given conclusion or for its negation ("against argument"). For example, before creating a commitment *SC*(*p*), an agent must find an

argument for p. This enables us to respect the commitment semantics by making sure that agents can always defend the content of their commitments.

Agent communication is done by sending and receiving messages. These messages are events that extend the basic *Jack*^{TM} event: *MessageEvent* class. *MessageEvents* represent events that are used to communicate with other agents. Whenever an agent needs to send a message to another agent, this information is packaged and sent as a *MessageEvent*. A *MessageEvent* can be sent using the primitive: Send (Destination, Message). In our protocol, Message represents the action that an agent applies to a commitment or to its content, for example: Create (Ag_1, *SC* (*p*)), etc.

Our dialogue games are implemented as a set of events (*Message Events*) and plans. A plan describes a sequence of actions that an agent can perform when an event occurs. Whenever an event is posted and an agent chooses a task to handle it, the first thing the agent does is to try to find a plan to handle the event. Plans are methods describing what an agent should do when a given event occurs. Each dialogue game corresponds to an event and a plan. These games are not implemented within the agents' program, but as event classes and plan classes that are external to agents. Thus, each conversational agent can instantiate these classes. An agent Ag_1 starts a dialogue game by generating an event and by sending it to its interlocutor Ag_2. Ag_2 executes the plan corresponding to the received event and answers by generating another event and by sending it to Ag_1. Consequently, the two agents can communicate by using the same protocol since they can instantiate the same classes representing the events and the plans. For example, the event *Event_Attack_Commitment* and the plan *Plan_ev_Attack_commitment* implement the defence game. The architecture of our conversational agents is illustrated in Figure 5.

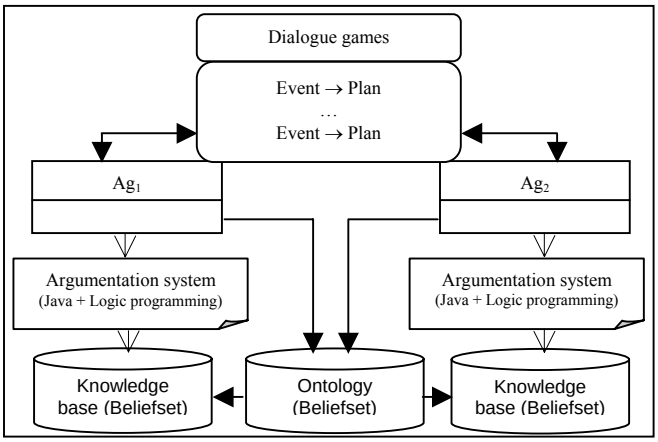


Figure 5. ***The architecture of conversational agents***

To start the entry game, an agent (initiator) chooses a goal that it tries to achieve. This goal is to persuade its interlocutor that a given propositional formula is true. For this reason, we use a particular event: BDI Event (Belief-Desire-Intention). BDI events model goal-directed behaviour in agents, rather than plan-directed behaviour. What is important is the desired outcome, not the method chosen to achieve it. This type of events allows an agent to pursue long term goals.

6 Conclusion and Future Work

The contribution of this paper is the proposition of a logical language for specifying autonomous and secure negotiation protocol between software agents. The proposed approach is based upon argumentation and trust. In this approach the agent's reasoning capabilities are linked to their ability to argue. The logical language has the advantage of being computationally efficient and expressing the public elements and the reasoning process that allows agents to choose an action among several

possible actions. Because our protocol is defined as a set of computational conversation policies, this protocol has the characteristic to be more flexible than the traditional protocols such as those used in FIPA-ACL. This flexibility results from the fact that these policies can be easily combined to produce complete and more complex protocols. We described the persuasive negotiation dynamics by the combination of five dialogue games and we presented the implementation of such a protocol using logical agent-based programming. Another contribution of this paper is the trust-based security model.

As an extension of this work, we intend to specify other protocols according to Walton and Krabbe's classification [26] using the same framework. Another interesting direction for future work is verifying these protocols using model checking techniques. The method we are investigating is an automata theoretic approach based on a tableau method [4]. This method can be used to verify the temporal and dynamic aspects of our protocol.

Furthermore, to improve the agents' negotiation abilities, agents can reason on the relevance of their offers and on the chance that their arguments can be accepted by the others. The idea is to go beyond the existing argumentation systems aiming simply to build an argument supporting a conclusion. The challenge is how to build a strong argument, ideally the stronger one. The idea we are investigating is to use a relevance-based reasoning in order to allow agents to optimize both their negotiation stances and the achievement of an agreement not only by justifying their choices, but by selecting the best choice that could be justified. Using rhetoric techniques combined with game theoretic and mechanism design strategies and some heuristics based on relevance theory seems a promising way. Agents can be equipped with "good" strategies enabling them to achieve their goals using an advanced reasoning on the utilities and the preferences of the other agents.

Acknowledgements

The two first authors are supported by the Faculty of Engineering and Computer Science at Concordia University (Start-up and SRT programs). We would also like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Amgoud, L., N. Maudet and S. Parsons (2000): "Modelling dialogues using argumentation", In Proc. of 4th Int. Conf. on Multi Agent Systems, pp. 31-38.
2. Bentahar, J., B. Moulin and B. Chaib-draa (2003): "Commitment and argument network", a new formalism for agent communication. In [9], pp. 146-165.
3. Bentahar, J., Moulin, B., and Chaib-draa, B. (2005): "Specifying and Implementing a Persuasion Dialogue Game using Commitment and Argument Network", Argumentation in Multi-Agent Systems, vol. (3366)(1), pp. 130-148. Springer.
4. Bentahar, J., Moulin, B., and Meyer, J-J.Ch. (2006): "A Tableau Method for Verifying Dialogue Game Protocols for Agent Communication", In Declarative Agent Languages and Technologies, vol. (3904)(3), pp. 223-244. Springer.
5. Bentahar, J., Moulin, B., Meyer, J-J. Ch., and Chaib-draa, B. (2004): "A logical model for commitment and argument network for agent communication (extended abstract)", In 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 792-799.
6. Brewka, G. (2001): "Dynamic argument systems: A formal model of argumentation processes based on situation calculus", Journal of Logic and Computation, 11(2) 257-282.
7. Castelfranchi, C. (1995): "Commitments: from individual intentions to groups and organizations", In Proc. of Int. Conf. on Multi Agent Systems, pp. 41-48.
8. Dastani, M., Hulstijn, J., and der Torre, L.V. (2000): "Negotiation protocols and dialogue games", In Proc. of Belgium/Dutch AI Conference, pp. 13-20.
9. Dignum, F. (Ed.) (2003). "Advances in Agent Communication", Int. Workshop on Agent Communication Languages. LNAI 2922, Springer.

10. Elvang-Goransson, M., Fox, J., and Krause, P. (1993): "Dialectic reasoning with inconsistent information", In Proc. of 9th Conf. on Uncertainty in Artificial Intelligence, pp. 114-121.
11. Endriss, U., Maudet, N., Sadri, F., and Toni, F. (2003): "Logic_based agent communication protocols", In [9], pp. 91-107.
12. Fornara, N. and Colombetti, M. (2003): "Protocol specification using a commitment based ACL", In [9], pp. 108-127.
13. Huynh, T.D., Jennings, N.R., and Shadbolt, N.R. (2006): "An integrated trust and reputation model for open multi-agent systems", In Journal of Autonomous Agents and Multi-Agent Systems, pp. 119-154.
14. Karunatillake, N.C., Jennings, N.R., Rahwan, I., Norman, T.J. (2005): "Argument-based negotiation in a social context", In proc. of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 1331-1332.
15. Kraus, S., Sycara, K.P. Evenchik, A. (1998): "Reaching agreements through argumentation: a logical model and implementation", In Artificial Intelligence 104(1-2):1-69.
16. Li, C., Giampapa, J.A., Sycara, K.P. (2006): "Bilateral negotiation decisions with uncertain dynamic outside options", In IEEE Transactions on Systems, Man, and Cybernetics, Part C 36(1):31-44
17. McBurney, P., and Parsons, S. (2002): "Games that agents play: A formal framework for dialogues between autonomous agents", In Journal of Logic, Language, and Information, 11(3):1-22.
18. Moulin, B., and Chaib-draa, B. (1996): "Distributed artificial intelligence: an overview", In Foundations of Distributed Artificial Intelligence, Jennings, N. and O'Hare, G. (eds.), Wiley, pp. 3-55.
19. Prakken, H. (2001): "Relating protocols for dynamic dispute with logics for defeasible argumentation", In Synthese (127):187-219.
20. Rahwan, I., Sonenberg, L., Jennings, N.R., McBurney, P. STRATUM. (2007): "A methodology for designing heuristic agent negotiation strategies", In Applied Artificial Intelligence, 21(10).
21. Ramchurn, S.D., Huynh, T.D., and Jennings, N.R. March (2004): "Trust in multi-agent systems", In The Knowledge Engineering Review, 19(1):1-25.
22. Sabater, J. (2003): "Trust and Reputation for Agent Societies", Ph.D. Thesis, Universitat Autμonoma de Barcelona.
23. Sycara, K. (1998): "Multiagent Systems", In AI Magazine, American Association for Artificial Intelligence, 19(2):79-92.
24. Sycara K., Pannu, A., Williamson, M., Zeng, D., and Decker, K. (1996): "Distributed intelligent agents", In IEEE Expert, 11(6):36-46.
25. The Agent Oriented Software Group. Jack 4.1. 2005. www.agent-software.com/
26. Walton, D.N., and Krabbe, E.C.W. (1995): "Commitment in dialogue: basic concepts of interpersonal reasoning", State University of New York Press, NY.
27. Wooldridge, M. and Jennings, N.R. (1995): "Intelligent agents: theory and practice", In The Knowledge Engineering Review, 10(2):115-152.
28. Wooldridge, M. (2003): "Reasoning about Rational Agents", The MIT Press: Cambridge, MA.
29. Yu. B, and Singh, M.P. (2003): "Detecting deception in reputation management", In Proc. of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM Press, pp. 73-80.