Group Decis Negot (2011) 20:411–435
DOI 10.1007/s10726-009-9163-0

An Argumentation-Driven Model for Flexible and Efficient Persuasive Negotiation

Jamal Bentahar · Jihad Labban

Published online: 21 April 2009

Abstract The purpose of this paper is to propose a formal description and implementation of a negotiation protocol between autonomous agents using persuasive argumentation. This protocol is designed to be simple and computationally efficient. The computational efficiency is achieved by specifying the protocol as a set of simple logical rules that software 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 discusses the formal properties of the protocol and addresses, as proof of concept, the implementation issues using an agent-oriented platform equipped with logical programming mechanisms.

Keywords Agent communication · Dialogue games · Negotiation · Argumentation

1 Introduction

Software autonomous agents are promising as technology for developing flexible and intelligent applications such as electronic trading, Web services and distributed business process (Bentahar et al. 2007; Shakshuki et al. 2007; Sycara et al. 1996). In the agent community, negotiation is one important type of interaction that is gaining increasing prominence, whereby agents with conflicting interests, try to achieve

J. Bentahar (✉) · J. Labban
Concordia Institute for Information Systems Engineering,
Concordia University, Montreal, Canada
e-mail: bentahar@ciise.concordia.ca

J. Labban
e-mail: j_labban@encs.concordia.ca

a mutually acceptable agreement on the division of scarce resources (Dastani 2000; Karunatillake 2005; Li 2006; Rahwan et al. 2007). In this paper, we propose a new formal protocol for agent negotiation based on the agents' persuasive capabilities. To take part in this type of negotiation, called *persuasive negotiation*, agents should persuade each other about their offers. Persuasive negotiation is a type of negotiation where one agent is trying to influence the behaviour of another agent using arguments supporting the proposed offers. Persuasive negotiation is a particular case of persuasive dialogues where the purpose is to achieve an agreement supported by an acceptable argument for all the participants (Walton and Krabbe 1995).

The proposed protocol is designed to be simple and computationally efficient. The computational efficiency is achieved by specifying the protocol as a set of simple logical rules that software 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 discusses the formal properties of the protocol and addresses, as proof of concept, the implementation issues using an agent-oriented platform equipped with logical programming mechanisms.

In fact, in the area of agent-based computing, it is extensively recognized that communication between software agents is one of the major topics of research (Bentahar et al. 2004; Dignum 2003; Endriss et al. 2003). All the applications of multi-agent systems ranging from digital libraries through cooperative engineering to electronic commerce, have one thing in common: the agents operating in these systems have to communicate (Moulin and Chaib-draa 1996; Shakshuki et al. 2007; Sycara et al. 1996; Wooldridge 2003). These systems consist of multiple interacting agents aiming to solve some problems. If the problem is particularly complex, large, or unpredictable, an interesting way to address it is to develop a number of functionally specific agents, which are able to solve particular problem aspects (Sycara 1998). This decomposition allows each agent to use the most appropriate paradigm to solve its particular problem.

The issue of resource negotiation via communication in distributed settings is core to a number of applications, particularly the emerging semantic grid computing-based applications such as e-science and e-business. Resources can be commodities, services, time, money, etc. These resources are scarce in the sense that competing claims over them cannot be fully satisfied simultaneously. 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 (Brewka 2001; Elvang-Goransson et al. 1993; Kraus et al. 1998). Argumentation has been researched extensively in the last decade, especially for inference, decision support, dialogue, and negotiation (Amgoud et al. 2000; Bench-Capon et al. 2009; Bentahar et al. 2005; McBurney et al. 2002; Rahwan et al. 2007). A single agent may use such an argumentation to perform its internal reasoning because it needs to make decisions in highly dynamic environments, considering interacting preferences and utilities. Also, 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

(Bentahar et al. 2005; Brewka 2001; Elvang-Goransson et al. 1993). 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.

We propose to use dialectical argumentation to assist agents to reach a decision and convince each other. We implement dialectical argumentation through a set of connected dialogue games. Dialogue games are interaction games in which each agent plays a move in turn by performing utterances according to a pre-defined set of rules (Karunatillake et al. 2009; McBurney and Parsons 2002). Argumentation-based dialogue games 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 combine and connect different dialogue games in order to establish a common knowledge of each other's commitments and find compromises, and persuade one another to make commitments.

Several formal frameworks for argumentative inferences have been developed in the literature (Amgoud et al. 2006; Dastani 2000; Kraus et al. 1998; Prakken 2001). 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 computational complexity. For concrete applications like e-business and Web services, this issue is of great utility, and the underlying algorithms should be tractable and computationally efficient. The objective of this paper is to address these two issues by specifying an efficient computational model for agent negotiation using an argumentation-driven framework.

1.1 Paper Overview

The rest of the paper is organized as follows. In Sect. 2, we introduce our conceptual framework of the agent communication mechanism. We discuss the theoretical considerations, agent architecture, and argumentation framework. In Sect. 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. The presented concepts are explained by illustrative examples. In Sect. 4, we analyse the protocol from a formal and computational point of view by discussing and proving the protocol properties and its computational complexity. In Sect. 5, we illustrate how the specification of our dialogue games is implemented by describing the development of a prototype serving as proof of concept. In Sect. 6, we present an overview of some significant proposals about dialogue games and argumentation-based negotiation and we compare our protocol with them. We also discuss how our protocol meets the requirements of formal dialogue games discussed in this overview. Finally, in Sect. 7, we draw some conclusions and identify some directions for future work.

2 Conceptual Framework

2.1 Theoretical Consideration

Although there is little consensus about the definition of software agents, it is generally held that agents are autonomous pieces of software, able to take initiatives in order to satisfy some goals, and are able to communicate (Wooldridge 2003). 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” (Bentahar et al. 2004; Castelfranchi 1995; Fornara and Colombetti 2003). 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 (Bentahar et al. 2004). Commitment-based protocols enable the content of agent interactions to be represented and reasoned about (Fornara and Colombetti 2003).

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 (Walton and Krabbe 1995). 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 Sect. 3, the actions that an agent can perform on a commitment are: $Act \in \{Create, Withdraw\}$. *Create* means that making an offer, and *Withdraw* means that withdrawing it. In this negotiation setting, the actions that an agent can perform on commitment content are: *Act-content* $\in \{Accept, Refuse, Challenge, Defend, Attack, Justify\}$. This set is built based on the philosophical analysis of negotiation dialogues as discussed in Walton and Krabbe (1995). We will show in Sect. 3 that this set is sufficient to model negotiation dialogue games. This set is also compatible with other proposals in the literature such as Amgoud et al. (2000), Karunatillake et al. (2009), McBurney et al. (2002).

The only difference is that these proposals use *Assert* to gather *Defend, Attack*, *Justify* in one action. However, using three different actions, namely*Defend, Attack*, *and Justify* instead of one allows us to clarify the speaker's position without considering the commitment content. In our framework, a speech act is interpreted as an action applied to a commitment when the speaker is the debtor, and/or as an action applied to its content when the speaker is the debtor or the creditor Bentahar et al. (2004). Formally, a speech act can be defined as follows:

Definition 1 (Speech Act)

$$SA\,(Ag_1, Ag_2,\ U) \triangleq Act\,(Ag_1, SC\,(Ag_1,\ Ag_2,\ p))$$
$$\&|Act - content\,\left(Ag_k, SC\,\left(Ag_i, Ag_j, p\right)\right)$$

where i, j $\in \{1, 2\}$ *and* ($k = i$ *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-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. Using first order logic syntax, this example can be formalized as follows:

$$SA(Ag_1,\ Ag_2, U1) \triangleq Create(Ag_1, SC(Ag_1, Ag_2, (Capital(Quebec, Canada))))$$
$$SA(Ag_2,\ Ag_1, U2) \triangleq$$
$$Refuse\text{-}content(Ag_2, SC(Ag_1,\ Ag_2, (Capital(Quebec, Canada))))$$
$$\&\,Create(Ag_2, SC(Ag_2, Ag_1, (Capital(Ottawa, Canada))))$$

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

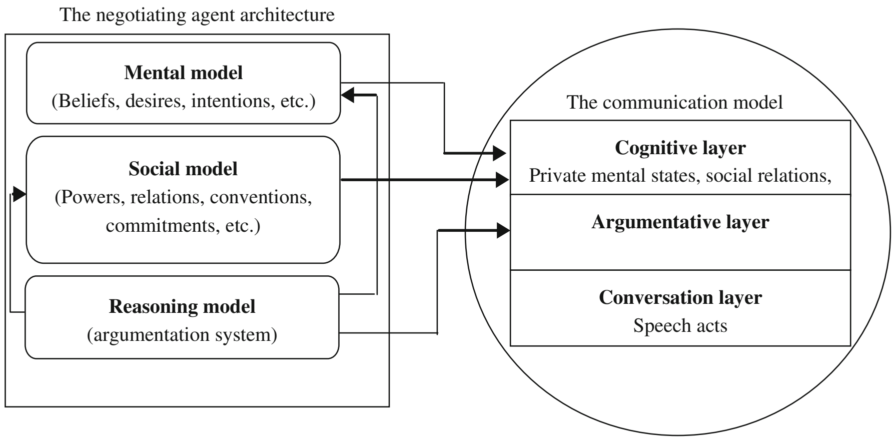

The negotiating agent architecture
Mental model
(Beliefs, desires, intentions, etc.)
Social model
(Powers, relations, conventions, commitments, etc.)
Reasoning model
(argumentation system)
The communication model
Cognitive layer
Private mental states, social relations,
Argumentative layer
Conversation layer
Speech acts

Fig. 1 The conceptual framework

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 model 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 (Fig. 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.

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. In our framework, the language is a set of propositional Horn clauses. A propositional Horn clause is a disjunction of literals with at most one positive literal $\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee c$ (also written as implication $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow c$. A propositional Horn formula is a conjunction of propositional Horn clauses. Hereafter we define the concepts that will be used in the persuasive negotiation where $\vdash$ stands for classical inference, and Γ, Γ_1, and Γ_2 indicate agents' knowledge bases. Each base is supposed to be consistent.

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 $\Gamma = \{a, a \rightarrow b, c \rightarrow \neg b, c\}$. Then, $(\{a, a \rightarrow b\}, b)$ and $(\{a \rightarrow b\}, \neg a \vee b)$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 over Γ_1 and Γ_2 respectively. (H_1, h_1) attacks (H_2, h_2) is denoted by: $(H_1, h_1) \nRightarrow (H_2, h_2)$. $(H_1, h_1) \nRightarrow (H_2, h_2)$ iff $H_2 \vdash \neg h_1$.

Example 3 Let $\Gamma_1 = \{a, a \rightarrow b, \neg c\}$ and $\Gamma_2 = \{c \rightarrow \neg b, c, \neg b \rightarrow \neg d\}$. Then, the argument $(\{a, a \rightarrow b\}, b)$ over Γ_1 attacks the argument $(\{c, c \rightarrow \neg b, \neg b \rightarrow \neg d\}, \neg d)$ over Γ_2. Also, the argument $(\{\neg c\}, \neg c)$ over Γ_1 attacks the argument $(\{c, c \rightarrow \neg b, \neg b \rightarrow \neg d\}, \neg d)$ over Γ_2.

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.:

$$\begin{aligned}
&(SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_1)) \\
&\wedge SC(Ag_1, Ag_2, p_1) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0))) \\
&\Rightarrow SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0))
\end{aligned}$$

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 Persuasive 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 (Bentahar et al. 2005; Endriss et al. 2003).

Computational dialogue games (Bentahar et al. 2005; Dastani 2000; McBurney and Parsons 2002) aim to offer 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 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 Dialogue Games for Persuasive Negotiation

Our persuasive 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

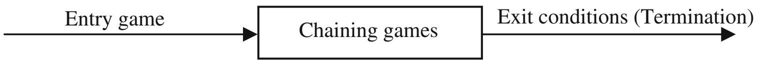


Fig. 2 The general form of the protocol

generated from the argumentation system of Ag_1 denoted $Arg\_Sys(Ag_1)$. The formula $\neg(p \triangle Arg\_Sys(Ag_1))$ 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-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 five types of dialogue games: *entry*, *defence*, *challenge*, *justification* and *attack*. The last four games (i.e. defence, challenge, justification and attack) constitute what we call *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. In fact, these four dialogue games can be combined in different ways. The negotiation terminates when the exit conditions are satisfied (Fig. 2). We assume that the negotiation process we consider in this paper is not function of time (i.e. unbounded times). Consequently, the exit conditions are not expressed in terms of time.

3.2.1 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):

$$\begin{tikzcd}
 & Accept(Ag_2, SC(p)) \arrow[r] & Termination \\
Create(Ag_1, SC(p)) \arrow[ur, "a_1"] \arrow[dr, "b_1"'] & & \\
 & \boxed{Refuse(Ag_2, SC(p)) \longrightarrow Negotiation} &
\end{tikzcd}$$

where a_1 and b_1 are two conditions specified as follows:

$a_1 = p \triangle Arg\_Sys(Ag_2)$
$b_1 = \neg p \triangle Arg\_Sys(Ag_2)$

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).

3.2.2 Defence 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):

$$\begin{tikzcd}
 & Accept(Ag_2, S_1) \\
Defend(Ag_1, [S], SC(p)) \arrow[ur, "a_2"] \arrow[r, "b_2"] \arrow[dr, "c_2"'] & Challenge(Ag_2, S_2) \\
 & Attack(Ag_2, [S'], S_3)
\end{tikzcd}$$

where:

$S = \{SC(p_i) | i = 0, \ldots, n\}$, p_i are propositional formulas.
$\bigcup_{i=1}^{3} S_i = S \cup SC(p)$, $S_i \cap S_j = \emptyset$, $\quad i, j = 1, \ldots, 3$ & $i \neq j$

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: $\bigcup_j S'{}_j = S'$. 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:

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

where: $Content(S'_j) = \{p_k | SC(p_k) \in S'_j\}$.$Content(S'_j)$ indicates then the set of contents p_k of the commitments $SC(p_k)$ included in S'_j.

3.2.3 Challenge Game

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

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

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 (Bentahar et al. 2004).

3.2.4 Justification Game

For this game we distinguish two cases:

Case 1 ($SC(p) \notin S$) 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):

$$\begin{tikzcd}
& \mathit{Accept}(Ag_2, S_1) \\
\mathit{Justify}(Ag_1, [S], SC(p)) \arrow[ur, "a_4"] \arrow[r, "b_4"] \arrow[dr, "c_4"'] & \mathit{Challenge}(Ag_2, S_2) \\
& \mathit{Attack}(Ag_2, [S'], S_3)
\end{tikzcd}$$

where:

$S = \{SC(p_i) | i = 0, \ldots, n\}$, p_i are propositional formulas.
$\bigcup_{i=1}^{3} S_i = S \cup SC(p)$, $S_i \cap S_j = \emptyset, i, j = 1,\ldots, 3$ & $i \neq j$
$a_4 = a_2, b_4 = b_2, c_4 = c_2$

Case 2 ($\{SC(p)\} = S$) In this case, the justification game has the following specification (Specification 5):

$$\begin{tikzcd}
& \mathit{Accept}(Ag_2, SC(p)) \\
\mathit{Justify}(Ag_1, [S], SC(p)) \arrow[ur, "a'_4"] \arrow[dr, "b'_4"'] & \\
& \mathit{Refuse}(Ag_2, SC(p))
\end{tikzcd}$$

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.

3.2.5 Attack Game

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

$$\begin{tikzcd}
 & \mathit{Refuse}(Ag_2, S_1) \wedge (p' \text{⌂} \mathit{Arg\_Sys}(Ag_2) \Rightarrow \mathit{Create}(Ag_2, SC(p'))) \\
\mathit{Attack}(Ag_1, [S], SC(p)) \arrow[ur, "a_5"] \arrow[r, "b_5"] \arrow[dr, "c_5"] \arrow[ddr, "d_5"'] & \mathit{Accept}(Ag_2, S_2) \\
 & \mathit{Challenge}(Ag_2, S_3) \\
 & \mathit{Attack}(Ag_2, [S'], S_4)
\end{tikzcd}$$

where:

$S = \{SC(p_i)|i = 0, \ldots, n\}$, p_i are propositional formulas.
$\bigcup_{i=1}^{4} S_i = S \cup SC(p), Card(S_1) = 1, S_i \cap S_j = \emptyset, \quad i, j = 1, \ldots, 4 \;\&\; i \neq j$

The conditions a_5, b_5, c_5 and d_5 are specified as follows:

$a_5 = \exists i, SC(p_i) \in \mathit{Create\_Support}(Ag_2, SC(\neg q))$
where $\mathrm{S}_1 = \{SC(q)\}$
$b_5 = \forall i, SC(p_i) \in S_2 \Rightarrow p_i \text{⌂} \mathit{Arg\_Sys}(Ag_2)$
$c_5 = \forall i, SC(p_i) \in \mathrm{S}_3 \Rightarrow (\neg(p_i \text{⌂} \mathit{Arg\_Sys}(Ag_2)) \wedge \neg(\neg p_i \text{⌂} \mathit{Arg\_Sys}(Ag_2)))$
$d_5 = \forall i, SC(p_i) \in \mathrm{S}_4 \Rightarrow \exists \mathrm{S'}_j \subseteq S'$,
$\mathit{Content}(S'j) = \mathit{Support}(Ag_2, \neg p_i) \wedge = \nexists k, SC(p_k) \in$ 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 a refusal move could be an exit condition. If Ag_2 has an argument supporting a new offer p', it makes this counter-offer. 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 counter-offer p' by creating a new commitment. In this situation, Ag_1 will play an entry game by accepting or refusing the counter-offer. 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 $\mathit{Create\_Support}(Ag_2, SC(\neg q))$ and Create_Support(Ag_2, $SC(\neg p_i)$).

3.2.6 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.

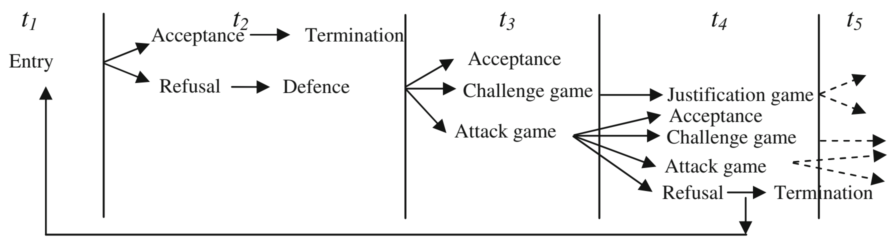


Fig. 3 The persuasive negotiation dynamics

3.3 Protocol Dynamics

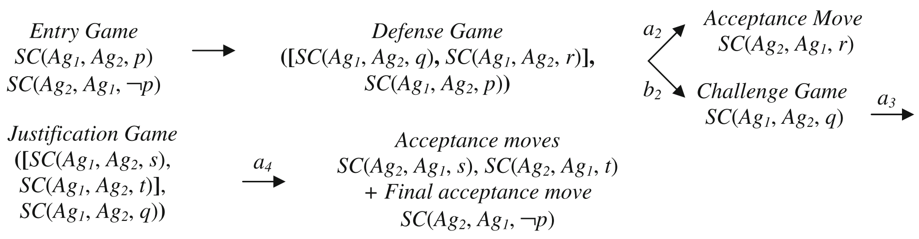
The persuasive negotiation dynamics is described by the chaining of a finite set of dialogue games: *entry game*, *acceptance move*, *refusal move*, *defence*, *challenge*, *attack* and *justification games*. These games can be combined in a sequential and parallel way (Fig. 3). In this figure, sold arrows have the same meaning as in the specification of dialogue games presented above, and the conditions are omitted for simplification reasons. The dotted arrows are used to indicate unspecified possible continuations.

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 refusal, the player can propose a counter-offer, after which a renegotiation will start. The persuasive negotiation dynamics continues until the exit conditions become satisfied (final acceptance or a refusal). We note that during a same conversation, an agent cannot play the same move (with the same content) more than once. After using an argument, an agent cannot use it again during this conversation (reiterations are prohibited). 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: $(\{q, r\}, p)$ and $(\{s, t\}, q)$. 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 Formal Analysis

4.1 Termination, Soundness, and Completeness

In this section we discuss the formal properties of our persuasive negotiation protocol from a computational point of view. These properties are: termination (there is no deadlock in the protocol), soundness (the protocol specification is correct), and completeness (the protocol is complete with respect to the agents' knowledge bases).

Theorem 1 (Termination) *For any set of dialogue games, the persuasive negotiation protocol always terminates.*

Proof The persuasive negotiation protocol is defined by the chaining of a finite set of dialogue games that can be played recurrently. Because the same move is prohibited during a conversation, and the content of communicative acts is finite in term of size, challenge and attack games are finite. In addition, because the agents' knowledge bases are finite and when an argument is justified by itself, the addressee could only accept or refuse (case 2 of justification game), then justification games are finite as well. Consequently, the protocol always converges toward executing either a final refusal or final acceptance. □

Theorem 2 (Soundness) *If the protocol terminates by a final acceptance (resp. final refusal), then an agreement is (rep. is not) achieved.*

Proof According to the dialogue game specifications, if one of the participating agents plays the final acceptance move, this means that it has an argument supporting the addressee's argument advanced by playing a defence, attack, or justification game. Consequently, this agent has an argument supporting the last offer made by the addressee. Having this argument in the knowledge base means that an agreement is achieved. In the opposite case, if an agent plays a final refusal, then all the exchanged offers can not be supported by one of the two agents. This means that there is no argument from the two agents' knowledge bases supporting one of the offers. Consequently an agreement is not achieved. □

The soundness property shows that the protocol is correct. However, what is important is to show that if an agreement is possible given the two agents' knowledge bases, then the protocol execution will achieve this agreement.

Theorem 3 (Completeness) *If an agreement can be achieved from the agents' knowledge bases, then the protocol execution will results in achieving this agreement.*

Proof Let us suppose that from the union of the two agents' knowledge bases, it is possible to build an argument supporting a given offer p which is not attacked by another argument from the union. Consequently, this argument is accepted by the two agents. Let us show how this argument can be achieved when executing the protocol. We will use a proof by construction.

If p is the initial offer made, for example, by Ag_1, then Ag_2 will accept it in the entry game. So the agreement is achieved. If the initial offer is p' (p and p' are different but related because it's about the same topic) which is refused by Ag_1, then Ag_2 will defend it by proposing an argument. Ag_1 will probably accept a part of this argument, challenge a second part, and attack a third part by possibly making a counter-offer. At this level, Ag_1 can not completely accept the Ag_2's argument because its knowledge base is consistent. If this counter-offer is p (which is possible since Ag_1's argumentation system supports p), we are done, because it will be accepted by Ag_2. If not, Ag_2 will justify the challenge part and plays an attack game by possibly making a new counter-offer or refuse the attack and make a new counter-offer. If the counter-offer is p, then we are done. If not, Ag_1 will play the same games. The process will continue until a counter-argument p is made by one of the two agents. There is a guarantee that p will be made, because if not, one of the two agents will play a final refusal since according to *Theorem 1* the protocol always terminates. This means that the final offer can not be supported by one of the two agents' knowledge bases and this agent can not make a counter-offer, which is contradictory with the initial hypothesis. □

4.2 Complexity Analysis

It is proved that using first order logic and fully propositional logic for argumentative reasoning is not appropriate for automated negotiation since first order logic is semi-decidable and propositional logic is intractable (exponential time complexity) (Bentahar et al. 2007; Parsons 1998). Here we prove that our protocol is efficient because the reasoning procedures are polynomial with respect to the size of agent's knowledge base ($(O(|\Gamma|)$). This is due to the fact that our logical language (Propositional Horn logic) is simpler and the dialogue games are simple logical rules. Because all these dialogue games are based on argumentation, and the decision parameters (the conditions associated to the rules) that agents use to combine these dialogue games are expressed in terms of the possibility of building arguments, the complexity of the protocol is determined by the complexity of building arguments. In the following we present the different complexity results.

Proposition 1 *Given a Horn knowledge base Γ, a subset $H \subseteq \Gamma$, and a formula h. Checking whether (H, h) is a non-necessarily minimal argument is polynomial with respect to the sizes of H and h, i.e. $O(|H| \times |h|)$.*

Proof From the linear time algorithms for Horn satisfiability in Dowling and Gallier (1984), it follows that the Horn implication problem $H \vdash h$ is decidable in $O(|H| \times |h|)$

time. From the same result, it also follows that deciding whether H is consistent is polynomial with respect to the size of H. □

Proposition 2 *Given a Horn knowledge base* Γ, *and an argument* (H, h) *over* Γ. *Checking whether* (H, h) *is minimal is polynomial* $(O(|H| \times |h|))$.

Proof Let l be a literal. The following algorithm resolves the problem: $\forall l \in H$ check if $H - \{l\} \vdash h$. Because the implication problem is polynomial, we are done. □

Proposition 3 *Let* Γ *be a consistent Horn knowledge base, h a formula, and A the set of arguments over* Γ

$$\exists H \subseteq \Gamma : (H, h) \in A \Rightarrow \forall H' : H \subseteq H' \subseteq \Gamma, (H', h) \in A$$

Proof If (H, h) is an argument where H is a set of Horn formulas under the form c or $p_1 \vee p_2 \vee \cdots \vee p_n \rightarrow c$ where $p_1, p_2, \ldots, p_n$ are positive literals, then adding any Horn formula to H will result in a consistent set of formulas $H' : \Gamma \supseteq H' \supseteq H$. Since $H \vdash h$, it follows that $H \vdash h$,whence the proposition. □

Theorem 4 *Given a consistent Horn knowledge base* Γ *and a formula h.Building an argument* (H, h) *from* Γ *is polynomial* $(O(|\Gamma|))$.

Proof From *Proposition 3*, it follows that there is an argument supporting h iff $(\Gamma, H) \in A$. By *Propositions 1* and *2*, the theorem follows. □

5 Implementation

In this section we describe a prototype implementation as proof of concept of the different dialogue games. The prototype is implemented using the $Jack^{TM}$ platform (The Agent Oriented Software Group 2005). 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.

Negotiating agents in the developed system are implemented as $Jack^{TM}$ agents, i.e. they inherit from the basic class $Jack^{TM}$ Agent. Their 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 propositional Horn 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.

The agents' knowledge bases (KBs) contain two types of information: arguments and beliefs. Arguments have the form ([*Support*], *Conclusion*), where *Support* is a set of propositional Horn 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.

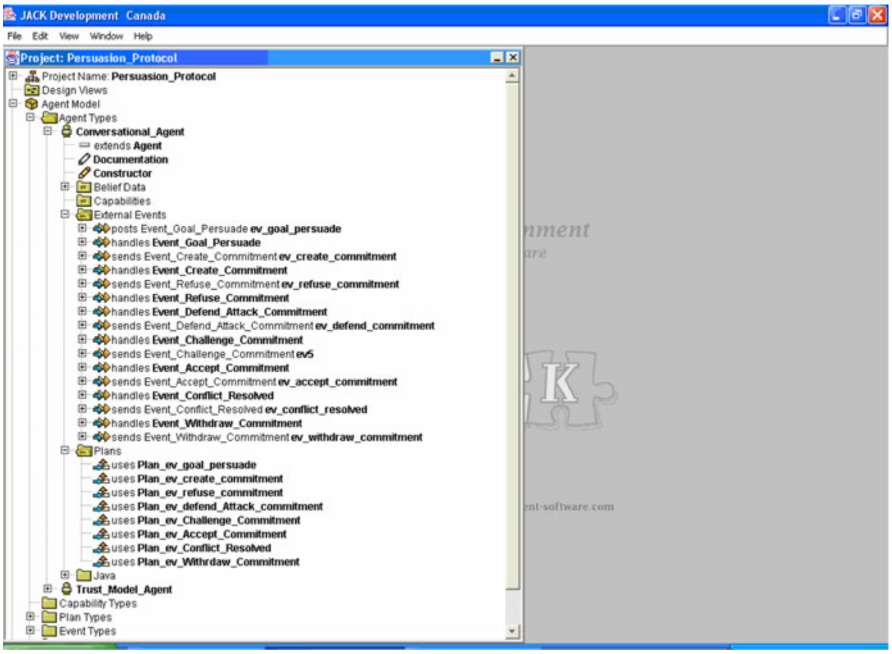
To open a dialogue game, an agent uses its argumentation system. The argumentation system allows this agent to seek in its knowledge base an argument for a given conclusion or for its negation ("counter-argument"). For example, before creating a commitment $SC(p)$, an agent must find an argument for 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 negotiating agent can instantiate these classes. An agent Ag_1 starts a dialogue game by generating an event and by sending it to the addressee 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. Figs. 4 and 5 illustrate snapshots of the system.

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.

Figure 5 describes a detailed example generated by the implemented prototype. In this example, symbolic propositions are used. The persuasive negotiation starts when Ag_1 makes an offer p to Ag_2 that it refuses because it has an argument against p, which makes the entry conditions satisfied (entry game). Ag_1 plays then the defence game by defending p using a, b, and c. As a reply to this defence, Ag_2 plays the attack



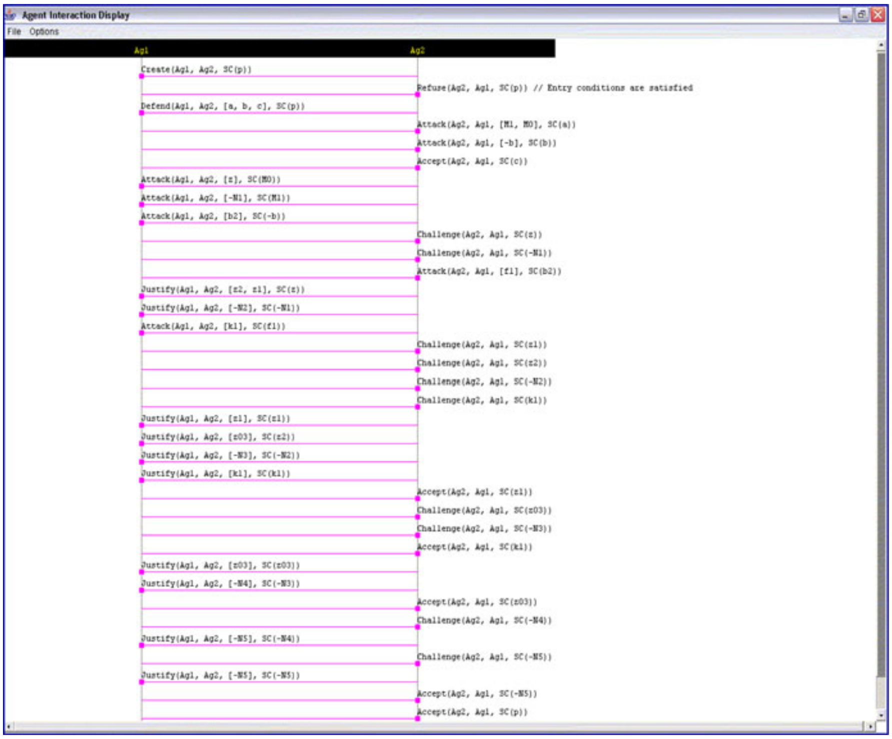
```
Project Name: Persuasion_Protocol
Design Views
Agent Model
  Agent Types
    Conversational_Agent
      extends Agent
      Documentation
      Constructor
      Belief Data
      Capabilities
      External Events
        posts Event_Goal_Persuade ev_goal_persuade
        handles Event_Goal_Persuade
        sends Event_Create_Commitment ev_create_commitment
        handles Event_Create_Commitment
        sends Event_Refuse_Commitment ev_refuse_commitment
        handles Event_Refuse_Commitment
        handles Event_Defend_Attack_Commitment
        sends Event_Defend_Attack_Commitment ev_defend_commitment
        handles Event_Challenge_Commitment
        sends Event_Challenge_Commitment ev5
        handles Event_Accept_Commitment
        sends Event_Accept_Commitment ev_accept_commitment
        handles Event_Conflict_Resolved
        sends Event_Conflict_Resolved ev_conflict_resolved
        handles Event_Withdraw_Commitment
        sends Event_Withdraw_Commitment ev_withdraw_commitment
      Plans
        uses Plan_ev_goal_persuade
        uses Plan_ev_create_commitment
        uses Plan_ev_refuse_commitment
        uses Plan_ev_defend_Attack_commitment
        uses Plan_ev_Challenge_Commitment
        uses Plan_ev_Accept_Commitment
        uses Plan_ev_Conflict_Resolved
        uses Plan_ev_Withrdaw_Commitment
      Java
    Trust_Model_Agent
  Capability Types
  Plan Types
  Event Types
```

Fig. 4 The system data structures

game by attacking a using $M1$ and $M0$ and attacking b using $\neg b$. Ag_2 plays also an acceptance move by accepting the third part c. Afterward, Ag_1 replies by playing the same game (attack game). This agent attacks then $M0$, $M1$, and $\neg b$ using respectively $z, \neg N1$ *and* $b2$. Subsequently, Ag_2 plays in parallel the challenge game by challenging z and $\neg N1$ and the attack game by attacking $b2$ using $f1$. As a reply to the challenge game, Ag_1 plays the justification game by justifying z and $\neg N1$ using respectively the arguments $(\{z2, z1\}, z)$ and $(\{\neg N2\}, \neg N1)$. In parallel, Ag_1 replies to the attack game by playing another attack game in which $f1$ is attacked using $k1$. The persuasive negotiation continues in many turns as depicted in Fig. 5 and terminates by a final acceptance move where Ag_2 accepts Ag_1's proposal.

6 Related Work and Discussion

A key component for designing an agent communication system is the dialogue game protocol. Such a protocol is specified as a set of rules that govern the well-behaviour of interacting agents in order to generate dialogues. It specifies the set of speech acts allowed in a dialogue and their allowed types of replies (Amgoud et al. 2006). Agents in the protocol are considered as playing games with personal goals and a set of moves (i.e. instantiated speech acts) that can be used to try to reach those goals. According to McBurney and Parsons (2002), a dialogue game specification consists of the following elements:



```
Create(Ag1, Ag2, SC(p))
                                        Refuse(Ag2, Ag1, SC(p)) // Entry conditions are satisfied
Defend(Ag1, Ag2, [a, b, c], SC(p))
                                        Attack(Ag2, Ag1, [M1, M0], SC(a))
                                        Attack(Ag2, Ag1, [-b], SC(b))
                                        Accept(Ag2, Ag1, SC(c))
Attack(Ag1, Ag2, [z], SC(M0))
Attack(Ag1, Ag2, [-N1], SC(M1))
Attack(Ag1, Ag2, [b2], SC(-b))
                                        Challenge(Ag2, Ag1, SC(z))
                                        Challenge(Ag2, Ag1, SC(-N1))
                                        Attack(Ag2, Ag1, [f1], SC(b2))
Justify(Ag1, Ag2, [z2, z1], SC(z))
Justify(Ag1, Ag2, [-N2], SC(-N1))
Attack(Ag1, Ag2, [k1], SC(f1))
                                        Challenge(Ag2, Ag1, SC(z1))
                                        Challenge(Ag2, Ag1, SC(z2))
                                        Challenge(Ag2, Ag1, SC(-N2))
                                        Challenge(Ag2, Ag1, SC(k1))
Justify(Ag1, Ag2, [z1], SC(z1))
Justify(Ag1, Ag2, [z03], SC(z2))
Justify(Ag1, Ag2, [-N3], SC(-N2))
Justify(Ag1, Ag2, [k1], SC(k1))
                                        Accept(Ag2, Ag1, SC(z1))
                                        Challenge(Ag2, Ag1, SC(z03))
                                        Challenge(Ag2, Ag1, SC(-N3))
                                        Accept(Ag2, Ag1, SC(k1))
Justify(Ag1, Ag2, [z03], SC(z03))
Justify(Ag1, Ag2, [-N4], SC(-N3))
                                        Accept(Ag2, Ag1, SC(z03))
                                        Challenge(Ag2, Ag1, SC(-N4))
Justify(Ag1, Ag2, [-N5], SC(-N4))
                                        Challenge(Ag2, Ag1, SC(-N5))
Justify(Ag1, Ag2, [-N5], SC(-N5))
                                        Accept(Ag2, Ag1, SC(-N5))
                                        Accept(Ag2, Ag1, SC(p))
```

Fig. 5 A snapshot from the prototype: a persuasive negotiation example

1. Commencement rules: Rules which define the circumstances under which a dialogue starts.
2. Locutions: rules which indicate what moves are permitted.
3. Combination rules: Rules which define the dialogical contexts under which particular locutions are permitted or not.
4. Commitments: Rules which define the circumstances under which agents express commitment to a proposition.
5. Termination rules: Rules that define the circumstances under which a dialogue ends.

McBurney et al. (2002) discuss a set of desiderata for argumentation-based dialogue game protocols. They propose thirteen criteria that permit assessing these protocols:

1. Stated dialogue purpose: A dialogue game protocol should have one or more publicly-stated purposes, and its locutions and rules should facilitate their achievements.
2. Diversity of individual purposes: A dialogue game protocol should permit participating agents to achieve their own individual purposes consistent with the overall purpose of the dialogue.

3. Inclusiveness: A dialogue game protocol should not exclude participation for any potential agent which is qualified and willing to participate.
4. Transparency: Participants to a dialogue game should know the rules and structure of the dialectical system prior to commencement of the dialogue.
5. Fairness: A dialogue game protocol should either treat all participants equally, or, if not, make explicit any asymmetries in their treatment.
6. Clarity of argumentation theory: A dialogue game protocol should conform, at least at the outset, to a stated theory of argument.
7. Separation of syntax and semantics: The syntax of a dialogue game protocol should be defined separately from its semantics. There are two reasons for such a separation: (1) this approach enables the same protocol syntax to be used with multiple semantics; (2) the problem of checking the compliance of agents with this semantics will be possible since the semantics is publicly and clearly declared.
8. Rule consistency: The locutions and rules of a dialogue game should together be internally consistent.
9. Encouragement of resolution: Resolution of each dialogue (normal termination) should be facilitated and not precluded by the locutions and rules of a dialectical system.
10. Discouragement of disruption: Normally, the rules of a dialogue game protocol should discourage or preclude disruptive behaviour, such as uttering the same locution repeatedly.
11. Enablement of self-transformation: A dialogue game protocol should permit participants to undergo self-transformation in the course of a dialogue; e.g., participants to a negotiation should be able to change their preferences or their valuations of utility as a result of information they receive from others in the dialogue.
12. System simplicity: The locutions and rules of a dialogue game protocol should be as simple as possible.
13. Computational simplicity: A dialogue game protocol should be designed to minimize any computational demands on its participants, and on the system itself.

Amgoud et al. (2006) identify seven parameters considered as essential for defining negotiation dialogue game protocols. These parameters are: (1) the underlying logical language used to express negotiation rules and allowed locutions; (2) the set of speech acts or locutions uttered in a negotiation; (3) the set of involved agents; (4) a function (*Reply*) associating to each speech act its expected replies; (5) a variable *Back* $\in \{0, 1\}$ such that *Back* = 1 (resp. 0) means that the protocol allows (resp. does not allow) for backtracking; (6) a function (*Turn*) governing the turn-taking of the agents; (7) a function (*N_Move*) determining at each turn and for each agent the number of moves that is allowed to perform at that turn.

Parsons (1998) propose a formal negotiation framework based upon an argumentation system. The proposed negotiation protocol includes five main locutions: *proposal*, *critique*, *counter-proposal*, *accept*, and *withdraw*. The protocol includes also a special locution called *meta-information* to express explanations or preferences. For example, proposals, counter-proposals, and critiques can be supplied with explanations which are a form of justifications that agents supply to support their locutions. The process starts when an agent generates a proposal. Other agents then either accept it, critique it by providing a comment on which parts of the proposal they like and which parts they dislike, make counter-proposals, or provide meta-information. The first agent may make a second proposal, accept the counter-proposal, critique it, or withdraw from the process. If a critique is made, either agent can keep the process moving forward by making another proposal or withdraw from the process. The process iterates until one of the agents accept or withdraw. The authors use beliefs–desires–intentions architecture for agents (BDI) and a BDI propositional logic as the underlying language. Because BDI architecture and logic are based on agents' private states, checking the compliance of agents with the negotiation protocol is not possible. Also using a BDI propositional logic makes the underlying reasoning mechanisms intractable.

Sadri et al. (2001) introduce a logic-based approach for one-to-one agent negotiation based on dialogue games. In this approach, agents agree upon a language for negotiation, while possibly adopting different negotiation policies, each corresponding to an agent program type (deterministic or exhaustive). A single dialogue game is used to obtain a resource (resource reallocation problem). After the termination of one game, new games can be triggered in a sequence in order to allow the agent to collect all the resources needed to achieve its goal. The negotiation protocol allows the following moves: *Request*, *Accept*, *Refuse*, *Challenge*, *Justify*, and *Promise*. The authors prove some interesting properties of the protocol: soundness (the successful execution of the protocol results in a successful resource reallocation) and completeness (if there exists a solution to the resource reallocation problem, then the protocol could be successfully executed to achieve this solution). However, the protocol is specified for one specific case which is the negotiation of resources and its complexity is not specified.

Atkinson et al. (2005) define a formal and computational dialogue game protocol for multi-agent argument over proposals for action. Called Persuasive Argument for Multiple Agents (PARMA), the protocol uses an argumentation theory that agents use in order to have a rational behaviour. The idea is that one agent endorses a particular action, and seeks to persuade another agent do the same. The protocol permits actions to be proposed, attacked, and defended by agents engaged in a persuasion interaction. In this protocol a commitment store is associated with each participant, which stores publicly the commitments made by the participants. The protocol introduces the post-conditions and pre-conditions utterances that indicate post and prior commitments. Commitments in this protocol are statements which an agent must defend if attacked, and may not be a true expression of the agent's beliefs or intension. Although the protocol is equipped with an interesting denotational semantics and an implementation, its complexity is not specified. In addition, the protocol is defined for persuasion settings and ca not be used for automated negotiation.

Let us now discuss how our persuasive negotiation protocol meets the requirements discussed above. On one hand, it is easy to verify that this protocol specifies the five factors discussed by Amgoud et al. (2006) since the underlying logical language is explicitly propositional Horn clauses, and the dialogue game specifications clearly implements *Reply*, *Backtracking*, *Turn* and *N_Move* functions. On the other hand, McBurney and Parsons's factors (McBurney and Parsons 2002) are considered as follows:

1. Commencement rules: The entry game implements these rules.
2. Locutions: The chaining dialogue games specify these locutions and the associated rules.
3. Combination rules: The protocol dynamics specifies these rules.
4. Commitments: The conditions associated to the dialogue games reflect the agents strategies and the commitment rules.
5. Termination rules: The termination condition and the termination proof illustrate these rules.

Our persuasive negotiation protocol is fully specified according to McBurney et al.s' desiderata (McBurney et al. 2002). This is not the case in a good number of other dialogue game protocols particularly from the computational complexity point of view. The following illustrates how this protocol meets these desiderata:

1. Stated dialogue purpose: Our protocol is explicitly used for persuasive negotiation.
2. Diversity of individual purposes: The protocol permits agents to achieve their own purposes in terms of negotiating with peers.
3. Inclusiveness: There is no elimination of agents in the protocol.
4. Transparency: The protocol's rules and structure are shared by all agents.
5. Fairness: Agents in the protocol are treated equally and they are governed by the same rules.
6. Clarity of argumentation theory: The argumentation theory is explicit and clear in the protocol's specification.
7. Separation of syntax and semantics: The protocol's specification language separates syntax from semantics. The syntax is expressed in terms of actions and conditions, and the semantics is defined in term of argumentation theory.
8. Rule consistency: All the rules are specified and the consistency is guaranteed by the conditions, which are expressed in terms of arguments. Furthermore, there are no infinite cycles and repeated locutions. The protocol termination, soundness and completeness are also formally proved.
9. Encouragement of resolution: The normal termination is not eliminated by the rules.
10. Discouragement of disruption: The persuasive negotiation protocol precludes disruptive behaviour by prohibiting the performance of the same communicative acts repeatedly and the use of the same arguments.
11. Enablement of self-transformation: Self-transformation is enabled in the protocol since acceptance moves are allowed, so that agents can accept addressees' arguments and thus, update their knowledge bases that contain their beliefs.
12. System simplicity: a small number of locutions and dialogue games are suggested. The combination rules describing the protocol dynamics are also simple.
13. Computational simplicity: It is proved that the reasoning is tractable and computationally efficient (polynomial complexity).

7 Conclusion and Future Work

The contribution of this paper is the specification and implementation of autonomous and efficient negotiation protocol between software agents. The proposed approach is based upon persuasive argumentation. In this approach, the agent's reasoning capabilities are linked to their ability to argue. The logical language used to specify the protocol 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 protocol and its dynamics by the combination of five dialogue games and we presented the implementation of such a protocol using a logical programming paradigm. We proved the formal and computational properties of this protocol (termination, soundness, and completeness) and we discussed its computational complexity.

For future work, we plan to investigate relevance-based reasoning and consider equipping agents with negotiation strategies using preferences and game theoretical analysis (Rahwan and Larson 2008). Another interesting direction for future work is verifying the proposed protocol using model checking techniques. The method we are investigating is an automata theoretic approach based on a tableau method (Bentahar et al. 2006). 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 promising. 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 authors would like to thank the reviewers for their very helpful comments and suggestions. The first author acknowledges the financial support provided by NSERC (Canada), FQRNT, and FQRSC (Quebec).

References

Amgoud L, Maudet N, Parsons S (2000) Modelling dialogues using argumentation. In: Proceedings of 4th international conference on multi agent systems, pp 31–38

Amgoud L, Belabbes S, Prade H (2006) A formal general setting for dialogue protocols. In: Proceedings of artificial intelligence: methodology, systems, and applications, pp 13–23

Atkinson K, Bench-Capon T, McBurney P (2005) A dialogue game protocol for multi-agent argument over proposals for action. J AAMAS (Special issue on Argumentation in Multi-Agent Systems) 11(2): 153–171

Bench-Capon T, Atkinson K, McBurney P (2009) Altruism and agents: an argumentation based approach to designing agent decision mechanisms. In: Proceedings of the international joint conference on autonomous agents and multiagent systems (in press)

Bentahar J, Moulin B, Meyer J-J Ch, 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

Bentahar J, Moulin B, Chaib-draa B (2005) Specifying and implementing a persuasion dialogue game using commitment and argument network. In: Argumentation in multi-agent systems, vol 3366(1). Springer, pp 130–148

Bentahar J, Moulin B, Meyer J-J Ch (2006) A tableau method for verifying dialogue game protocols for agent communication. In: Declarative agent languages and technologies, vol 3904. Springer, pp 223–244

Bentahar J, Maamar Z, Benslimane D, Thiran P (2007) An argumentation framework for communities of Web services. IEEE Intell Syst 22(6):75–83

Brewka G (2001) Dynamic argument systems: a formal model of argumentation processes based on situation calculus. J Logic Comput 11(2):257–282

Castelfranchi C (1995) Commitments: from individual intentions to groups and organizations. In: Proceedings of international conference on multi agent systems, pp 41–48

Dastani M, Hulstijn J, der Torre LV (2000) Negotiation protocols and dialogue games. In: Proceedings of Belgium/Dutch AI conference, pp 13–20

Dignum F (ed) (2003) Advances in agent communication. In: International workshop on agent communication languages. LNAI 2922, Springer

Dowling W, Gallier JH (1984) Linear-time algorithms for testing the satisfiability of propositional horn theories. J Logic Program 1(3):267–284

Elvang-Goransson M, Fox J, Krause P (1993) Dialectic reasoning with inconsistent information. In: Proceedings of 9th conference on uncertainty in artificial intelligence, pp 114–121

Endriss U, Maudet N, Sadri F, Toni F (2003) Logic-based agent communication protocols. In: Dignum F (ed) Advances in agent communication. In: International workshop on agent communication languages. LNAI 2922, Springer, pp 91–107

Fornara N, Colombetti M (2003) Protocol specification using a commitment based ACL. In: Dastani M, Hulstijn J, der Torre LV (2000) Negotiation protocols and dialogue games. In: Proceedings of Belgium/Dutch AI conference, pp 108–127

Karunatillake NC, Jennings NR, Rahwan I, Norman TJ (2005) Argument-based negotiation in a social context. In: Proceedings of the international joint conference on autonomous agents and multi-agent systems, pp 1331–1332

Karunatillake NC, Jennings NR, Rahwan I, McBurney P (2009) Dialogue games that agents play within a society. Artif Intell (in press)

Kraus S, Sycara KP, Evenchik A (1998) Reaching agreements through argumentation: a logical model and implementation. Artif Intell 104(1–2):1–69

Li C, Giampapa JA, Sycara KP (2006) Bilateral negotiation decisions with uncertain dynamic outside options. IEEE Trans Syst Man Cybern Part C 36(1):31–44

McBurney P, Parsons S (2002) Games the agents play: A formal framework for dialogues between autonomous agents. J Logic, Lang Inf 11(3):1–22

McBurney P, Parsons S, Wooldridge M (2002) Desiderata for agent argumentation protocols. In: International conference on autonomous agents and multi-agent systems, ACM Press, pp 402–409

Moulin B, Chaib-draa B (1996) Distributed artificial intelligence: an overview. In: Jennings N, O'Hare G (eds) Foundations of distributed artificial intelligence. Wiley, pp 3–55

Parsons S, Sierra C, Jennings N (1998) Agents tat reason and negotiate by arguing. J Logic Comput 8(3):261–292

Prakken H (2001) Relating protocols for dynamic dispute with logics for defeasible argumentation. Synthese 127:187–219

Rahwan I, Larson K (2008) Mechanism design for abstract argumentation. In: Proceedings of the international joint conference on autonomous agents and multiagent systems, pp 1031–1038

Rahwan I, Sonenberg L, Jennings NR, McBurney P (2007) STRATUM: a methodology for designing heuristic agent negotiation strategies. Appl Artif Intell 21(10)

Sadri F, Toni F, Torroni P (2001) Dialogues for negotiation: agent varieties and dialogue sequences. Intelligent agent series VIII, vol 2333 of LNAI. Springer, pp 405–421

Shakshuki E, Trudel A, Xu Y (2007) A multi-agent temporal constraint satisfaction system based on Allen's interval algebra and probabilities. Int J Inf Technol Web Eng 2(2):45–64
Sycara K (1998) Multiagent systems. AI Mag Am Assoc Artif Intell 19(2):79–92
Sycara K, Pannu A, Williamson M, Zeng D, Decker K (1996) Distributed intelligent agents. IEEE Expert 11(6):36–46
The Agent Oriented Software Group. Jack 4.1. 2005. http://www.agent-software.com/
Walton DN, Krabbe ECW (1995) Commitment in dialogue: basic concepts of interpersonal reasoning. State University of New York Press, NY
Wooldridge M (2003) Reasoning about rational agents. The MIT Press, Cambridge, MA