A Computational Model for Conversation Policies for Agent Communication

Jamal Bentahar[1], Bernard Moulin[1], John-Jules Ch. Meyer[2], and Brahim Chaib-draa[1]

[1] Laval University, Department of Computer Science and Software Engineering, Canada
jamal.bentahar.1@ulaval.ca
{bernard.moulin, brahim.chaib-draa}@ift.ulaval.ca
[2] University Utrecht, Department of Computer Science, The Netherlands
jj@cs.uu.nl

Abstract. In this paper we propose a formal specification of a persuasion protocol between autonomous agents using an approach based on social commitments and arguments. In order to be flexible, this protocol is defined as a combination of a set of conversation policies. These policies are formalized as a set of dialogue games. The protocol is specified using two types of dialogue games: entry dialogue game and chaining dialogue games. The protocol terminates when exit conditions are satisfied. Using a tableau method, we prove that this protocol always terminates. The paper addresses also the implementation issues of our protocol using logical programming and an agent-oriented platform.

1 Introduction

Research in agent communication has received much attention during the past years [9; 13; 14]. Agent communication protocols specify the rules of interaction governing a dialogue between autonomous agents in a multi-agent system. These protocols are patterns of behavior 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 expressive formalisms. Indeed, logic-based protocols seem an interesting way for specifying these protocols [3; 16].

On the one hand, conversation policies [18] and dialogue games [12; 21] aim at offering more flexible protocols [20]. This is achieved by combining different policies and games to construct complete and more complex protocols. In this paper we argue that conversation policies and dialogue games are related and can be used together to specify agent communication. Conversation policies are declarative specifications that govern communication between autonomous agents. We propose to formalize these policies as a set of dialogue games. Dialogue games are interactions between players, in which each player moves by performing utterances according to a pre-defined set of roles. Indeed, protocols specified using, for example, finite state machines are not flexible in the sense that agents must respect the whole protocol from the beginning to

J. Leite and P. Torroni (Eds.): CLIMA V, LNAI 3487, pp. 178 – 195, 2005.

the end. Thus, we propose to specify these protocols by small conversation policies that can be logically put together using a combination of dialogue games.

On the other hand, in the last years, some research works addressed the importance of social commitments in the domain of agent communication [4; 5; 11; 20; 24; 27]. These works showed that 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 behaviors. This framework has the advantage of being expressive because all speech act types can be represented by commitments [11]. Commitment-based protocols enable the content of agent interactions to be represented and reasoned about [17; 28]. In opposition to the BDI mental approach, the commitment-approach stresses the importance of conventions and the public aspects of dialogue. A speaker is committed to a statement when he makes this statement or when he agreed upon this statement made by another participant. In fact, we do not speak here about the expression of a belief, but rather about a particular relationship between a participant and a statement. What is important in this approach is not that an agent agrees or disagrees upon a statement, but rather the fact that the agent *publicly expresses* agreement or disagreement, and acts accordingly.

In this paper we present a persuasion dialogue which is specified using conversation policies, dialogue games and a framework based on commitments. In addition, in order to allow agents to effectively reason on their communicative actions, our framework is also based on an argumentative approach. In our framework the agent's reasoning capabilities are linked to their ability to argue. In this paper *we consider conversation policies as units specified by dialogue games whose moves are expressed in terms of actions that agents apply to commitments and arguments*. Indeed, the paper presents three results: 1- A new formal language for specifying a persuasion dialogue as a combination of conversation policies. 2- A termination proof of the dialogue based on a tableau method [10]. 3- An implementation of the specification using an agent oriented and logical programming.

The paper is organized as follows. In Section 2, we introduce the main ideas of our approach based on commitments and arguments. In Section 3 we address the specification of our persuasion protocol based on this approach. We present the protocol form, the specification of each dialogue game and the protocol dynamics. We also present our termination proof. In Section 4 we describe the implementation of a prototype allowing us to illustrate how the specification of dialogue games is implemented. In Section 5 we compare our protocol to related work. Finally, in Section 6 we draw some conclusions and we identify some directions for future work.

2 Commitment and Argument Approach

2.1 Social Commitments

A social commitment *SC* is a public commitment made by an agent (the *debtor*), that some fact is true or that something will be done. This commitment is directed toward a set of agents (*creditors*) [8]. A commitment is an obligation in the sense that the debtor must respect and behave in accordance with this commitment. A representation

of this notion as directed obligations using a deontic logic is proposed in [19]. Commitments are social in the sense that they are expressed publicly. Consequently, they are different from the private mental states like beliefs, desires and intentions. 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 [4]. 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 dialogue games that we specify in Section 3, the actions that an agent can perform on a commitment are: $Act \in \{Create, Withdraw\}$. The actions that an agent can perform on a 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 [4]. Formally, a speech act can be defined as follows:

Definition 1. $SA(Ag_1, Ag_2, U) =_{def} Act(Ag_1, SC(Ag_1, Ag_2, p)) \mid Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$

where $i, j \in \{1, 2\}$ and ($k = i$ 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.

2.2 Argumentation and Social Commitments

An argumentation system essentially includes a logical language *L*, a definition of the argument concept, and a definition of the attack relation between arguments. Several definitions were also proposed to define arguments. In our model, we adopt the following definitions from [15]. Here Γ indicates a knowledge base with deductive closure. $\vdash$ Stands for classical inference and $\equiv$ for logical equivalence.

Definition 2. *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 h.*

Definition 3. *Let* (H_1, h_1), (H_2, h_2) *be two arguments.* (H_1, h_1) *attacks* (H_2, h_2) *iff* $h_1 \equiv \neg h_2$.

In fact, before committing to some fact *h* being true (i.e. before creating a commitment whose content is *h*), the speaker agent must 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: $S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, p))$ to indicate the set of commitments S *created* by agent Ag_1 to support the content of $SC(Ag_1, Ag_2, p)$. This support relation is transitive i.e.:

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

Other details about our commitment and argument approach are described in [4]. Surely, an argumentation system is essential to help agents to act on commitments and on their contents. However, reasoning on other social attitudes should be taken into account in order to explain the agents' decisions. In our persuasion protocol we use the agents' trustworthiness to decide, in some cases, about the acceptance of arguments [6].

3 Conversation Policies for Persuasion Dialogue

3.1 Protocol Form

Our persuasion protocol is specified as a set of conversation policies. In order to be flexible, these policies are defined as initiative/reactive dialogue games. In accordance with our approach, the game moves are considered as actions that agents apply to commitments and to their contents. A dialogue game is specified as follows:

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

This specification indicates that if an agent Ag_1 performs the action $Action\_Ag_1$, and that the condition *Cond* is satisfied, then the interlocutor Ag_2 will perform the action $Action\_Ag_2$. The condition *Cond* is expressed in terms of the possibility of generating an argument from the agent's argumentation system and in terms of the interlocutor's trustworthiness. We use the notation: $p \triangle Arg\_Sys(Ag_1)$ 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 $\neg(p \triangle Arg\_Sys(Ag_1))$ 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 that supports p. To simplify the formalism, we use the notation $Act'(Ag_x, SC(Ag_i, Ag_j, p))$ to indicate the action that agent Ag_x performs on the commitment $SC(Ag_i, Ag_j, p)$ or on its content ($Act' \in \{Create, Withdraw, Accept, Challenge, Refuse\}$). For the actions related to the argumentation relations, we write $Act\text{-}Arg(Ag_x, [SC(Ag_n, Ag_m, q)], SC(Ag_i, Ag_j, p))$. This notation indicates that Ag_x defends (resp. attacks or justifies) the content of $SC(Ag_i, Ag_j, p)$ by the content of $SC(Ag_n, Ag_m, q)$ ($Act\text{-}Arg \in \{Defend, Attack, Justify\}$). The commitment that is written between square brackets [ ] is the support of the argument. In a general way, we use the notation $Act'(Ag_x, S)$ to indicate the action that Ag_x performs on the set of commitments S or on the contents of these commitments, and the notation $Act\text{-}Arg(Ag_x, [S], SC(Ag_i, Ag_j, p))$

to indicate the argumentation-related action that Ag_x performs on the content of $SC(Ag_i, Ag_j, p)$ using the contents of S as support. We also introduce the notation $Act\text{-}Arg(Ag_x, [S], S')$ to indicate that Ag_x 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 persuasion dialogue. The chaining games make it possible to construct the conversation. The protocol terminates when the exit conditions are satisfied (Figure 1).

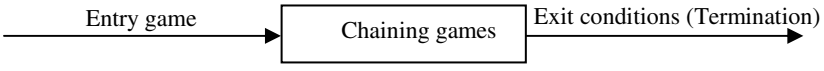
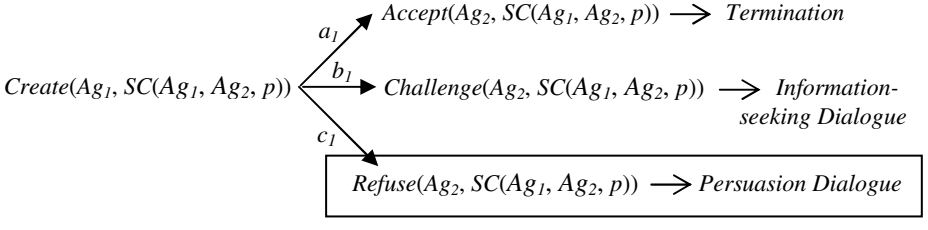


Fig. 1. The general form of the protocol

3.2 Dialogue Games Specification

A Entry Game

The conversational policy that describes the entry conditions in our persuasion protocol about a propositional formula p is described by the entry dialogue game as follows (*Specification 1*):



where a_1, b_1 and c_1 are three conditions specified as follows:

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

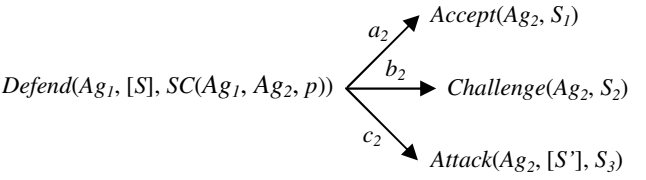
$b_1 = \neg(p \triangle Arg\_Sys(Ag_2)) \wedge \neg(\neg p \triangle Arg\_Sys(Ag_1))$

$c_1 = \neg p \triangle Arg\_Sys(Ag_2)$

If Ag_2 has an argument for p then it accepts p (the content of $SC(Ag_1, Ag_2, p)$) and the conversation terminates as soon as it begins (Condition a_1). If Ag_2 has neither an argument for p nor for $\neg p$, then it challenges p and the two agents open an information-seeking dialogue (condition b_1). The persuasion dialogue starts when Ag_2 refuses p because it has an argument against p (condition c_1).

B Defense Game

Once the two agents opened a persuasion dialogue, the initiator must defend its point of view. Thus, it must play a defense game. Our protocol is specified in such a way that the *persuasion dynamics* starts by playing a defense game. We have (*Specification 2*):



where:
$S=\{SC(Ag_1,Ag_2,p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\hbar_{i=1}^{3} S_i = S\ ,\ S_i \hbar\ S_j = \varnothing,\ i, j = 1,...,3 \,\&\, i \neq j$

By definition, *Defend*(Ag_1, [S], SC(Ag_1, Ag_2, p)) means that Ag_1 creates S in order to defend the content of SC(Ag_1, Ag_2, p). Formally:

Defend(Ag_1, [S], SC(Ag_1, Ag_2, p)) $=_{def}$ (*Create*(Ag_1, S)
$\wedge\ S$ = *Create_Support*(Ag_1, SC(Ag_1, Ag_2, p)))

We consider this definition as an *assertional* description of the *Defend* action.

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) $=_{def}$ ($\forall i$, SC(Ag_1, Ag_2, p_i) $\in S_1$ *Accept*(Ag_2, SC(Ag_1, Ag_2, p_i)))

Challenge(Ag_2, S_2) $=_{def}$ ($\forall i$, SC(Ag_1, Ag_2, p_i) $\in S_2$ *Challenge*(Ag_2, SC(Ag_1, Ag_2, p_i)))

Attack(Ag_2, [S'], S_3) $=_{def}$ $\forall i$, SC(Ag_1, Ag_2, p_i) $\in S_3$ $\exists S'_j \subseteq S'$:
Attack(Ag_2, [S'_j], SC(Ag_1, Ag_2, p_i))

where: $\hbar_{j=0}^{m}\ S'_j = S'$. 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:

$a_2 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_1$ $\quad p_i \sqcup Arg\_Sys(Ag_2)$

$b_2 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_2$ $\quad (\neg(p_i \sqcup Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \sqcup Arg\_Sys(Ag_2)))$

$c_2 = \forall i, SC(Ag_1, Ag_2, p_i) \in S_3$ $\quad \exists S'_j \subseteq S'$, $Content(S'_j) = Support(Ag_2, \neg p_i)$

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*):

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

where the condition a_3 is specified as follows:

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

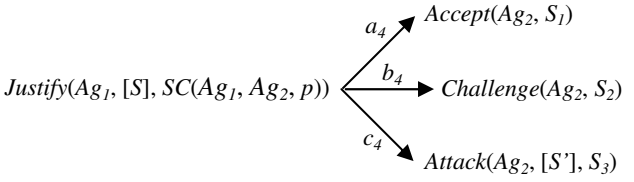
In this game, the condition a_3 is always true. The reason is that in accordance with the commitment semantics, an agent must always be able to defend the commitment it created [5].

D Justification Game

For this game we distinguish two cases:

Case1. $SC(Ag_1, Ag_2, p) \notin S$

In this case, Ag_1 justifies the content of its commitment $SC(Ag_1, Ag_2, p)$ 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:

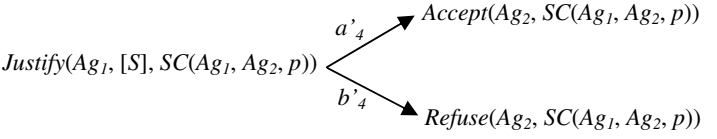
$S=\{SC(Ag_1,Ag_2,p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\hbar_{i=1}^{3} S_i = S$, $S_i \hbar S_j = \varnothing$, $i, j = 1,...,3 \,\&\, i \neq j$

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

Case2. $\{SC(Ag_1, Ag_2, p)\} = S$

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



Ag_1 justifies the content of its commitment $SC(Ag_1, Ag_2, p)$ 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 $SC(Ag_1, Ag_2, p)$ 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.

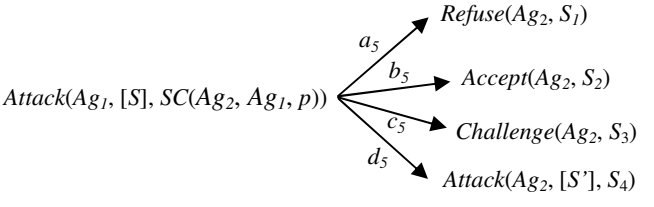
Like the definition of the *Defend* action, we define the *Justify* action as follows:

$$Justify(Ag_1, [S], SC(Ag_1, Ag_2, p)) =_{def} (Create(Ag_1, S) \wedge S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, p)))$$

This means that Ag_1 creates the set S of commitments to support the commitment $SC(Ag_1, Ag_2, p)$.

E Attack Game

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



where:
$S=\{SC(Ag_1,Ag_2,p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\hbar_{i=1}^{4} S_i = S$, $Card(S_1)=1$, $S_i \hbar S_j = \varnothing$, $i, j = 1,...,4 \,\&\, i \neq j$

Formally, the *Attack* action is defined as follows:

$Attack(Ag_1, [S], SC(Ag_2, Ag_1, p)) =_{def} (Create(Ag_1, SC(Ag_1, Ag_2, \neg p)) \wedge Create(Ag_1, S)$
$\wedge S = Create\_Support(Ag_1, SC(Ag_1, Ag_2, \neg p)))$

This means that by attacking $SC(Ag_2, Ag_1, p)$, Ag_1 creates the commitment $SC(Ag_1, Ag_2, \neg p)$ and the set S to support this commitment.

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

$a_5 = \exists i$: $SC(Ag_2, Ag_1, p_i) \in Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg q))$
where $S_1 = \{SC(Ag_1, Ag_2, q)\}$
$b_5 = \forall i$, $SC(Ag_1, Ag_2, p_i) \in S_2$ $\quad p_i \sqsubset Arg\_Sys(Ag_2)$
$c_5 = \forall i$, $SC(Ag_1, Ag_2, p_i) \in S_3$ $\quad (\neg(p_i \sqsubset Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \sqsubset Arg\_Sys(Ag_2)))$
$d_5 = \forall i$, $SC(Ag_1, Ag_2, p_i) \in S_4$ $\quad \exists S'_j \subseteq S'$: $Content(S'_j) = Support(Ag_2, \neg p_i)$
$\wedge \nexists k$: $SC(Ag_2, Ag_1, p_k) \in Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg p_i))$

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 defense game. 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 $Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg q))$ and $Create\_Support(Ag_2, SC(Ag_2, Ag_1, \neg p_i))$.

F Termination

The protocol terminates either by a final acceptance or by a refusal. There is a final acceptance when Ag_2 accepts the content of the initial commitment $SC(Ag_1, Ag_2, p)$ or when Ag_1 accepts the content of $SC(Ag_2, Ag_1, \neg p)$. Ag_2 accepts the content of $SC(Ag_1, Ag_2, p)$ iff it accepts all the supports of $SC(Ag_2, Ag_1, p)$. Formally:

$Accept(Ag_2, SC(Ag_1, Ag_2, p)) \Leftrightarrow$
$[\forall i, SC(Ag_1, Ag_2, p_i) \in Create\_Support(Ag_1, SC(Ag_1, Ag_2, p))$
$Accept(Ag_2, SC(Ag_1, Ag_2, p_i))]$

The acceptance of the supports of $SC(Ag_1, Ag_2, p)$ by Ag_2 does not mean that they are accepted directly after their creation by Ag_1, but it can be accepted after a number of challenge, justification and attack games. When Ag_2 accepts definitively, then it withdraws all commitments whose content was attacked by Ag_1. Formally:

$Accept(Ag_2, SC(Ag_1, Ag_2, p))$ $\quad [\forall i, \forall S, Attack(Ag_1, [S], SC(Ag_2, Ag_1, p_i))$
$Withdraw(Ag_2, SC(Ag_2, Ag_1, p_i))]$

On the other hand, Ag_2 refuses the content of $SC(Ag_1, Ag_2, p)$ iff it refuses one of the supports of $SC(Ag_1, Ag_2, p)$. Formally:

$$Refuse(Ag_2, SC(Ag_1, Ag_2, p)) \Leftrightarrow [\exists i: SC(Ag_1, Ag_2, p_i) \in Create\_Support(Ag_1, SC(Ag_1, Ag_2, p)) \wedge Refuse(Ag_2, SC(Ag_1, Ag_2, p_i))]$$

3.3 Protocol Dynamics

The persuasion dynamics is described by the chaining of a finite set of dialogue games: acceptance move, refusal move, defense, challenge, attack and justification games. These games can be combined in a sequential and parallel way (Figure 2).

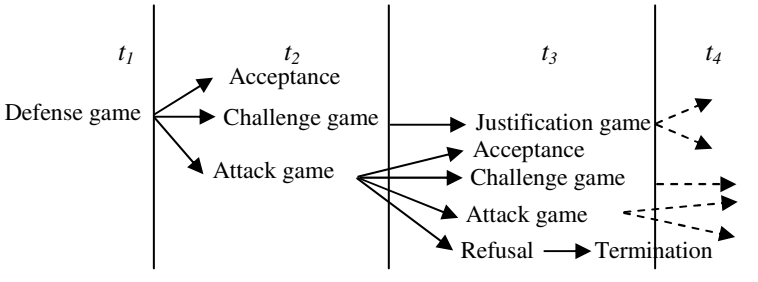


Fig. 2. The persuasion dialogue dynamics

After Ag_1's defense game at moment t_1, Ag_2 can, at moment t_2, 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_3, 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 final refusal move. The persuasion 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.

Indeed, our persuasion protocol can be described by the following BNF grammar:

Persuasion protocol : *Defense game* ~ *Dialogue games*
Dialogue games : (*Acceptance move*
// (*Challenge game* ~ *Justification game* ~ *Dialogue games*)
// (*Attack game* ~ *Dialogue games*))
| *refusal move*

where: "~" is the sequencing symbol, "//" is the possible parallelization symbol. Two games *Game1* and *Game 2* are possibly parallel (i.e. *Game1* // *Game2*) iff an agent can play the two games in parallel or only one game (*Game1* or *Game2*).

3.4 Termination Proof

Theorem. The protocol dynamics always terminates.

Proof. To prove this theorem, we use a tableau method [10]. The idea is to formalize our specifications as tableau rules and then to prove the finiteness of the tableau. Tableau rules are written in such a way that premises appear above conclusions. Using a tableau method means that the specifications are conducted in a top-down fashion. For example, specification 2 (p 3.2) can be expressed by the following rules:

$$R1: \frac{Defend(Ag_1,[S],SC(p))}{Accept(Ag_2,S_1)} \quad R2: \frac{Defend(Ag_1,[S],SC(p))}{Challenge(Ag_2,S_1)}$$

$$R3: \frac{Defend(Ag_1,[S],SC(p))}{Attack(Ag_2,[S'],S_1)}$$

We denote the formulas of our specifications by σ, and we define E the set of σ. We define an ordering $\hbar$ on E and we prove that $\hbar$ has no infinite ascending chains. Intuitively, this relation is to hold between σ_1 and σ_2 if it is possible that σ_1 is an ancestor of σ_2 in some tableau. Before defining this ordering, we introduce some notations: $Act^*(Ag, [S], S')$ with $Act^* \in \{Act', Act\text{-}Arg\}$ is a formula. We notice that formulas in which there is no support $[S]$, can be written as follows: $Act^*(Ag, [\phi], S')$. $\sigma[S] \rightarrow_R \sigma[S']$ indicates that the tableau rule R has the formula $\sigma[S]$ as premise and the formula $\sigma[S']$ as conclusion, with $\sigma[S] = Act^*(Ag, [S], S')$. The size $|S|$ is the number of commitments in S.

Definition 4. *Let $\sigma[S_i]$ be a formula and E the set of $\sigma[S_i]$. The ordering $\hbar$ on E is defined as follows. We have $\sigma[S_0]\ \hbar\ \sigma[S_1]$ if:*
$|S_1| < |S_0|$ *or*
For all rules Ri such that $\sigma[S_0] \rightarrow_{R0} \sigma[S_1] \rightarrow_{R1} \sigma[S_2] \ldots \rightarrow_{Rn} \sigma[S_n]$ we have $|S_n| = 0$.

Intuitively, in order to prove that a tableau system is finite, we need to prove the following:
1- if $\sigma[S_0] \rightarrow_R \sigma[S_1]$ then $\sigma[S_0]\ \hbar\ \sigma[S_1]$.
2- $\hbar$ has no infinite ascending chains (i.e. the inverse of $\hbar$ is well-founded).

Property 1 reflects the fact that applying tableau rules results in shorter formulas, and property 2 means that this process has a limit. The proof of 1 proceeds by a case analysis on R. Most cases are straightforward. We consider here the case of $R3$. For this rule we have two cases. If $|S_1| < |S_0|$, then $\sigma[S_0]\ \hbar\ \sigma[S_1]$. If $|S_1| \geq |S_0|$, the rules corresponding to the attack specification can be applied. The three first rules are straightforward since $S_2 = \phi$. For the last rule, we have the same situation that $R3$. Suppose that there is no path in the tableau $\sigma[S_0] \rightarrow_{R0} \sigma[S_1] \rightarrow_{R1} \sigma[S_2] \ldots \rightarrow_{Rn} \sigma[S_n]$ such that $|S_n| = 0$. This means that i) the number of arguments that agents have is infinite or that ii) one or several arguments are used several times. However, situation i is not possible because the agents' knowledge bases are finite sets, and situation ii is not allowed in our protocol.

Because the definition of $\hbar$ is based on the size of formulas and since $|S_0| \in N\ (< \infty)$ and $<$ is well-founded in N, it follows that there is no infinite ascending chains of the form $\sigma[S_0]\ \hbar\ \sigma[S_1] \ldots$

4 Implementation

In this section we describe the 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' behaviors.
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.

4.1 General Architecture

Our system consists of two types of agents: conversational agents and trust model agents. These agents are implemented as *Jack*^{TM} agents, i.e. they inherit from the basic class *Jack*^{TM} *Agent*. Conversational agents are agents that take part in the persuasion dialogue. 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 [6].

4.2 Implementation of the Dialogue Games

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 $SC(Ag_1, Ag_2, p)$, agent Ag_1 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: $Create(Ag_1, SC(Ag_1, Ag_2, p))$, etc.

Our dialogue games are implemented as a set of *events* (*MessageEvents*) 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 defense game. The architecture of our conversational agents is illustrated in Figure 3.

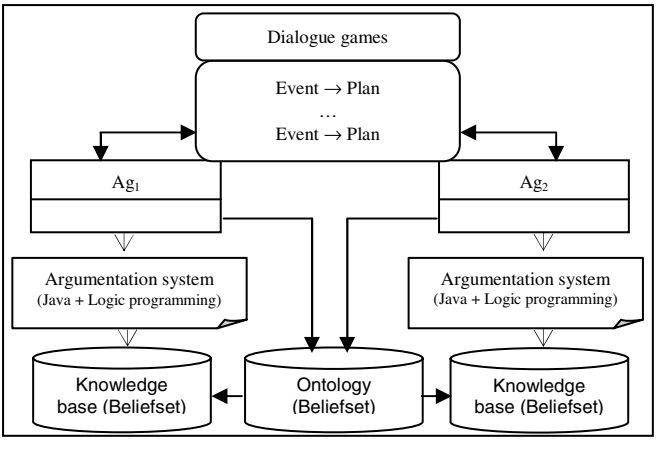


Fig. 3. The architecture of conversational agents

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

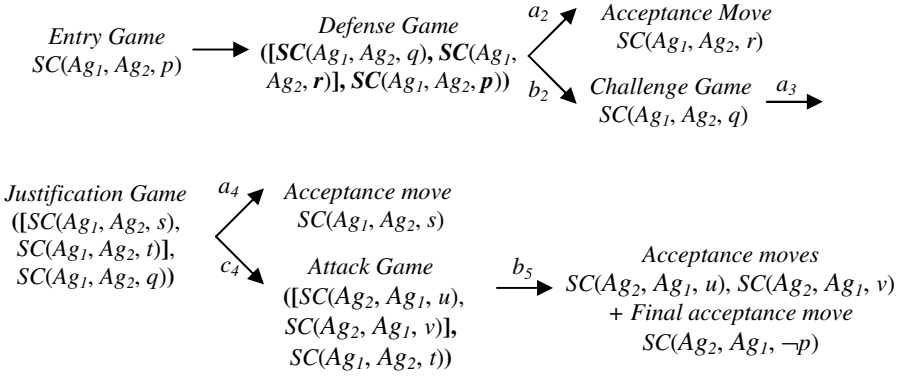
events model goal-directed behavior in agents, rather than plan-directed behavior. What is important is the desired outcome, not the method chosen to achieve it. This type of events allows an agent to pursue long term goals.

4.3 Example

In this section we present a simple example dialogue that illustrates some notions presented in this paper.

> 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 are public (t).
> Ag_2: Information concerning a Minister is not necessarily public, because information I is about the health of Minister (u), and information about the health remains private (v).
> Ag_1: I accept your argument.

This example was also studied in [2] in a context of strategical considerations for argumentative agents. The letters on the left of the utterances are the propositional formulas 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: ($[\neg t]$, $\neg p$), ($[u, v]$, $\neg t$), ($[u]$, u) and ($[v]$, v). The combination of the dialogue games that allows us to describe the persuasion dialogue dynamics is as follows:



Ag_1 creates $SC(Ag_1, Ag_2, p)$ to achieve the goal of persuading Ag_2 that p is true. Ag_1 can create this commitment because it has an argument for p. Ag_2 refuses $SC(Ag_1, Ag_2, p)$ because it has an argument against p. Thus, the entry game is played and the persuasion dialogue is opened. Ag_1 defends $SC(Ag_1, Ag_2, p)$ by creating $SC(Ag_1, Ag_2, q)$ and $SC(Ag_1, Ag_2, r)$. Ag_2 accepts $SC(Ag_1, Ag_2, r)$ because it has an argument for r and challenges $SC(Ag_1, Ag_2, q)$ because it has no argument for q or against q. Ag_1

plays a justification game to justify $SC(Ag_1, Ag_2, q)$ by creating $SC(Ag_1, Ag_2, s)$ and $SC(Ag_1, Ag_2, t)$. Ag_2 accepts the content of $SC(Ag_1, Ag_2, s)$ and attacks the content of $SC(Ag_1, Ag_2, t)$ by creating $SC(Ag_2, Ag_1, u)$ and $SC(Ag_2, Ag_1, v)$. Finally, Ag_1 plays acceptance moves because it has an argument for u and it does not have arguments against v and the dialogue terminates. Indeed, before accepting v, Ag_1 challenges it and Ag_2 defends it by itself (i.e. ([$SC(Ag_2, Ag_1, v)$, $SC(Ag_2, Ag_1, v)$])). Then, Ag_1 accepts this argument because it considers Ag_2 trustworthy. This notion of agent trust and its role as an acceptance criteria of arguments are detailed in [6]. Ag_1 updates its KB by removing the attacked argument and including the new argument. Figure 4 illustrates the screen shot of this example generated by our prototype. In this figure commitments are described only by their contents and the identifiers of the two agents are the two first arguments of the exchanged communicative actions. The contents are specified using a predicate language that the two agents share (the ontology).

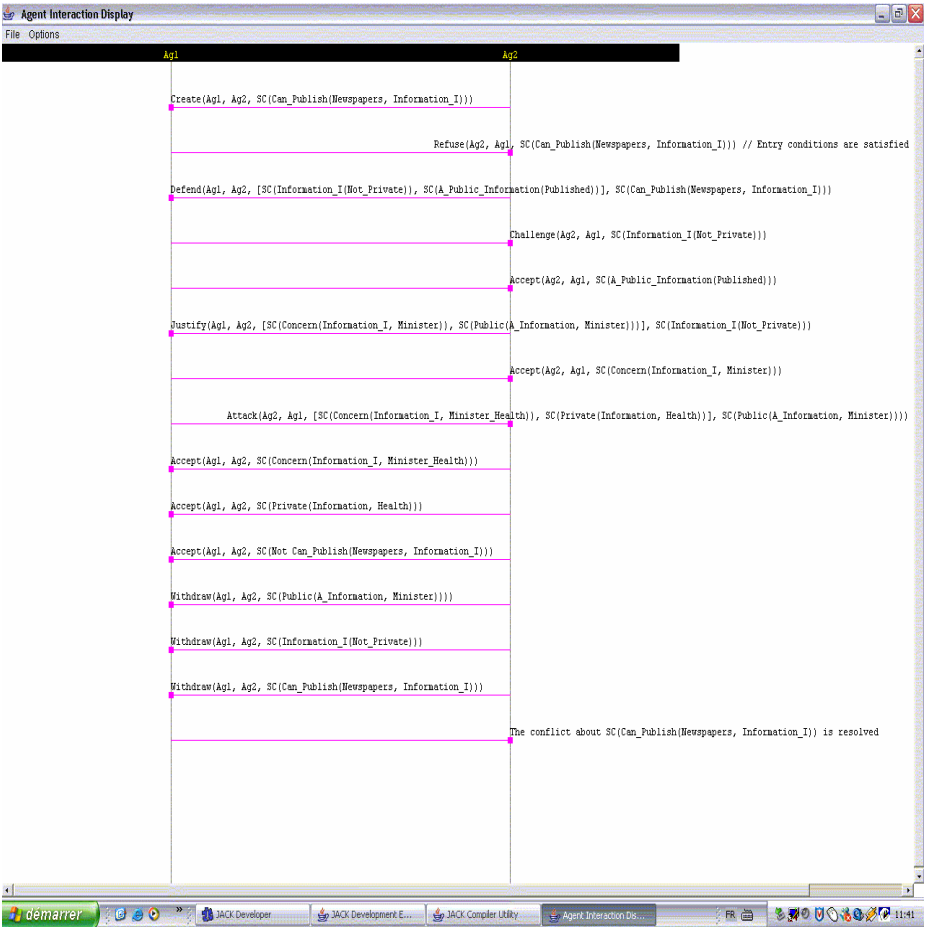


Fig. 4. The example screen shot

5 Related Work

In this section, we compare our protocol with some proposals that have been put forward in two domains: dialogue modeling and commitment based protocols.

1- ***Dialogue modeling*****.** In [1] and [22] Amgoud, Parsons and their colleagues studied argumentation-based dialogues. They proposed a set of atomic protocols which can be combined. These protocols are described as a set of dialogue moves using Walton and Krabbe's classification and formal dialectics. In these protocols, agents can argue about the truth of propositions. Agents can communicate both propositional statements and arguments about these statements. These protocols have the advantage of taking into account the capacity of agents to reason as well as their attitudes (confident, careful, etc.). In addition, Prakken [23] proposed a framework for protocols for dynamic disputes, i.e., disputes in which the available information can change during the conversation. This framework is based on a logic of defeasible argumentation and is formulated for dialectical proof theories. Soundness and completeness of these protocols have also been studied. In the same direction, Brewka [7] developed a formal model for argumentation processes that combines nonmonotonic logic with protocols for dispute. Brewka pays more attention to the speech act aspects of disputes and he formalizes dispositional protocols in situation calculus. Such a logical formalization of protocols allows him to define protocols in which the legality of a move can be disputed. Semantically, Amgoud, Parsons, Prakken and Brewkas' approaches use a defeasible logic. Therefore, it is difficult, if not impossible, to formally verify the proposed protocols.

There are many differences between our protocol and the protocols proposed in the domain of dialogue modeling: 1. Our protocol uses not only an argumentative approach, but also a public one. Locutions are formalized not as agents' private attitudes (beliefs, intentions, etc.), but as social commitments. In opposition of private mental attitudes, social commitments can be verified. 2. Our protocol is based on a combination of dialogue games instead of simple dialogue moves. Using our dialogue game specifications enables us to specify the entry and the exit conditions more clearly. In addition, computationally speaking, dialogue games provide a good balance between large protocols that are very rigid and atomic protocols that are very detailed. 3. From a theoretical point of view, Amgoud, Parsons, Prakken and Brewkas' protocols use moves from formal dialectics, whereas our protocol uses actions that agents apply on commitments. These actions capture the speech acts that agents perform when conversing (see Definition 1). The advantage of using these actions is that they enable us to better represent the persuasion dynamics considering that their semantics is defined in an unambiguous way in a temporal and dynamic logic [5]. Specifying protocols in this logic allows us to formally verify these protocols using model checking techniques. 4. Amgoud, Parsons and Prakkens' protocols use only three moves: assertion, acceptance and challenge, whereas our protocol uses not only creation, acceptance, refusal and challenge actions, but also attack and defense actions in an explicit way. These argumentation relations allow us to directly illustrate the concept of dispute in this type of protocols. 5. Amgoud, Parsons, Prakken and Brewka use an acceptance criterion directly related to the argumentation system, whereas we use an acceptance criteria for conversational

agents (supports of arguments and trustworthiness). This makes it possible to decrease the computational complexity of the protocol for agent communication.

2- ***Commitment-based protocols*****.** Yolum and Singh [28] developed an approach for specifying protocols in which actions' content is captured through agents' commitments. They provide operations and reasoning rules to capture the evolution of commitments. In a similar way, Fornara and Colombetti [17] proposed a method to define interaction protocols. This method is based on the specification of an interaction diagram (ID) specifying which actions can be performed under given conditions. These approaches allow them to represent the interaction dynamics through the allowed operations. Our protocol is comparable to these protocols because it is also based on commitments. However, it is different in the following respects. The choice of the various operations is explicitly dealt with in our protocol by using argumentation and trustworthiness. In commitment-based protocols, there is no indication about the combination of different protocols. However, this notion is essential in our protocol using dialogue games. Unlike commitment-based protocols, our protocol plays the role of the dialectical proof theory of an argumentation system. This enables us to represent different dialogue types as studied in the philosophy of language. Finally, we provide a termination proof of our protocol whereas this property is not yet studied in classical commitment-based protocols.

6 Conclusion and Future Work

The contribution of this paper is the proposition of a logical language for specifying persuasion protocols between agents using an approach based on commitments and arguments. This language has the advantage of 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 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 combined to produce complete and more complex protocols. We formalized these conversation policies as a set of dialogue games, and we described the persuasion dynamics by the combination of five dialogue games. Another contribution of this paper is the tableau-based termination proof of the protocol. We also described the implementation of this protocol. Finally, we presented an example to illustrate the persuasion dynamics by the combination of different dialogue games.

As an extension of this work, we intend to specify other protocols according to Walton and Krabbe's classification [27] 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 [10]. This method can be used to verify the temporal and dynamic aspects of our protocol. Finally, we intend to extend our implementation using ideas from the agent programming language 3APL, namely the concept of cognitive agents. An important characteristic of this language that is interesting for us is its dynamic logic semantics [26] because our protocol is based on an action theory and the semantics of our approach is also based on dynamic logic [5].

Acknowledgements. We'd like to deeply thank the three anonymous reviewers for their valuable comments and suggestions. We'd also like to thank Rance Cleaveland and Girish Bhat for their interesting explanations on the tableau method.

References

1. Amgoud, L., Maudet, N., and Parsons, S. Modelling dialogues using argumentation. In Proc. of 4th Int. Conf. on Multi Agent Systems (2000) 31-38.
2. Amgoud, L., and Maudet, N. Strategical considerations for argumentative agents. In Proc. of 10th Int. Workshop on Non-Monotonic Reasoning (2002) 409-417.
3. Baldoni, M., Baroglio, C., Martelli, A., Patti, V., Schifanella, C. Verifying protocol conformance for logic-based communicating agents. In Proc. of 5th Int. Workshop on Computational Logic in Multi-Agent Systems (2004) 82-97.
4. Bentahar, J., Moulin, B., and Chaib-draa, B. Commitment and argument network: a new formalism for agent communication. In [13] (2004) 146-165.
5. Bentahar, J., Moulin, B., Meyer, J-J. Ch., and Chaib-draa, B. A logical model for commitment and argument network for agent communication (extended abstract). In 3rd Int. J. Conf. on Autonomous Agents and Multi-Agent Systems AAMAS (2004) 792-799.
6. Bentahar, J., Moulin, B., and Chaib-draa, B. Specifying and implementing a persuasion dialogue game using commitment and argument network. In I. Rahwan, P. Moraitis and C. Reed (Eds.), Argumentation in Multi-Agent Systems, LNAI 3366, Springer, (2005). (in press).
7. Brewka, G. Dynamic argument systems: A formal model of argumentation processes based on situation calculus. Journal of Logic and Computation, 11(2) (2001) 257-282.
8. Castelfranchi, C. Commitments: from individual intentions to groups and organizations. In Proc. of Int. Conf. on Multi Agent Systems (1995) 41-48.
9. Chaib-draa, B., and Dignum, F. Trends in agent communication languages. In Computational Intelligence, (18)2 (2002) 89-101.
10. Cleaveland, R. Tableau-based model checking in the propositional mu-calculus. In Acta Informatica, 27(8) (1990) 725-747.
11. Colombetti, M. A commitment-based approach to agent speech acts and conversations. In Proc. of Int. Autonomous Agent Workshop on Conversational Policies (2000) 21-29.
12. Dastani, M., Hulstijn, J., and der Torre, L.V. Negotiation protocols and dialogue games. In Proc. of Belgium/Dutch AI Conference (2000) 13-20.
13. Dignum, F. (Ed.). Advances in Agent Communication. Int. Workshop on Agent Communication Languages. LNAI 2922, Springer, (2004).
14. Dignum, F., and Greaves, M. (Eds.). Issues in agent communication. LNAI 1916, Springer (2000).
15. Elvang-Goransson, M., Fox, J., and Krause, P. Dialectic reasoning with inconsistent information. In Proc. of 9th Conf. on Uncertainty in Artificial Intelligence (1993) 114-121.
16. Endriss, U., Maudet, N., Sadri, F., and Toni, F. Logic_based agent communication protocols. In [13] (2004) 91-107.
17. Fornara, N. and Colombetti, M. Protocol specification using a commitment based ACL. In [13] (2004) 108-127.
18. Greaves, M., Holmback, H., and Bradshaw, J. What is a conversation policy? In [14] (2000) 118-131.
19. Herrestad, H. and Krogh, C. Obligations directed from bearers to counterparties. In Proc. of 5th Int. Conf. on Artificial Intelligence and Law (1995) 210-218.

20. Maudet, N., and Chaib-draa, B. Commitment-based and dialogue-game based protocols, new trends in agent communication languages. In Knowledge Engineering Review, 17(2), Cambridge University Press (2002) 157-179.
21. McBurney, P., and Parsons, S. Games that agents play: A formal framework for dialogues between autonomous agents. In Journal of Logic, Language, and Information, 11(3) (2002) 1-22.
22. Parsons, S., Wooldridge, M., and Amgoud, L. On the outcomes of formal inter-agent dialogues. In Proc. of 2nd Int. J. Conf. on Autonomous Agents and Multi-Agent Systems (2003) 616-623.
23. Prakken, H. Relating protocols for dynamic dispute with logics for defeasible argumentation. In Synthese (127) (2001) 187-219.
24. Singh, M.P. A social semantics for agent communication language. In [14] (2000) 31-45.
25. The Agent Oriented Software Group. *Jack 4.1*. 2004. www.agent-software.com/
26. van Riemsdijk, M.B., de Boer, F.S., and Meyer, J-J. Ch. Dynamic logic for plan revision in intelligent agents. In Proc. of 5th Int. Workshop on Computational Logic in Multi-Agent Systems (2004) 196-211.
27. Walton, D.N., and Krabbe, E.C.W. Commitment in dialogue: basic concepts of interpersonal reasoning. State University of New York Press, NY (1995).
28. Yolum, P. and Singh, M.P. Flexible protocol specification and execution: applying event calculus planning using commitments. In Proc. of 1st Int. J. Conf. on Autonomous Agents and Multi-Agent Systems (2002) 527-534.