2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Towards Practical Argumentation-Based Dialogues in Multi-Agent Systems

Alison R. Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H. Bordini
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Postgraduate Programme in Computer Science – School of Informatics (FACIN)
Porto Alegre, RS – Brazil
Email: alison.panisson@acad.pucrs.br, {felipe.meneguzzi, renata.vieira, rafael.bordini}@pucrs.br

***Abstract*—Although argumentation has been a prominent topic of research in artificial intelligence and in particular agent communication, there has been little work on practical (but provably sound) argumentation approaches integrated with agent programming languages. In this paper, we develop a formally-grounded mechanism for practical argumentation-based dialogues in an agent platform based on a multi-agent programming language. We formalise a protocol to govern such dialogues, where agents use an argumentation-based reasoning mechanism that has been implemented. We prove that dialogues following our protocol always terminate and that ideal solutions are reached under certain conditions. The protocol is simple but was shown to be useful in a multi-agent system application that supports teams of cooperating humans.**

I. Introduction

Argumentation is a rich interdisciplinary area of research that borrows from philosophy, communication studies, linguistics and psychology [1]. Argumentation methods have been improved through collaborative efforts between argumentation theorists and computer scientists. Recent developments encompass the adoption of argumentation models and techniques to topics in artificial intelligence such as MAS (Multi-agent Systems) and artificial intelligence for legal reasoning [2].

This paper focuses on argumentation for MAS, which has two main lines of research [3]: (i) argumentation focused on (nonmonotonic) reasoning over incomplete, conflicting, or uncertain information, where arguments for and against certain conclusions (beliefs, goals, etc.) are constructed and compared; and (ii) argumentation focused on communication/interaction between agents which allows the exchange of arguments to justify a stance and to provide reasons that defend claims. This exchange of additional information allows agents to reach consistent inferences in the presence of multiple diverging points of view, in situations in which other approaches would not. This has been shown to be the case for example in negotiation [4], where argumentation is compared to game-theoretic and heuristic-based approaches.

Although argumentation has received significant interest in the MAS community in recent years, practical works are scarce, the exceptions being [5], [6]. Our work moves in the direction of both formalising and implementing provably finite and ideal argumentation-based dialogues taking into consideration argumentation-based reasoning mechanisms that appeared in the literature and the semantics given for typical argumentation performatives in the context of multi-agent programming languages.

The contributions of this work are twofold. First, we formalise a protocol to govern argumentation-based dialogues between agents. In the formalisation, we define a transition system for the allowed dialogue moves; this has enabled a principled implementation of the protocol as we have exemplified using the Jason platform. Second, we prove two properties of dialogues governed by our protocol: namely that dialogues terminate and that, under certain conditions, dialogues reach the ideal solution.

II. Argumentation-based Reasoning

We assume that our agents have an internal rule-based argumentation mechanism capable of generating and evolving arguments. Rule-based argumentation frameworks can be found in the literature, for example in [7] which extends the well-known work of Dung [8] with structures to arguments based on *strict* and *defeasible* rules, and the work of Berariu [5] and our previous work [6] which extends Jason agents [9] with such argumentative reasoning capabilities.

We assume that agents use a semantics that allows a unique set of acceptable arguments such as *grounded semantics* defined in [8] and used in [5], [7], or the *defeasible semantics* defined in [10] and used by us in [6]. Further, agents only accept propositions/claims which they do not have an acceptable argument against (i.e., the *cautious* attitude [11], [12]), and the agents assert propositions/claims for which they have an acceptable argument (i.e., the *thoughtful* attitude [11], [12]).

An important point to be mentioned is that to define our protocol we will need to define the acceptability of an argument from the agent's perspective (if an agent does or does not have an argument for a given subject/predicate). However, as we are interested in the practical implementation of our work, we assume that our agents have an implemented argumentation-based reasoning as in our previous work [6].

III. Speech-acts for Argumentation-based Communication

In our mechanism, agents argue using a subset of the speech-acts found in the literature of argumentation-based dialogue [13], [11], [12]. The speech-acts used and the informal meaning are as follows:

- *assert:* an agent that performs an assert utterance declares, to all participants of the dialogue, that it is committed to defending this claim – the receivers of the message become aware of this commitment;

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.208

- *accept:* an agent that performs an accept utterance declares, to all participants of the dialogue, that it accepts the previous claim of another agent – the receivers of the message become aware of this acceptance;
- *question:* an agent that performs a question utterance desires to know the reasons for a previous claim of another agent – the receiver of the message is committed to defend its claim, and provide the support set for its claim; and
- *justify:* the justify message is similar to the assert message, and it is the response to the question message previously uttered, in which the agent provide the support to its previous claim.

We adopt the formal definition of the semantics of these speech acts from Panisson *et al.* [14], [15] which specify the exact effect in the agent's circumstance, as well as in the MAS as a whole[1]. The formal semantics definition allows direct implementation of the effects of receiving and sending the speech-act in any agent-oriented programming language based in the 'mental attitudes' described by this specification [15], which makes it possible to fully implement our approach in this aspect. From that work, we make use of the stated effects of each speech act in an agent's commitment store for the specification of our protocol. Those effects are described below.

IV. Rules to update the Commitment Store

The commitment store (CS) consists of one or more structures, accessible to both agents in a dialogue, containing commitments made by the agents during the dialogue[2]. The CS is simply a subset of the knowledge base, and the union of the CSs can be viewed as the global state of the dialogue at a given time [12].

In the course of the dialogue the agents use rules that define how the CS is updated, these rules are implicit in the semantics definition used in this work, and are specified as follows:

- *assert*: the agent's CS is updated with the asserted content p: $CS = CS \cup p$;
- *accept*: the agent's CS is updated with the accepted content p: $CS = CS \cup p$;
- *question*: no effect over the CS; and
- *justify*: the agent's CS is updated with the justified content contained in the set of rules and facts S: $CS = CS \cup S$;

V. Agent Configuration

We assume that two agents a and b (as in [13], [18]) participate in each instance of an argumentation-based dialogue. Each agent has a knowledge base that contains facts and rules. The agents are capable of generating acceptable arguments from this knowledge base, as well as evaluating the acceptability of the arguments [8], when new information is available.

Agents rationally decide their next argumentative move (e.g., `accept`, `question`, etc.) based on their argumentation systems (i.e., depending on whether or not the agents have an argument for or against a certain claim). These decisions taken by the agents correspond to their strategies. Each agent has a commitment store (`CS`) which is accessible to all agents participating in the dialogue, but only the owner agent can update the information in its commitment store (the other agents can only *read* its contents).

The commitment store of each agent is updated following the rules presented in Section IV, depending on the performative used in that interaction. We use $\texttt{CS}_\texttt{a}$ to represent the *commitment store* of agent `a` at the current moment. Agents can build an acceptable argument `S` that supports a claim/predicate `p` (denoted as $\texttt{S} \models \texttt{p}$) from its knowledge base and the commitment store of the other participant. For example, agent `a` can build an acceptable argument `S`, which supports a predicate `p`, from its knowledge base ($\texttt{KB}_\texttt{a}$) and the commitment store of `b` ($\texttt{CS}_\texttt{b}$) (denoted $(\texttt{KB}_\texttt{a} \cup \texttt{CS}_\texttt{b}) \models \texttt{S}$)[3].

In this work, we are interested in cooperative multi-agent systems as we only aim to support teams of cooperating humans (i.e., we are not interested in systems of self-interested agents). Therefore, the agents always take into consideration the information provided by others agents during their reasoning, and agents do not lie. Given our assumption that agents are cooperative, the agents will argue about the specific subject introduced by the proponent of the dialogue, presenting arguments for and against the subject, but they will not argue about the support of each other's arguments.

We use the notation $\overline{\texttt{p}}$ to describe the complement and contrary of a predicate, e.g., we can describe the complement of `p` as $\overline{\texttt{p}}$ where $\overline{\texttt{p}} = \neg\texttt{p}$. We can have `p` as "good", and "good" is the contrary of "bad"; in this case "bad" can be also denoted by $\overline{\texttt{p}}$.

VI. Argumentation-based Dialogue

As described in [19], [20], the topic of discussion in a dialogue needs to be represented in some logical language; in our approach the representation follows our previous work [6]. The elements that correspond to the dialogue game specification in our domain are:

Commencement Rule: an agent can start a dialogue when it needs to argue about the *subject*. To start a dialogue, the agent needs to have an argument that allows it to conclude the assertion (the subject of the dialogue).

Locutions: agents can use the set of locutions from Section III, following the protocol described in Section VI-A, below.

Combination Rules: the combination rules depend on the strategy of the agent (corresponding the agent attitudes to assert and accept claims in the dialogue [12], [11]).

Commitments: the update of commitments of the participants follows the rules of the speech acts used, where each speech-act/performative introduces commitments of participants.

Termination Rules: the dialogue ends when the *proponent* executes the `closedialogue` move, either because the *opponent* accepts the subject of the dialogue (executing an `accept` move); or because it cannot make the *opponent* accept the `subject` of the dialogue.

[1]Due to space limitations, we do not detail these definitions further and refer the reader to [14], [15].

[2]Other names can be founded to CS as *dialogue obligation store* in [16] and *dialogue store* in [17].

[3]The commitment store of the agent `a` ($\texttt{CS}_\texttt{a}$) is a subset of the knowledge base of `a` ($\texttt{KB}_\texttt{a}$), formally $\texttt{CS}_\texttt{a} \subseteq \texttt{KB}_\texttt{a}$.

A. Dialogue Game Protocol

The dialogue game protocol restricts the moves allowed by agents. A dialogue game starts with an agent executing an `assert` move, introducing the subject of the dialogue. When an agent receives an `assert` message, it can either accept, executing an `accept` move, or question, executing a `question` move. When an agent receives a `question` message, it can only execute a `justify` move (the agent is committed to provide an argument which supports the previously asserted predicate). When an agent receives a `justify` message, it can either accept the subject of the dialogue (executing an `accept` move) or provide an argument not to accept the subject of the dialogue (executing a `justify` move). Finally, after receiving a `justify` message, the agent may close the dialogue as well. When an agent receives an `accept` message it may close the dialogue. This dynamic is illustrated in the diagram of Figure 1, which visually represents the transition system we now formally define. Here, the white circle represents the start move, grey circles intermediate moves and the black circle the finish move.

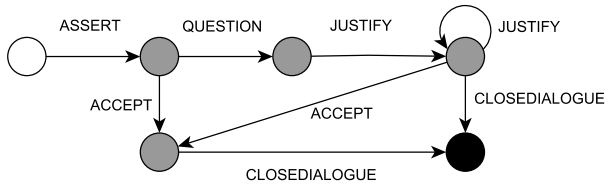

Fig. 1. Allowed Moves.

The protocol also restricts who can execute the moves and who can close the dialogue. The dialogue starts with an agent executing an `assert` move. We call the agent that starts the dialogue the *proponent*. An agent can never repeat a move with the same content; this is guaranteed by dialogue rules and the structure which maintains the information introduced in the dialogue as commitments (the *commitment store*). The dialogue can be closed only by the proponent, so there is only a single agent that can execute the `closedialogue` move in a particular dialogue, namely the *proponent*. The dialogue ends when the proponent executes a `closedialogue` move. We prove that a dialogue following our protocol will always terminate in the next sections.

B. Dialogue Rules

Now, we describe the *dialogue rules* that govern the interactions between the agents, where each agent moves by performing the allowed utterances. These rules (which correspond to a dialogue game [20]) are expressed as if-then rules, which are then easy to implement. The dialogue rules specify the moves each player can make, and so specify the *protocol* under which the dialogue takes place [13].

Definition 1 (Dialogue Game Protocol)**.** *A dialogue game protocol is formally represented as a tuple* $\langle \mathtt{MO}, \mathtt{DI} \rangle$*, where* `MO` *is a finite set of allowed moves, and* `DI` *a set of dialogue rules.*

Definition 2 (Dialogue Move)**.** *We denote a move in* `MO` *as* $\mathtt{M}^{\mathtt{i}}(\alpha, \beta, \mathtt{cont}, \mathtt{t})$*, where* `i` *is the type of move made by agent* α *and addressed to agent* β *at time* `t` *regarding content* `cont`*.*

We consider the following set of types of moves, denoted by `P` *(q.v. Section 3):* `assert`*,* `accept`*,* `question`*,* `justify`*, and* `closedialogue`*. The content of a move (*`cont`*) can be an argument (a set of predicates and rules) or a predicate (e.g., in an* `assert` *move the content is a predicate and in a* `justify` *move the content will be an argument that supports a claim/predicate uttered in a previous* `assert` *move).*

The dialogue rules in `DI` indicate the possible moves that an agent can make following a previous move of the other agent. The formalisation we give here follows the work of Bentahar *et al.* [21]. To define the dialogue rules, we use a set of condition (denoted by `C`) which reflect the agents' strategies. Formally, we have:

Definition 3 (Dialogue Rules)**.** *Dialogue rules can assume one of two forms:*

1) First, we have dialogue rules that specify which moves are allowed given the previous move and conditions (corresponding to the combination rules of the dialogue game).

$$\bigwedge_{\substack{0<\mathtt{k}\leq \mathtt{n_i},\\ \mathtt{i},\mathtt{j}\in \mathtt{P}}} (\mathtt{M}^{\mathtt{i}}(\alpha, \beta, \mathtt{cont}, \mathtt{t}) \wedge \mathtt{C_k} \Rightarrow \mathtt{M}^{\mathtt{j}}_{\mathtt{k}}(\beta, \alpha, \mathtt{cont_k}, \mathtt{t}'))$$

where `P` *is the set of move types,* $\mathtt{M}^{\mathtt{i}}$ *and* $\mathtt{M}^{\mathtt{j}}$ *are in* `MO`*,* $\mathtt{t} < \mathtt{t}'$ *and* $\mathtt{n_i}$ *is the number of allowed communicative acts that* β *could perform after receiving a move of type* `i` *from* α*.*

2) Second, we have the initial conditions (corresponding to the commencement rules of the dialogue game), which do not require that any move was previously executed.

$$\bigwedge_{\substack{0<\mathtt{k}\leq \mathtt{n},\\ \mathtt{j}\in \mathtt{P}}} (\mathtt{C_k} \Rightarrow \mathtt{M}^{\mathtt{j}}_{\mathtt{k}}(\alpha, \beta, \mathtt{cont_k}, \mathtt{t}_0)$$

where $\mathtt{t}_0$ *is the initial time and* `n` *is the number of allowed moves that* α *could make initially.*

1) ***Initial Rule****:* The first move (commencement rule) introduces the subject of the dialogue (where `subject(p)` means that the predicate `p` is the subject of the dialogue). In our approach, each dialogue has only one `subject`. The agent that introduces the subject of the dialogue is called *proponent*, and the other agent participating in the dialogue is called *opponent* (we use `Pr` and `Op`, respectively, to refer to them).

$\mathtt{C}_{in1} \Rightarrow \mathtt{assert}(\alpha, \beta, \mathtt{p})$
Where:
$\mathtt{C}_{in1} = \exists \mathtt{S},\ \mathtt{S} \models \mathtt{p} : (\mathtt{KB}_\alpha \cup \mathtt{CS}_\beta) \models \mathtt{S} \wedge \mathtt{subject}(\mathtt{p})$

The dialogue starts when an agent needs to argue about a given subject. The initial rule restricts that an agent needs to have an argument that defends its claim in order to start an argumentation-based dialogue (as the agent will be committed to defending the initial assertion).

2) ***Assert Rules****:* We have two dialogue rules that restrict the possible next move for agents to respond to an `assert` move:

$\mathtt{assert}(\alpha, \beta, \mathtt{p}) \wedge \mathtt{C}_{as1} \Rightarrow \mathtt{accept}(\beta, \alpha, \mathtt{p})$
$\mathtt{assert}(\alpha, \beta, \mathtt{p}) \wedge \mathtt{C}_{as2} \Rightarrow \mathtt{question}(\beta, \alpha, \mathtt{p})$
Where:
$\mathtt{C}_{as1} = \nexists \mathtt{S},\ \mathtt{S} \models \overline{\mathtt{p}} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}$
$\mathtt{C}_{as2} = \exists \mathtt{S},\ \mathtt{S} \models \overline{\mathtt{p}} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}$

The options of the agent are: (i) to accept the previous claim (the subject asserted in the dialogue), where condition $\mathtt{C}_{as1}$ means that the agent will accept a claim if it has no argument against it; and (ii) when the agent has an argument against the previous assertion, $\mathtt{C}_{as2}$, the agent will question the other agent to provide the support of its previous claim.

3) ***Question Rule****:* The dialogue rule that restricts the moves after an agent receives a `question` message is:

$$\mathtt{question}(\alpha, \beta, \mathtt{p}) \wedge \mathtt{C}_{qs1} \Rightarrow \mathtt{justify}(\beta, \alpha, \mathtt{S})$$

Where:

$$\mathtt{C}_{qs1} = \exists \mathtt{S},\ \mathtt{S} \models \mathtt{p} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}$$

As the agent has asserted a predicate `p` previously (which allowed the `question` move), the agent is committed to defend its claim in the dialogue, so it will provide the support to this claim.

4) ***Justify Rules****:* We have four dialogue rules to restrict the moves to respond to a `justify` move:

$$\mathtt{justify}(\alpha, \beta, \mathtt{S}) \wedge \mathtt{C}_{js1} \Rightarrow \mathtt{accept}(\beta, \alpha, \mathtt{p})$$
$$\mathtt{justify}(\alpha, \beta, \mathtt{S}) \wedge \mathtt{C}_{js2} \Rightarrow \mathtt{justify}(\beta, \alpha, \mathtt{S}')$$
$$\mathtt{justify}(\alpha, \beta, \mathtt{S}) \wedge \mathtt{C}_{js3} \Rightarrow \mathtt{closedialogue}(\beta, \alpha)$$
$$\mathtt{justify}(\alpha, \beta, \mathtt{S}) \wedge \mathtt{C}_{js4} \Rightarrow \mathtt{justify}(\beta, \alpha, \mathtt{S}')$$

Where:

$$\mathtt{C}_{js1} = \nexists \mathtt{S}',\ \mathtt{S}' \models \overline{\mathtt{p}} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}' \wedge \mathtt{S}' \notin \mathtt{CS}_\beta \wedge \mathtt{Op}(\beta) \wedge \mathtt{subject}(\mathtt{p})$$
$$\mathtt{C}_{js2} = \exists \mathtt{S}',\ \mathtt{S}' \models \overline{\mathtt{p}} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}' \wedge \mathtt{S}' \notin \mathtt{CS}_\beta \wedge \mathtt{Op}(\beta) \wedge \mathtt{subject}(\mathtt{p})$$
$$\mathtt{C}_{js3} = \nexists \mathtt{S}',\ \mathtt{S}' \models \mathtt{p} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}' \wedge \mathtt{S}' \notin \mathtt{CS}_\beta \wedge \mathtt{Pr}(\beta) \wedge \mathtt{subject}(\mathtt{p})$$
$$\mathtt{C}_{js4} = \exists \mathtt{S}',\ \mathtt{S}' \models \mathtt{p} : (\mathtt{KB}_\beta \cup \mathtt{CS}_\alpha) \models \mathtt{S}' \wedge \mathtt{S}' \notin \mathtt{CS}_\beta \wedge \mathtt{Pr}(\beta) \wedge \mathtt{subject}(\mathtt{p})$$

The agent will accept the subject of the dialogue, $\mathtt{C}_{js1}$, if the justification received from the proponent has changed the agent's conclusion, otherwise the agent will justify why it cannot accept the subject, $\mathtt{C}_{js2}$. The agent cannot accept the subject because it still has an argument against it, even after receiving this new information. In the case in which the agent that receives the justify from the opponent but cannot itself reach the same conclusion given the new information received (i.e., the agent does not have an acceptable argument for the subject), the agent closes the dialogue, $\mathtt{C}_{js3}$. In the final case, the agent sends the new argument[4] to support the subject of the dialogue, $\mathtt{C}_{js4}$.

5) ***Accept Rule****:* The dialogue rule that restricts the moves when an agent receives an `accept` message is:

$$\mathtt{accept}(\alpha, \beta, \mathtt{p}) \wedge \mathtt{C}_{ac1} \Rightarrow \mathtt{closedialogue}(\beta, \alpha)$$

Where:

$$\mathtt{C}_{ac1} = \mathtt{subject}(\mathtt{p}) \wedge \mathtt{Pr}(\beta)$$

When the agent receives an accept move it will close the dialogue. Only the proponent will receive an accept move when the opponent accepts the subject of the dialogue.

[4]The argument is new because, as defined in the protocol, the agent cannot repeat a move with the same content.

C. Properties of the Protocol

After we have defined the protocol, it is interesting to demonstrate its effectiveness through the properties commonly found in the literature. One of the more important properties to be proved over a protocol is that a dialogue, following such protocol, will always terminate.

Theorem 1 (Termination)**.** *Any argumentation-based dialogue, following the protocol defined above, eventually terminates.*

Proof*:* Considering that at least one agent (the agent α) needs to have `p` as acceptable (where we use `p` as the `subject` of the dialogue), the initial configurations are restricted to two:

In the first case, where $\mathtt{KB}_\alpha \models \mathtt{p}$ and $\mathtt{KB}_\beta \not\models \overline{\mathtt{p}}$, agent α will introduce `p` in the dialogue using the `assert` move; as agent β has no argument against (i.e., no argument for $\overline{\mathtt{p}}$), following the dialogue rule condition $\mathtt{C}_{as1}$ the agent will accept `p` executing the `accept` move. Agent α receives the `accept` message and, following the dialogue rule condition $\mathtt{C}_{ac1}$, closes the dialogue.

In the second case, where $\mathtt{KB}_\alpha \models \mathtt{p}$ and $\mathtt{KB}_\beta \models \overline{\mathtt{p}}$, agent α, as before, will introduce `p` in the dialogue using the `assert` move. As agent β has an argument against (i.e., an argument to $\overline{\mathtt{p}}$), agent β, following the dialogue rule condition $\mathtt{C}_{as2}$, will execute the `question` move. Agent α receives the `question` message and, following the dialogue rule condition $\mathtt{C}_{qs1}$, executes the `justify` move with the argument which support `p` (the existence of this argument is guaranteed by initial dialogue rule condition $\mathtt{C}_{in1}$ which allowed the agent to start the dialogue). The agent β receives this new information and has two options: (i) When the new information changes the acceptability of `p` to agent β (the agent has no acceptable argument for $\overline{\mathtt{p}}$ which has not yet been introduced in the dialogue), then the agent, following the dialogue rule condition $\mathtt{C}_{js1}$, accepts `p` (executing the `accept` move). The process continues as in the first case, where agent α receives the accept message and closes the dialogue; (ii) Otherwise, even considering the new information received, agent β still has an argument for $\overline{\mathtt{p}}$ which has not yet been introduced in the dialogue, then the agent, following the dialogue rule condition $\mathtt{C}_{js2}$, will execute a `justify` move. The agent α receives the `justify` message and has two options: (i) Agent α closes the dialogue, following the dialogue rule condition $\mathtt{C}_{js3}$, because it has no acceptable argument to support `p` that has not yet been introduced in the dialogue; (ii) The agent α introduces a new acceptable argument, considering the new information, following the dialogue rule condition $\mathtt{C}_{js4}$, support `p`. As the knowledge bases of the agents are finite and the `justify` move cannot be repeated with the same content, as formally defined in the dialogue rule conditions $\mathtt{C}_{js2}$ and $\mathtt{C}_{js4}$, eventually agent α will close the dialogue, because it has no acceptable argument that has not yet been introduced in the dialogue or agent β will accept `p` because it has no acceptable argument for $\overline{\mathtt{p}}$ that has not been introduced in the dialogue, and α will close the dialogue as in the first case. Therefore, any argumentation-based dialogue following that protocol eventually terminates. ■

Termination of a dialogue is an important and well-known property of protocols. However, as described by Amgoud and Vesic [22], except the termination of each dialogue generated under those protocols, nothing is said on their quality. Therefore, we will also prove such properties demonstrating that the

dialogues following our protocol will always end with the best solution (*ideal* solution [22]) when it exists.

As described by Amgoud and Vesic [22], the *ideal solution* is the best solution in the general and is time-independent. Considering the reasoning mechanism used, the ideal solution is the result achieved when the agents have all information related to the `subject` (all arguments for and against the `subject`). This definition is characterised as *integrative*, where both sides are looking for solutions that are "good" for everyone [23]. A situation is also called "ideal" where the agents have complete information about the others agents, suggesting the integration of the theories of all agents into a single theory (the so-called *aggregate argumentation system*) [23]. According to [23] this integration of all arguments (agent theories) allows a centralised reasoning mechanism to identify the outcomes that are "good" for all agents participating in the dialogue. However our work differs from [23] because we do not have a centralised reasoning mechanism, but with the exchange of all arguments related to the `subject` the agents can, rationality, reach the same conclusion of a centralised reasoning mechanism.

Definition 4 (Ideal Solution)**.** *The ideal solution is the conclusion resulting from the all information related to* `subject` *(arguments and preferences regarding* `subject`*). The conclusion is whether the* `subject` *is acceptable, it is not acceptable, or it is not possible to determine the acceptability of the* `subject` *(i.e., the ideal solution does not exist).*

As in the real life, arguing does not necessarily lead to an agreement, it may be the case that two agents will exchange arguments and at the end the dialogue fail to agree about the `subject` [22]. This disagreement is caused by different preferences between the agents, but even with the disagreement, in the worst case, argumentation-based dialogue improves the choices made by each agent (using the additional information exchanged). Therefore, as described by Amgoud and Vesic [22], argumentation may improve the quality of the outcome but never decrease it.

To prove that the agents, using our protocol, strategies, and reasoning mechanism defined will reach the *ideal solution* (when it exists), we will consider the union of the knowledge bases (as the work of Dimopoulos *et al.* [23]).

Theorem 2 (Ideal Solution)**.** *The ideal solution for a dialogue with subject* $\mathtt{p}$ *is for both agents to agree about* $\mathtt{p}$ *when* $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \mathtt{p}$ *and for both agents to conclude* $\overline{\mathtt{p}}$ *if* $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \overline{\mathtt{p}}$*; otherwise, the ideal solution will not exist (neither* $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \mathtt{p}$ *nor* $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \overline{\mathtt{p}}$ *hold).*

Proof: First, we will prove that the *ideal solution* $\mathtt{p}$ is agreed upon by both agents when $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \mathtt{p}$. As $\mathtt{p}$ is always acceptable to the proponent (we refer to the proponent in this proof as α), the case where $\mathtt{KB}_\alpha \models \overline{\mathtt{p}}$ and $\mathtt{KB}_\beta \models \overline{\mathtt{p}}$, and the case where $\mathtt{KB}_\alpha \models \overline{\mathtt{p}}$ and $\mathtt{KB}_\beta \models \mathtt{p}$ do not exist. Therefore, the possible cases are limited to two:

1) $\mathtt{KB}_\alpha \models \mathtt{p}$ and $\mathtt{KB}_\beta \not\models \overline{\mathtt{p}}$. In this case the proponent agent will start the dialogue asserting $\mathtt{p}$ (using the `assert` move) and the other agent will execute the `accept` move (the other agent does not have an acceptable argument against), hence terminating the dialogue with both agents agreeing on $\mathtt{p}$, i.e., the ideal solution in this case. Note that, as we are assuming in this part of the proof that $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \mathtt{p}$, agreement on $\mathtt{p}$ is indeed the ideal solution.

2) $\mathtt{KB}_\alpha \models \mathtt{p}$ and $\mathtt{KB}_\beta \models \overline{\mathtt{p}}$. In this case the agent identified by α starts the dialogue using the move `assert`, the agent identified by β will execute the move `question` because, as yet, $\overline{\mathtt{p}}$ is acceptable to it. The agent α does a `justify` move; if this argument changes the conclusion of agent β, the agent accepts the `subject` (executing the move `accept`). Otherwise, agent β will send its arguments for not accepting the subject (doing another `justify` move). The agents will exchange arguments until a new argument from α changes the conclusion of β and agent β accepts the `subject` of the dialogue (executing the `accept` move), given that $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \mathtt{p}$. It follows that the ideal solution $\mathtt{p}$ is reached.

The part of the proof for when $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \overline{\mathtt{p}}$ is similar to the one above. The difference is that, in case two (i.e., when agents initially disagree), at a certain moment agent β will introduce an argument (using the `justify` move) where the new information will change the acceptability of the `subject` to agent α, and it will then close the dialogue. Given that $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \overline{\mathtt{p}}$, it follows that $\overline{\mathtt{p}}$, the *ideal solution*, is reached through the dialogue following our protocol.

In the last case, where the *ideal solution* does not exist, the agents exchange arguments with the `justify` move and the dialogue terminates in disagreement. As before, the agents initially disagree about the `subject`, at a certain moment agent α will introduce an argument (using the `justify` move) where the new information will not make the `subject` acceptable for agent β. Agent β will justify its position executing a `justify` move hence not accepting the `subject`, when the new information does not change the acceptability of `subject` for agent α. At this moment, if agent α does not satisfy the dialogue rule condition $\mathtt{C}_{js3}$, i.e., it has no new argument which supports the `subject`, the agent will close the dialogue and the dialogue will ends in disagreement. Otherwise, a new round of `justify` moves will occur. Given that neither $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \mathtt{p}$ nor $(\mathtt{KB}_\alpha \cup \mathtt{KB}_\beta) \models \overline{\mathtt{p}}$ hold, the dialogue ends with the agents disagreeing about the `subject`. ■

An important point in our proof is that the non-existence of the *ideal solution* is a consequence of agent preferences. When individual preferences are considered, an agent's own argument will be acceptable even with the argument against presented by the other agent. For example, agent α executes the `justify` move, introducing an argument to $\mathtt{p}$ in the dialogue, agent β receives this information and uses the `justify` move because it has an argument to $\overline{\mathtt{p}}$, agent α receives this information but its own argument is still acceptable. At this moment, the agent detects that there exist different preferences between the agents and, following the dialogue rule condition $\mathtt{C}_{js3}$, the agent closes the dialogue.

VII. Example

As an example, we will describe a scenario of an application developed in the context of assisted living [24]. The application provides functionalities such as activity recognition and task reallocation among agents representing human users through the use of planning, agent, and semantic technologies.

A user of the application, named Paul, has a new appointment and he will not be able to take his father, named John,

to physiotherapy. Then Paul's agent will try to reallocate the task. First Paul's agent checks if the task can be postponed; in this case the task can, normally, be postponed. With this information Paul's agent initiates a dialogue with John's agent, asserting that Paul cannot execute the task. John's agent receives the message and executes a `question` move, because the agent has an argument against, with information that John is in severe pain that day, and this implies that someone needs to take John to the physiotherapy. Paul's agent receives the `question` message and provides the justification using the `justify` move with the argument that allowed the initial *assert* message, more specifically: that a physiotherapy appointment can be postponed. John's agent receives this message, but this information does not change its conclusion, then John's agent executes the `justify` move with the strict rule that John is in severe pain, and this implies that someone needs to take John to the physiotherapy. Paul's agent receives this new information and concludes that someone needs to take John to physiotherapy. Paul's agent closes that dialogue and checks, in the application's ontology, who can execute the task. In this case "Paul" and "Jane" can execute this task (because they are adults and can drive). Paul's agent starts a new dialogue with Jane's agent using the `assert` move, suggesting that Jane executes the task. Jane's agent receives the `assert` message and executes a `question` move, because it knows that the task can be postponed. Paul's agent sends the justification (argument) that someone needs to execute the task because John is in pain. Jane's agent receives this new information and accepts to execute the task[5].

VIII. Implementation

In this section we describe how the protocol can be implemented using the Jason platform[6] [9]. As described above, we assume that agents have an internal rule-based argumentation system. In the protocol implementation we need that the agent can itself query the acceptability of a given content. Considering our previous work in argumentation-based reasoning [6], the agents can query the acceptability of an argument in its internal reasoning process using a special predicate *argument(content, Arg)* where *Arg* is unified with the rules and facts used to derive the content. If such an argument does not exist, i.e., the *content* is not acceptable to the agent, the query will fail as in logic-based programming. Suppose the plan below:

```
+!argue_about(Content,Op): argument(Content,Arg)
            <- !start_dialogue(Content,Op).
```

In this example, an agent has a plan to achieve the goal of to argue about a given *content*. The agent itself needs to have an argument for this content in order to use this plan (the plan context or condition is *argument(content,Arg)*), then it can start a dialogue using the argument. In the case where the context is not applicable – i.e., the argument query fails – the plan cannot be used to achieve the agent's goal.

Following our protocol definition, an agent always starts a dialogue using an `assert` move. In *Jason*, a move corresponds to an action for sending a message using the internal action[7] *.send(receiver, performative, content)*. Therefore, the move $\texttt{assert}(\alpha, \beta, p)$ corresponds to an agent α sending a message of type `assert` with content p to agent β – i.e., $.send(\beta, assert, p)$ executed by Jason agent α.

```
+!start_dialogue(Subj,Op): argument(Subj,Arg)
            <- .send(Op,assert,Subj).
```

The plan above implements the *initial rule* from our protocol, which allows an agent to start a dialogue if it has an argument for the given subject ($Subj$) which will be discussed in the dialogue.

The other dialogue rules are of the format "$move \wedge condition \Rightarrow move$", where an agent receives a message and given a certain condition sends another message as a response. Receiving message in Jason is treated using plans, similar to the ones that treat receiving KQML performatives [26] already implemented in Jason.

```
@kqmlReceivedAssert
+!kqml_received(Sender, assert, Subj, MsgId)
            <- !respond(Sender,assert,Subj).

+!respond(Sender,assert,Subj): comp(Subj,Comp) &
                           not(argument(Comp,Arg))
            <- .send(Sender,accept,Subj).

+!respond(Sender,assert,Subj): comp(Subj,Comp) &
                           argument(Comp,Arg)
            <- .send(Sender,question,Subj).
```

The plans above implement the *assert rules* from our protocol. Following the example, when an agent receives a message with the performative `assert` (an `assert` move) it will then have a goal to respond to this message. The response, as described in the *assert rules*, depends on whether the agent does or does not have an argument against the subject of the dialogue. In the first case, where the agent does not have an argument against, it accepts the subject executing an `accept` move. Otherwise, when the agent does have an argument against (queried using the predicate *comp*), it questions the other agent executing a `question` move.

The remainder of the dialogue rules are implemented similarly to the examples above. The examples above were given to demonstrate that our formalisation can be intuitively/directly implemented in Jason. In a real application, such as the scenario described in Section VII, the agents will have other actions and subgoals in the course of the their plans because they need to interact with the users or to adopt the result of a dialogue as goal to be achieved. However, we argue that the plans presented here are enough to demonstrate how the protocol we devised can be completely implemented, as we have done in practice.

IX. Related Work

Some protocols can be found in the literature of argumentation-based dialogues in MAS, most focused in negotiation. An example is the work of Kakas and Moraitis [27], which proposed a protocol that is sensitive to the context and roles of the agents in which the agents can adapt their negotiation strategies and offers, as their environment changes,

[5]Jane's agent, as usual, queries Jane to ask for confirmation on the proposed course of action.

[6]Jason extends the AgentSpeak(L) language [25], and we refer the reader to [9] for details about Jason.

[7]The internal actions available in the *Jason* platform can be found in [9].

and when they exchange information within the negotiation. The agents can build arguments through a theory (formed by their knowledge base and goals). The work is based on argumentation systems with dynamic preference proposed by the same authors in [28], [29]. Differently of [27], we develop a protocol where both sides are looking for solutions that are "good" for everyone and this protocol is fully implementable as described in the Serction VIII.

More recently, Amgoud and Vesic [22] analyse the role of argumentation in negotiation dialogues, proposing an abstract framework for argumentation-based negotiation. The authors argue that argumentation can improve the quality of an outcome but never decrease it. The authors in [22], as in [27], argue that, by exchanging arguments, the theories of the agents (i.e., their mental states) may evolve and thus the status of offers may change, allowing *optimal solutions* for both agents. The protocol to argumentation-based negotiation is described as moves where the agents exchange offers. The paper describe that optimal solution do not dependent on dialogue step, they are offers that an agent would choose if it had access to all arguments owned by the other agent. New arguments allow agents to revise their mental states, thus the best decision for an agent is the one it makes under *complete* information. This situation, also, is described in [30], where the work shows how the beliefs of two agents that engage in an argumentation-based dialogue will converge over time (the new information changes the agents' conclusions). We have taken into consideration the important properties presented by [22], where we prove that dialogues following our protocol will achieve the optimal solution. This is an important aspect to be considered, mainly in the development of real applications as we have done.

Rueda and Martínez in [31] propose an interaction language that allows argumentation-based dialogue among collaborative BDI agents. All members of the system are autonomous and rational entities, but have a collaborative attitude (as in our work). Each agent elaborates arguments as part of its own planning process and justifies its proposals, counter-proposals, and rejections during the negotiation process [31]. The interaction language proposed in this work is presented in the form of *preconditions*, *meaning* (informal meaning), *response* (the possible responses for this locution) and *updates* (the effects over the belief base of the agents). The representation used by [31], although different from ours, has the same components. We can quote the *precondition* as receiving a move, which given a condition (acceptability of the content received) allows a *response*. However, in [31] only an informal interaction language is presented.

Hussain and Toni in [32] demonstrate the benefits of the use argumentation-based dialogues in a scenario of resource reallocation. It is assumed that the resources are unique and indivisible. In this work, the agents always justify their requests and responses. The requests are justified with the information that the agent needs the resource and does not have it. The responses given by agents depend on the information that they have, e.g., if an agent knows who has the resource requested, the agent can refuse to give the resource because it does not have it but inform that another agent has the resource, allowing the requesting agent to directly contact that agent. Although simple, the work presented in [32] shows benefits in "real" scenarios which make use of argumentation-based dialogues. In our work, we follow [32] developing and implementing applications which make use of argumentation-based dialogues.

Oliva et al. in [33] propose a conceptual framework for argumentation-based negotiation called SANA (Supporting Artifacts for Negotiation with Argumentation) including the SANAP protocol. In that work, artifacts play an important role in the framework, monitoring/assisting the participating agents, inferring mutually-accepted proposals, etc. Differently from [33], we use artifacts only to store the agents' commitments in dialogues, i.e., argument evaluation and generation is done by the agents themselves through reasoning. Therefore, the agent decision-making is based on its internal reasoning over the arguments and the possible dialogue moves available to them at that moment. Further, our work focuses on argumentation-based dialogues in an agent-oriented programming language, also differently from [33], which focuses on an infrastructure (provided by artifacts) for agents to use in argumentation-based negotiation.

Other protocols and dialogue mechanisms can be found in the literature, as the work of Rahwan and Larson [34], which introduce the called *argumentation mechanism design* (ArgMD) which enables the design and analysis of argumentation mechanism for self-interested agents. The ArgMD also is used in [35], where a number of preference relations over argumentation outcomes are analysed. The main difference of our work is that we do not use a centralised mechanism to decide the acceptability of arguments and to access the result of the dialogues. The agents' argumentation-based reasoning mechanism allows the agents rationally reach the same result of a centralised reasoning mechanism with the exchange of information related to the subject of the dialogue. The same authors, in [36], study argumentation-based dialogues over a semantics which produces only a set of acceptable arguments (as in our work). The authors assume that the agents have focal arguments and the goal of the agents is to have their focal argument accepted. Our approach differs from that, because we have no focal arguments, but the agents in our systems have the goal of agreeing or not about a subject (we can call of *topic of discussion*).

X. Conclusion

In this paper, we have formalised a practical protocol for argumentation-based dialogues using a transition system that is easy to implement. The resulting formalisation allowed us not only to implement a protocol for argumentation-based dialogues but also to prove fundamental properties about the dialogues generated by the protocol, providing a solid basis for building MAS with provable properties. The contributions of the work are twofold. First, we formalise a protocol to govern argumentation-based dialogues between agents. The formalisation takes into consideration practical (implemented) argumentation-based agent reasoning mechanisms as well as agents' individual strategies. Second, we prove two properties of dialogues governed by our protocol, namely, that dialogues always terminate, a that they reach the ideal solution under certain conditions. Termination is a well-known and fundamental property to be proved, which ensures that argumentation-based dialogues will not waste agent resources indefinitely. Moreover, reaching the ideal solution, although clearly an important

property, has as yet been poorly explored in the literature, perhaps because it has been just recently defined [22]. Both properties are generally studied using a centralised reasoning mechanism [23]. In our approach, we use the agent's internal reasoning mechanism (i.e., rational agents) to achieve such properties, which is a more realistic approach for distributed systems, such as MAS.

To the best of our knowledge, the only other practical approaches that integrate argumentation with agent programming languages are [5], [6], both of which focus on the reasoning mechanism, rather than dialogues, and lack formal proofs of their properties. Therefore, this work provides an important advance to the practical implementation of argumentation as a mechanism for agent dialogues. In fact, we have implemented and tested a version of our approach in the Jason [9] programming language. Thus, our mechanism can be used to implement argumentation-based negotiation and other dialogue types described in the literature which still lack concrete and general implementations. As future work, we intend to extend the work presented here to other application domains, including domains with self-interested agents.

ACKNOWLEDGMENT

Part of the results presented in this paper were obtained through research on a project titled "Semantic and Multi-Agent Technologies for Group Interaction", sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91.

REFERENCES

[1] F. H. Van Eemeren, R. Grootendorst, R. H. Johnson, C. Plantin, and C. A. Willard, *Fundamentals of argumentation theory: A handbook of historical backgrounds and contemporary developments*. Routledge, 2013.

[2] I. Rahwan and G. R. Simari, *Argumentation in Artificial Intelligence*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[3] N. Maudet, S. Parsons, and I. Rahwan, "Argumentation in multi-agent systems: Context and recent developments," in *Argumentation in Multi-Agent Systems*. Springer, 2007, pp. 1–16.

[4] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg, "Argumentation-based negotiation," 2004.

[5] T. Berariu, "An argumentation framework for bdi agents," in *Intelligent Distributed Computing VII*. Springer, 2014, pp. 343–354.

[6] A. R. Panisson, F. Meneguzzi, R. Vieira, and R. H. Bordini, "An Approach for Argumentation-based Reasoning Using Defeasible Logic in Multi-Agent Programming Languages," in *11th International Workshop on Argumentation in Multiagent Systems (ArgMAS)*, 2014.

[7] H. Prakken, "An abstract framework for argumentation with structured arguments," *Argument and Computation*, vol. 1, no. 2, pp. 93–124, 2011.

[8] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, pp. 321–357, 1995.

[9] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[10] G. Governatori, M. J. Maher, G. Antoniou, and D. Billington, "Argumentation semantics for defeasible logic." *J. Log. Comput.*, vol. 14, no. 5, pp. 675–702, 2004.

[11] S. Parsons and P. McBurney, "Argumentation-based dialogues for agent co-ordination," *Group Decision and Negotiation*, vol. 12, no. 5, pp. 415–439, 2003.

[12] S. Parsons, M. Wooldridge, and L. Amgoud, "An analysis of formal inter-agent dialogues," in *Proc. first international joint conference on Autonomous agents and multiagent systems: part 1*, ser. AAMAS '02. New York, NY, USA: ACM, 2002, pp. 394–401.

[13] L. Amgoud, N. Maudet, and S. Parsons, "Modeling dialogues using argumentation." in *ICMAS*. IEEE Computer Society, 2000, pp. 31–38.

[14] A. R. Panisson, F. Meneguzzi, M. Fagundes, R. Vieira, and R. H. Bordini, "Formal semantics of speech acts for argumentative dialogues," in *Proc. Thirteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014, pp. 1437–1438.

[15] A. R. Panisson, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Towards practical argumentation in multi-agent systems," in *2015 Brazilian Conference on Intelligent Systems, BRACIS 2015*, 2015.

[16] P. McBurney and S. Parsons, "Locutions for argumentation in agent interaction protocols," in *Agent Communication*. Springer, 2005, pp. 209–225.

[17] F. Sadri, F. Toni, and P. Torroni, "Logic agents, dialogues and negotiation: An abductive approach," in *Proc. AISB'01*. AISB, 2001.

[18] P. McBurney, R. V. Eijk, S. Parsons, and L. Amgoud, "A dialogue-game protocol for agent purchase negotiations," 2001.

[19] P. McBurney and S. Parsons, "Games that agents play: A formal framework for dialogues between autonomous agents," *Journal of Logic, Language and Information*, vol. 11, p. 2002, 2001.

[20] ——, "Dialogue games in multi-agent systems," *Informal Logic*, vol. 22, p. 2002, 2002.

[21] J. Bentahar, R. Alam, and Z. Maamar, "An argumentation-based protocol for conflict resolution," in *Workshop on Knowledge Representation for Agents and MultiAgent Systems (KRAMAS)*, 2008.

[22] L. Amgoud and S. Vesic, "A formal analysis of the role of argumentation in negotiation dialogues," *J. Log. and Comput.*, vol. 22, no. 5, pp. 957–978, oct 2012.

[23] Y. Dimopoulos, P. Moraitis, and L. Amgoud, "Characterizing the outcomes of argumentation-based integrative negotiation," *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, vol. 2, pp. 456–460, 2008.

[24] A. R. Panisson, A. Freitas, D. Schmidt, L. Hilgert, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Arguing About Task Reallocation Using Ontological Information in Multi-Agent Systems," in *12th International Workshop on Argumentation in Multiagent Systems*, 2015.

[25] A. S. Rao, "Agentspeak (l): Bdi agents speak out in a logical computable language," in *Agents Breaking Away*. Springer, 1996, pp. 42–55.

[26] J. Mayfield, Y. Labrou, and T. Finin, "Evaluation of kqml as an agent communication language," in *Intelligent Agents II Agent Theories, Architectures, and Languages*. Springer, 1996, pp. 347–360.

[27] A. Kakas and P. Moraitis, "Adaptive agent negotiation via argumentation," in *5th Int. Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06, 2006, pp. 384–391.

[28] A. C. Kakas and P. Moraitis, "Argumentative agent deliberation, roles and context." *Electr. Notes Theor. Comput. Sci.*, vol. 70, no. 5, pp. 39–53, 2002.

[29] A. Kakas and P. Moraitis, "Argumentation based decision making for autonomous agents," in *2th Int. Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '03, 2003, pp. 883–890.

[30] S. Parsons and E. Sklar, "How agents alter their beliefs after an argumentation-based dialogue," in *Argumentation in Multi-Agent Systems*. Springer, 2006, pp. 297–312.

[31] S. V. Rueda and M. V. Martínez, "Interaction among bdi argumentative agents: a dialogue games approach," in *XI Congreso Argentino de Ciencias de la Computación*, 2005.

[32] A. Hussain and F. Toni, "On the benefits of argumentation for negotiation-preliminary version," in *Proc. 6th European workshop on multi-agent systems (EUMAS-2008)*, 2008.

[33] E. Oliva, P. McBurney, A. Omicini, and M. Viroli, "Argumentation and artifacts for negotiation support," *International Journal of Artificial Intelligence*, vol. 4, no. S10, pp. 90–117, 2010.

[34] I. Rahwan and K. Larson, "Mechanism design for abstract argumentation," in *Proc. 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008, pp. 1031–1038.

[35] ——, "Pareto optimality in abstract argumentation." in *AAAI*, D. Fox and C. P. Gomes, Eds. AAAI Press, 2008, pp. 150–155.

[36] S. Pan, K. Larson, and I. Rahwan, "Argumentation mechanism design for preferred semantics." in *COMMA*. Citeseer, 2010, pp. 403–414.