Towards Practical Argumentation in Multi-Agent Systems

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

***Abstract*—Argumentation is a key technique for reaching agreements in multi-agent systems. However, there are few practical approaches to develop multi-agent systems where agents engage in argumentation-based dialogues. In this paper, we give formal semantics to speech acts for argumentation-based dialogues in the context of an agent-oriented programming language. Our approach uses operational semantics and builds upon existing work that provides computationally grounded semantics for agent mental attitudes such as beliefs and goals. The paper also shows how our formal semantics can be used to prove properties of argumentation in multi-agent systems with direct reference to mental attitudes. We do so with an example of a proof sketch of termination of multi-agent dialogues under certain assumptions.**

I. Introduction

Communication is one of the key issues in building multi-agent systems, where the agents need to communicate in order to resolve differences of opinion or conflicts of interest [1], to work coordinately [2], to resolve dilemmas, and to reach agreements [3]. Many of these communication requirements cannot be fulfilled by exchange of single messages. They require the exchange of sequences of messages upon related statements. Therefore, agents need the ability to engage in multi-agent dialogues [4].

In this paper, we extend the performatives normally available in implementations of the AgentSpeak agent-oriented programming language, such as ***Jason*** [5], as well as other agent programming languages, to enable argumentation-based dialogues between agents. We use a selection of performatives widely used in the argumentation-based dialogue literature, and we give operational semantics for them. In particular, we also define the semantics of a multi-agent system, i.e., at the social level, extending the work presented in [6], where operational semantics was given to basic speech acts, for which it sufficed to show how the mental attitudes of an individual agent were altered when a message with a particular speech act was received, whereas in this work both sending and receiving dialogue statements alters the social state of the argumentative system. Also, because we need to address the social perspective of a dialogue based on argumentation, we have adapted the operational semantics to include two separate transition systems, at the individual and social levels, and how one affects the other. Initial ideas towards this approach were published in [7].

The contribution of the paper is twofold. First, we define the formal semantics of speech acts for argumentation-based dialogues building on a computationally grounded semantics for agent mental attitudes [8]. Those mental attitudes are directly involved in the semantics of the argumentation speech acts. This means that if a performative refers to agent beliefs or intentions, this can be concretely realised in a computational system (providing a more principled way to implement real-world agent systems). As a consequence, we also make the semantics more detailed, covering the relation to individual and social changes in the system states, although perhaps slightly more complex to specify. Second, we demonstrate that our semantics can be used to prove properties of dialogue protocols or argumentation systems where the specific mental states of the participating agents play a part. This is possible because the proofs makes direct use of the inference rules that define the transition relation of the operational semantics, which make fully formal both the social aspects as well as the mental attitudes of individual agents.

In this paper, we assume the reader is familiar with the theory of speech acts [9], argumentation system [10], argumentation-based dialogue [11], [12], [4], and most importantly with the AgentSpeak language (the language that we use as example, and arguably the best know agent language in the relevant literature) and its semantics. A textbook explanation of AgentSpeak can be found in [5].

The remainder of this paper is organised as follows. In the next section, we discuss briefly some related work. Then we introduce the operational semantics of a selection of performatives for argumentation-based dialogues, the main aim of this paper. Next, we provide an example of a proof using the semantics showing that, under certain assumptions, an argumentation-based dialogue between agents complying with the provided semantics for the performatives always terminates. In the final section of this paper, we make some final remarks and discuss possible directions for future work.

II. Related Work

Much work on operational semantics for agent-oriented programming languages can be found in the literature, among which is [6], defining operational semantics for speech-act based communication, which serves as the basis for our work. In that paper, semantics is given for basic performatives that allow the communication between agents through simple message exchanges. We follow and extend that work with new performatives to allow argumentation-based dialogues, which also requires the exchange of sequences of interactions.

The performatives used in this paper can be found in the literature of argumentation-based dialogues. Work such as [11], [12], [4] use some combination of these performatives to support argumentation through dialogues, as well as the work in [13] that extends such performative set to support argumentation-based negotiation.

978-1-5090-0016-6/15 $31.00 © 2015 IEEE
DOI 10.1109/BRACIS.2015.30

These performatives can also be found in the work by McBurney and Parsons [14], which proposes some performatives, that they consider necessary for argumentation, to be added to FIPA ACL. In that work, they also give axiomatic and operational semantics to a protocol called *Fatio*. Our work has a similar objective, but we give semantics for the performatives in the context of an agent-oriented programming language and not for a particular protocol.

More recently, the work reported in [15] proposed the use of some performatives for argumentation in AgentSpeak and attempted to give semantics to those performatives. Unlike what we present here, the work in [15], as well as [13], is focused on negotiation and uses an electronic trading scenario. Also, the work [15] is similar to [6] in treating the communication of a single message exchange and not as a sequence of interactions (i.e., a dialogue) as in our work.

III. Formal Semantics

A. New Performatives for AgentSpeak

The performatives selected to enable argumentation-based dialogues in AgentSpeak are presented below, along with the intended (informal) meaning:

- `assert`: an agent that sends an `assert` message 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.
- `accept`: the sender declares, to all participants of the dialogue, that it accepts a previous claim of another agent. The receivers of the message become aware of this acceptance.
- `retract`: the agent declares, to all participants of the dialogue, that it is no longer committed to defending its previous claim. The receivers of the message become aware of this fact.
- `question`: the sender desires to know the reasons for a previous claim from another agent. The receiver of the message is committed to defending its claim, so presumably it will provide the support set for its claim.
- `challenge`: the `challenge` performative is similar to `question`, except that the sender of the message is committed to defending a claim contrary to the previous claim of another agent.

Further, performatives `opendialogue` and `closedialogue` are used for creating and concluding dialogues, respectively; `justify` is used to respond to a `question` or `challenge` message; and two other performatives, `acceptdialogue` and `refusedialogue`, are used by the participants to accept or refuse taking part in a dialogue, respectively.

B. The Basis for the Operational Semantics

We define the semantics of speech acts for argumentation-based dialogues in AgentSpeak using operational semantics, a widely used method for giving semantics to programming languages [16]. The operational semantics is given by a set of inference rules that define a transition relation between configurations $\langle AG, D\rangle$ of the multi-agent system[1] where:

- The AG component is a set of tuples $\langle id, Conf\rangle$ representing each agent in the society, where each agent is identified by a unique identifier id and the agent current internal state is represented by $Conf$. The agent state is in fact given by a *configuration* of the operational semantics of AgentSpeak as formalised in the existing literature (e.g., [6]); we assume some familiarity with the semantics of AgentSpeak.
- The set of all dialogues in that society, D, is a set of tuples $\langle did, Ags, Status\rangle$ where:
  - did is a dialogue identifier (which is unique for each dialogue within that multi-agent system);
  - Ags is a set of tuples $\langle id, CS\rangle$, where id identifies a particular agent that is participating in the dialogue and CS is its *commitment store*;
  - $Status$ represents the status of the dialogue and for the time being we assume it is one of only two values: `OPEN` if the dialogue is ongoing and `CLOSED` otherwise.

The agent configuration ($Conf$) is given by a tuple $\langle ag, C, M, T, s\rangle$, originally defined in [6], where:

- ag is a set of beliefs bs and a set of plans ps.
- An agent's circumstance C is a tuple $\langle I, E, A\rangle$ where:
  - I is a set of *intentions* $\{i, i', \ldots\}$. Each intention i is a stack of partially instantiated plans.
  - E is a set of *events* $\{(te, i), (te', i'), \ldots\}$. Each event is a pair (te, i), where te is a triggering event and i is an intention — a stack of plans in case of an internal event, or the empty intention $\top$ in case of an external event. For example, when the belief revision function (which is not part of the AgentSpeak interpreter but rather of the agent's overall architecture), updates the belief base, the associated events — i.e., additions and deletions of beliefs — are included in this set. These are called *external* events; internal events are generated by additions or deletions of goals from plans currently executing.
  - A is a set of *actions* to be performed in the environment.
- M is a tuple $\langle In, Out, SI\rangle$ whose components characterise the following aspects of communicating agents (note that communication is typically asynchronous):
  - In is the mail inbox: the multi-agent system runtime infrastructure includes all messages addressed to this agent in this set. Elements of this set have the form $\langle mid, id, ilf, cnt\rangle$, where mid is a message identifier, id identifies the sender of the message, ilf is the illocutionary force of the message, and cnt its content: a (possibly singleton) set of AgentSpeak predicates or plans, depending on the illocutionary force of the message.
  - Out is where the agent posts messages it wishes to send; it is assumed that some underlying communication infrastructure handles the delivery of such messages. Messages in this set have exactly the same format as above, except that here id refers to the agent to which the message is to be sent.
  - SI is used to keep track of intentions that were suspended due to the processing of communication messages; the intuition is as follows: intentions associated with illocutionary forces that require a reply from the interlocutor are suspended, and they are only resumed when such reply has been received.
- When giving semantics to an AgentSpeak agent's reasoning cycle, it is useful to have a structure which keeps track of temporary information that may be subsequently required

[1] We use only components that are needed to demonstrate the semantics, but we emphasise the existence of other components such as roles, norms, etc.

within a reasoning cycle. T is a tuple $\langle R, Ap, \iota, \varepsilon, \rho\rangle$ with such temporary information; these components are as follows:

- R is the set of *relevant plans* (for the event being handled).
- Ap is the set of *applicable plans* (the relevant plans whose contexts are true).
- ι, ε, and ρ record a particular intention, event, and applicable plan (respectively) being considered along the execution of one reasoning cycle.

- The current step within an agent's reasoning cycle is symbolically annotated by $s \in \{\mathsf{ProcMsg}, \mathsf{SelEv}, \mathsf{RelPl}, \mathsf{ApplPl}, \mathsf{SelAppl}, \mathsf{AddIM}, \mathsf{SelInt}, \mathsf{ExecInt}, \mathsf{ClrInt}\}$. These labels stand for, respectively: processing a message from the agent's mail inbox, selecting an event from the set of events, retrieving all relevant plans, checking which of those are applicable, selecting one particular applicable plan (the intended means), adding the new intended means to the set of intentions, selecting an intention, executing the selected intention, and clearing an intention or intended means that may have finished in the previous step.
- The semantics of AgentSpeak makes use of "selection functions" which allow for user-defined components of the agent architecture. We use here only the S_M functions, as originally defined in [6]; the *select message* function is used to select one message from an agent's mail inbox.

In the interests of readability, we adopt the following notational conventions in our semantics rules:

- If C is an AgentSpeak agent circumstance, we write C_E to make reference to the E component of C, and similarly for other components of the multi-agent system and of the configuration of each agent.
- We write AG^{id} to identify the agent represented by that id in the set of agents AG. We use this whenever the component corresponds to a set of tuples $\langle id, \ldots\rangle$. Also, if AG is a set of tuples $\langle id, Conf\rangle$, then we refer to a configuration ($Conf$) of one agent (identified by id) in AG by AG^{id}_{Conf}.
- We write $b[d(did), s(id)]$ to identify the origin of a belief related to a dialogue, where did is a dialogue identifier, and id an agent identifier (d refers to $dialogue$ and s refers to $source$). Whenever an agent makes a statement related to a dialogue, the dialogue identifier did is added as an annotation.
- We use two transitions to represent the state change of the multi-agent system, where the transition $\longrightarrow_{AS}$ (transition of the configuration of an individual agent) is part of the transition $\longrightarrow_{DS}$ (the transition of the multi-agent system). So each transition in the agent configuration also causes a transition in the multi-agent system configuration exactly in the component AG^{aid}_{Conf}, where aid refers to the identifier of the agent which went through the transition.

Also, we use aid to refer to the agent that is executing an internal action of interest or receiving a message. Finally, we make use of a function called CTJ (where CTJ stands for "care to justify") that returns TRUE if the agent wishes to justify its previous assertion (this depends on the agent's reasoning and makes reference to agents' autonomy).

Due to space restrictions, we left out of the semantics presented in this paper two internal actions used by agents to create a dialogue and to close a dialogue respectively and, further, four semantic rules for receiving messages related to creating and closing the dialogue. The internal action for creating a dialogue adds a new dialogue in the multi-agent system, sends messages to all agents with the performative `opendialogue`, inviting them to the new dialogue, and waits for all agents to reply. The internal action close dialogue simply sends, to all agents in the dialogue to be closed, a message warning them that the dialogue is being closed (using the performative `closedialogue`). The four semantic rules lefts out were for receiving the following messages:

- `opendialogue`: adds a belief about the dialogue to be started with the source of the agent that proposed the opening of the dialogue (i.e., the sender of the message); The agents that receive the message should react to the event generated, deciding whether to accept or not to participate in the dialogue, and presumably responding with the appropriate message (`acceptdialogue` or `refusedialogue`).
- `closedialogue`: removes the belief about the ongoing dialogue. The dialogue can be closed only by the same agent (identified by id) which previously opened the dialogue.
- `acceptdialogue`: the sender of the message is removed from the set of agents that are expected to respond. If all agents respond, the intention can be resumed, otherwise the intention remains suspended.
- `refusedialogue`: the sender of the message is removed from the set of agents that are expected to respond, as well as from the set of agents participating in the dialogue. If all agents have replied, the intention can be resumed.

C. Semantic Rules for Sending the New Performatives

In this section, we give semantics for sending the new performatives which allow argumentation-based dialogues, showing how it affects the state of the agent and the state of the dialogue.

(EXECACTSNDASSERT)

$$\frac{T_\iota = i[head \leftarrow \texttt{.send}(did, \texttt{assert}, p); h] \quad p \notin CS \quad \langle aid, CS\rangle \in D^{did}_{Ags}}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}}$$

where:

(a)	D'^{did}_{Ags}	=	$(D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\}$ with $CS' = CS \cup \{p\}$
	AG'^{aid}_{Conf}	=	the transition given by (b)
(b)	M'_{Out}	=	$M_{Out} \cup \{\langle mid, id, \texttt{assert}, p[d(did)]\rangle\}$ for each $Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\})$
	C'_I	=	$(C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$

Internal Action `.send` with `assert`: The action `.send` with performative `assert` updates the CS of the agent that performs the action and sends, to all agents in the dialogue, a message stating that the sender is willing to defend this claim.

The agent can use *assertion attitudes* as defined in [12], [4], but in any case the agent can only assert a formula it did not previously assert; that is, an agent cannot assert again formalæ that are already in its CS.

Another important point to be noticed is that an assertion is always made to a particular dialogue (identified by did) and not to a specific agent; this is because the agents will introduce new claims to be defended to all agents participating in the dialogue and not to an individual agent.

(EXECACTSNDACCEPT)

$$\frac{T_\iota = i[head \leftarrow \texttt{.send}(tid, \texttt{accept}, p[d(did)]);h]}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}}$$

where:

(a)	D'^{did}_{Ags}	$=$	$(D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\}$ with $CS' = CS \cup \{p\}$
	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{Out}	$=$	$M_{Out} \cup \{\langle mid, id, \texttt{accept}, p[d(did)]\rangle\}$ for each $Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\})$
	C'_I	$=$	$(C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$

Internal Action `.send` **with** `accept`: The action `.send` with performative `accept` updates the CS of the agent that performs the action and sends, to all agents in the dialogue, a message stating that the agent accepts the claim made by another agent identified by tid. Note that p (the formula that was accepted) has the annotation $[d(did)]$, this means that p has been previously asserted in that dialogue (identified by did). In other words, an agent can only accept a claim made by another agent in that same dialogue.

(EXECACTSNDRETRACT)

$$\frac{T_\iota = i[head \leftarrow \texttt{.send}(did, \texttt{retract}, p[d(did)]);h]}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}}$$

where:

(a)	D'^{did}_{Ags}	$=$	$(D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\}$ with $CS' = CS \setminus \{p\}$
	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{Out}	$=$	$M_{Out} \cup \{\langle mid, id, \texttt{retract}, p[d(did)]\rangle\}$ for each $Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\})$
	C'_I	$=$	$(C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$

Internal Action `.send` **with** `retract`: The agent performs this internal action to retract a previous claim that the agent itself asserted in that dialogue. The agent's CS is updated with the removal of the given formula. A message is sent to each agent in the dialogue informing the decision of that agent to retract its previous claim.

(EXECACTSNDQUESTION)

$$\frac{T_\iota = i[head \leftarrow \texttt{.send}(id, \texttt{question}, p[d(did)]);h]}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}}$$

where:

(a)	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{Out}	$=$	$M_{Out} \cup \{\langle mid, id, \texttt{question}, p[d(did)]\rangle\}$
	C'_I	$=$	$(C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$

Internal Action `.send` **with** `question`: The action `.send` with performative `question` is used when an agent wants to question another agent about an assertion it has previously made. This message is sent only to the agent that previously made the assertion.

(EXECACTSNDCHALLENGE)

$$\frac{T_\iota = i[head \leftarrow \texttt{.send}(tid, \texttt{challenge}, p[d(did)]);h]}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}}$$

where:

(a)	D'^{did}_{Ags}	$=$	$(D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\}$ with $CS' = CS \cup \{\neg p\}$
	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{Out}	$=$	$(M_{Out} \cup \{\langle mid, tid, \texttt{challenge}, p[d(did)]\rangle\}) \cup \{\langle mid, id, \texttt{assert}, \neg p[d(did)]\rangle\}$ for each $Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid} \cup Ags^{tid}\})$
	C'_I	$=$	$(C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$

Internal Action `.send` **with** `challenge`: The action `.send` with the performative `challenge` is performed when an agent wants to challenge another agent about an assertion it previously made. Differently from the question performative, when an agent makes a challenge move it is willing to defend a claim contrary to the claim of the other agent.

The message with a performative `challenge` is sent only to the agent that made the previous claim. As the agent is willing to defend its claim, messages are sent to all other agents in the dialogue with the respective `assert` messages.

D. Semantic Rules for Receiving the New Performatives

In this section we give semantics for receiving the new performatives that allow argumentation-based dialogues, showing how they affect the state of the agent and the state of the dialogue.

(ASSERT)

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{assert}, p[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \mathsf{ExecInt}\rangle\end{array}}$$

where:

(a)	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{In}	$=$	$M_{In} \setminus \{\langle mid, sid, \texttt{assert}, p[d(did)]\rangle\}$
	ag'_{bs}	$=$	$ag_{bs} + p[d(did), s(sid)]$
	C'_E	$=$	$C_E \cup \{\langle +p[d(did), s(sid)], \mathsf{T}\rangle\}$

Receiving an `assert` **Message:** The claim asserted in the dialogue is added to the belief base of the receiver with an annotation of the dialogue identifier $d(did)$ and the identifier of the agent that asserted the claim as the source of that information $s(sid)$. The agent that received the message can react to this claim because of the event generated by the belief addition, as usual in AgentSpeak. Whether an agent accepts or not the claim made by another agent depends on its *acceptance attitude* as described in [12], [4], which depends on if the agent has or not an acceptable argument to or against the claim.

(ACCEPT)

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{accept}, p[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \mathsf{ExecInt}\rangle\end{array}}$$

where:

(a)	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{In}	$=$	$M_{In} \setminus \{\langle mid, sid, \texttt{accept}, p[d(did)]\rangle\}$
	ag'_{bs}	$=$	$ag_{bs} + p[d(did), s(sid)]$
	C'_E	$=$	$C_E \cup \{\langle +p[d(did), s(sid)], \mathsf{T}\rangle\}$

Receiving an `accept` **Message:** This message means an agent (identified by sid) accepts a claim previously made, as part of this dialogue, by another agent. The receiver of the message become aware of this acceptance.

(RETRACT)

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{retract}, p[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \mathsf{ExecInt}\rangle\end{array}}$$

where:

(a)	AG'^{aid}_{Conf}	$=$	the transition given by (b)
(b)	M'_{In}	$=$	$M_{In} \setminus \{\langle mid, sid, \texttt{retract}, p[d(did)]\rangle\}$
	ag'_{bs}	$=$	$ag_{bs} - p[d(did), s(sid)]$
	C'_E	$=$	$C_E \cup \{\langle -p[d(did), s(sid)], \mathsf{T}\rangle\}$

Receiving a `retract` **Message:** This message means an agent, identified by sid, is withdrawing its earlier assertion. The formula is removed from belief base of the receiver of the message, with the appropriate source and dialogue annotation.

(QUESTION)

$$S_M(M_{In}) = \langle mid, sid, \texttt{question}, p[d(did)]\rangle \quad \mathsf{CTJ}(p) = \texttt{TRUE}$$

$$(a)\quad \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle$$
$$(b)\quad \langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag, C, M', T, \mathsf{ExecInt}\rangle$$

where:

(a)	D'^{did}_{Ags}	=	$(D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\}$ with $CS' = CS \cup \{Sp\}$
	AG'^{aid}_{Conf}	=	the transition given by (b)
(b)	M'_{In}	=	$M_{In} \setminus \{\langle mid, sid, \texttt{question}, p[d(did)]\rangle\}$
	M'_{Out}	=	$M_{Out} \cup \{\langle mid, id, \texttt{justify}, Sp[d(did)]\rangle\}$ for each $Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\})$ where $Sp \models p$ *and* $Sp \in ag_{bs}$

Receiving a `question` Message: If the agent can or wants to reply, in keeping with agent autonomy (this is represented in the semantics through a CTJ function which is meant to be agent specific), then the agent's CS will be updated with the support of the previous claim p, and the support will be also sent to all other agents in the dialogue.

(JUSTIFY)

$$S_M(M_{In}) = \langle mid, sid, \texttt{justify}, Sp[d(did)]\rangle$$

$$(a)\quad \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle$$
$$(b)\quad \langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \mathsf{ExecInt}\rangle$$

where:

(a)	AG'^{aid}_{Conf}	=	the transition given by (b)
(b)	M'_{In}	=	$M_{In} \setminus \{\langle mid, sid, \texttt{justify}, Sp[d(did)]\rangle\}$
	and for each $p \in Sp$:		
	ag'_{bs}	=	$ag_{bs} + p[d(did), s(sid)]$
	C'_E	=	$C_E \cup \{\langle +p[d(did), s(sid)], \mathsf{T}\rangle\}$

Receiving a `justify` Message: This is similar to the assert performative, except for the fact that the content of the message is a set of formulæ that justify the previous claim of the sender of the message (identified by sid).

(CHALLENGE)

$$S_M(M_{In}) = \langle mid, sid, \texttt{challenge}, p[d(did)]\rangle \quad \mathsf{CTJ}(p) = \texttt{TRUE}$$

$$(a)\quad \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle$$
$$(b)\quad \langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C, M', T, \mathsf{ExecInt}\rangle$$

where:

(a)	D'^{did}_{Ags}	=	$(D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\}$ with $CS' = CS \cup \{Sp\}$;
	AG'^{aid}_{Conf}	=	the transition given by (b)
(b)	M'_{In}	=	$M_{In} \setminus \{\langle mid, sid, \texttt{challenge}, p[d(did)]\rangle\}$
	M'_{Out}	=	$M_{Out} \cup \{\langle mid, id, \texttt{justify}, Sp[d(did)]\rangle\}$ for each $Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\})$ where $Sp \models p$ *and* $Sp \in ag_{bs}$
	ag'_{bs}	=	$ag_{bs} + \neg p[d(did), s(sid)]$

Receiving a `challenge` Message: If the agent can or wants to reply (we assume the CTJ function determines whether that is the case or not), the agent's CS is updated with the support of the previous claim p and the support is also sent to all other agents in the dialogue. In addition to the question performative, this rule adds that the agent identified by sid (i.e., the sender of the message) is willing to defend the claim contrary to the previous claim that is being challenged.

IV. A SAMPLE PROOF

We now show that, using our semantics and under certain assumptions described below, dialogues among agents endowed with defeasible reasoning (as in [17] and [18]) following a basic protocol (also defined below) always *terminate*, either in agreement or disagreement. The main purpose of this proof is to exemplify the use of the operational semantics in proving properties of multi-agent systems that make use of the argumentation-based dialogues, which we have developed, in ***Jason***, or indeed any other implementation of the semantics introduced in this paper.

Assumptions: (a) the individual knowledge bases of the agents do not change due to external sources in the course of the dialogue (perception from environment, etc.); (b) the agents only assert formulæ that they believe to be true and that can be derived from their knowledge bases (veracity); and (c) agents always retract related assertions when newly accepted assertions change a conclusion they previously asserted (i.e., the former conclusion is no longer true for that agent).

Definition 1 (Basic Dialogue Protocol): A dialogue starts with an agent asserting a particular formula. Agents always position themselves with respect to assertions made by other agents, either accepting it or asserting its negation (depending on whether they consistently derive the formula or its negation from their knowledge bases together with the CS of all participants of the dialogue), always provide justification when questioned (i.e., asserting the support for their previous assertions), always accept formulæ asserted by other agents in the dialogue that they could not previously derive (i.e., agents trust each other and learn from them) provided it does not make their knowledge bases inconsistent, and always provide the support of their previous assertions if a justification given by another agent cannot be challenged (the agent cannot derive the negation of any formulæ in the justification) and the newly accepted formulæ from the received justification still do not change its previous conclusion (i.e., allow the agent to derive the complement of its previous assertion). The dialogue terminates if all participating agents choose not to send any further messages as part of that dialogue.

Proposition 1 (Termination): Given the assumptions above, consider a set of n agents participating in a dialogue, each capable of defeasible reasoning, and each participant following the *basic dialogue protocol* defined above. Let $\Delta = \{\Pi_1 \cup \ldots \cup \Pi_n\}$ with each Π_i, $1 \leq i \leq n$, be the knowledge base of agent i participating in the given dialogue. If $\Delta \models_{d,s} \psi$ (ψ is strictly or defeasibly derived from Δ), then eventually each Π_n will also individually derive ψ — $\Pi_n \models_{d,s} \psi$ — and the dialogue terminates in agreement on the initial assertion ψ. If $\Delta \models_{d,s} \bot$ (i.e., a contradiction) then the dialogue also eventually terminates, but it will terminate in disagreement.

Proof Sketch: There exists three possible derivations given Δ (the union of all individual knowledge bases) and an initial assertion ψ: $\Delta \models_{d,s} \bot$, $\Delta \models_{d,s} \psi$, and $\Delta \models_{d,s} \neg\psi$.

If an agent asserts (EXECACTSNDASSERT) ψ and Δ derives its complement, at least one other participating agent will not accept (the agent(s) that contributed to Δ the formulæ that allowed the derivation or the complement of ψ) and therefore assert the complement of ψ, by reacting to a belief addition through rule ASSERT, and sending an `assert` message with $\neg\psi$ through rule EXECACTSNDASSERT. When questioned (at least the dialogue initiator will react through ASSERT and execute EXECACTSNDQUESTION) the agent will provide the justification (QUESTION) as required by the assumed protocol. The agent receiving this new information (JUSTIFY) either accepts or makes an assertion against the justification or against some formula in the justification (in this case the agent must be

able to derive this assertion from its knowledge base, following the defined protocol). If the agent does not accept, the dialogue continues in the same way as in the previous steps, where new information will be exchanged (the missing information for the agent to accept the derivation from Δ). If the agent accepts this new information (EXECACTSNDACCEPT), thereby retracting its previous assertion — given that the union of all CSs does not derive a contradiction — that agent can now also derive $\neg\psi$. As the beliefs do not change due to external sources (assumed) and the belief bases are finite, the dialogue eventually terminates in agreement.

If an agent asserts (EXECACTSNDASSERT) ψ when $\Delta \models_{d,s} \psi$, either all agents in the dialogue accept the assertion (reacting the ASSERT with EXECACTSNDACCEPT) and the dialogue ends, or at least one agent sends an assert message against the previous assertion (reacting to the ASSERT with EXECACTSNDASSERT). In the latter case, the agent will eventually provide the justification for its previous assertion and the dialogue procedes as in the case above. The dialogue ends with all agents agreeing on the initial assertion.

If $\Delta \models_{d,s} \bot$, at least one agent will derive a contradiction, even with the new information received through rule JUSTIFY, but as the belief bases are *finite* (and assumed to be unchanging during the dialogue), the dialogue will eventually terminate in disagreement. ■

V. CONCLUSION

In this paper, we gave formal semantics for a set of performatives that enable argumentation-based dialogues in agent-oriented programming language. We built our semantics on top of the operational semantics of AgentSpeak and we implemented it extending the ***Jason*** platform, but the semantics can be used for other agent languages, which is facilitated by the fact that most agent languages are formalised using operational semantics as well. The semantics formalises the combination of state changes at the individual and social levels and is given for both receiving and sending messages.

As future work, we intend to develop applications using the ***Jason*** implementation of the semantics presented in this paper and show that the framework for argumentation-based dialogues among rational agents proposed here is effective. As a matter of fact, the framework has already been successfully used in an healthcare prototype application briefly reported in [19]. Other directions for our work are: (*i*) to use planning techniques in multi-agent dialogues, where the semantics presented here could serve as the basis to support the creation of planning domain models. We took initial steps towards this direction in [20], where we created a centralised model for planning argumentation-based agent interactions; and (*ii*) we intend to define more complex protocols using the formalisation presented here, extending our work in [21].

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] X. Fan, F. Toni, and A. Hussain, "Two-agent conflict resolution with assumption-based argumentation." in *COMMA*, 2010, pp. 231–242.

[2] P. Pardo, S. Pajares, E. Onaindia, L. Godo, and P. Dellunde, "Multiagent argumentation for cooperative planning in delp-pop," in *10th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2011, pp. 971–978.

[3] N. R. Jennings, S. Parsons, P. Noriega, and C. Sierra, "On argumentation-based negotiation," in *Int. Workshop on Multi-Agent Systems*, 1998.

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

[5] 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.

[6] R. Vieira, Á. Moreira, M. Wooldridge, and R. H. Bordini, "On the formal semantics of speech-act based communication in an agent-oriented programming language," *J. Artif. Int. Res.*, vol. 29, no. 1, pp. 221–267, Jun. 2007.

[7] A. R. Panisson, F. Meneguzzi, M. Fagundes, R. Vieira, and R. H. Bordini, "Formal semantics of speech acts for argumentative dialogues," in *Thirteenth Int. Conf. on Autonomous Agents and Multiagent Systems*, 2014, pp. 1437–1438.

[8] R. H. Bordini and Á. F. Moreira, "Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in AgentSpeak(L)," *Annals of Mathematics and Artificial Intelligence*, vol. 42, no. 1-3, pp. 197–226, 2004.

[9] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[10] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*. Cambridge University Press, 2008.

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

[12] S. Parsons and P. McBurney, "Argumentation-based dialogues for agent coordination. group decision and negotiation," *Group Decision and Negotiation*, 2004.

[13] L. Amgoud, S. Parsons, and N. Maudet, "Arguments, dialogue, and negotiation," *Journal of Artificial Intelligence Research*, vol. 23, p. 2005, 2000.

[14] P. McBurney and S. Parsons, "Locutions for argumentation in agent interaction protocols." in *AC*, ser. Lecture Notes in Computer Science, R. M. van Eijk, M.-P. Huget, and F. Dignum, Eds., vol. 3396. Springer, 2004, pp. 209–225.

[15] P. Bedi and P. Vashisth, "Extending speech-act based communication to enable argumentation in cognitive agents." in *Advances in computing, communication and control, ICAC3 2011, Mumbai, India*. Berlin: Springer, 2011, pp. 25–40.

[16] G. D. Plotkin, "A structural approach to operational semantics," 1981.

[17] T. Berariu, "An argumentation framework for bdi agents," in *Intelligent Distributed Computing VII*, ser. Studies in Computational Intelligence, F. Zavoral, J. J. Jung, and C. Badica, Eds. Springer International Publishing, 2014, vol. 511, pp. 343–354.

[18] 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*, 2014.

[19] 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.

[20] A. R. Panisson, G. Farias, A. Freitas, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Planning Interactions for Agents in Argumentation-Based Negotiation," in *11th International Workshop on Argumentation in Multiagent Systems*, 2014.

[21] A. R. Panisson, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Towards practical argumentation-based dialogues in multi-agent systems," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2015.