Argumentation Agent Architecture - Design Principles

Alexandru Sorici
University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computers, Computer Science Department
Emails: alex.sorici@gmail.com

Abstract

The paper presents the design principles used for creating the architecture of an intelligent argumentation agent. The internal representation of arguments, the communication protocol and message structure (based on the AIF ontology) will be presented. Finally, conclusions and possible future extensions of the model are presented.

Keywords: Agent Architecture, Communication Protocol, AIF

1. Introduction

Analyzing communication between entities could not be done throughout history without giving argumentation a great deal of study. Argumentation can be defined as the process of putting together a set of statements aimed to strengthen or weaken a claimed expression.
Today the argumentation field has been extended to the computer science and artificial intelligence domain, especially with areas dealing with multi-agent communication. It has also played a major role in the decision support systems field. One key challenge is the design of means of communication between intelligent agents. Considerable research effort has been expended on the design of artificial languages for agent communications, such as DARPA's Knowledge Query and Manipulation Language (KQML) [1] and the Foundation for Intelligent Physical Agents' (now IEEE FIPA) Agent Communications Language (FIPA ACL) [2].

A variety of systems are now available for users to create and represent arguments such as AML Araucaria [3], TruthMapping [4], Reason!Able [5], Compendium[6], and others. But, while significant progress has been made in the development of systems like the latter ones and in understanding the theoretical properties of different argumentation logics and in specifying argumentation dialogues, there remain major barriers to the development and practical deployment of argumentation systems. These are caused both by the lack of a standard agent architecture and a lack of a shared, agreed notation or "interchange format" for argumentation and arguments.

The development of a standard agent-based argumentation platform would certainly allow for an easier development of related systems such as Negotiation Systems, Decision support systems etc. The current paper aims at making an important start towards the removal of the earlier mentioned barriers by presenting the conceptual architecture of an argumentation agent based on the JADE platform and by presenting an argumentation conversation protocol whose messages rely on the Argumentation Interchange Format (AIF) [7].

The rest of the paper is structured as follows. Section two will introduce related work that explains the structure of the AIF ontology and that describes the usage of dialogue games for modeling agent communication. Section 3 will present the class and protocol diagrams used by the current initial agent architecture. Finally, section 4 will give conclusions and list the future possible extensions of the model.

2. Related work

Central to the new agent design are its communication protocol and the representation format of both the internal argument network and of the exchanged protocol messages. This section introduces work related to the usage of argumentation dialogue games used to model agent communication and work related to a standard argument interchange format, the extended AIF ontology.

2.1. Argumentation Dialogue games

When building multi-agent systems, we take for granted the fact that the agents which make up the system will need to communicate. They need to communicate in order to resolve differences of opinion and conflicts of interest, work together to resolve dilemmas or find proofs, or simply to inform each other of pertinent facts. Many of these communicate requirements cannot be fulfilled by the exchange of single messages. Instead, the agents concerned need to be able to exchange a sequence of messages which all bear upon the same subject. In other words they need the ability to engage in dialogues.
An influential model of human dialogues is the typology of primary dialogue types of argumentation theorists Douglas Walton and Erik Krabbe [8]. This categorization is based upon the information the participants have at the commencement of a dialogue (of relevance to the topic of discussion), their individual goals for the dialogue, and the goals they share. **Information-Seeking Dialogues** are those where one participant seeks the answer to some question(s) from another participant, who is believed by the first to know the answer(s). In **Inquiry Dialogues** the participants collaborate to answer some question or questions whose answers are not known to any one participant. **Persuasion Dialogues** involve one participant seeking to persuade another to accept a proposition he or she does not currently endorse.

Researchers in multi-agent systems and in argumentation have articulated dialogue game protocols for many of the types in the Walton and Krabbe typology. For example, the two-party protocol of Amgoud, Maudet and Parsons [9], which is based on MacKenzie's philosophical dialogue game *DC* [10], supports persuasion, inquiry and information-seeking dialogues.
It is on this work that we base the proposed protocol for our Argumentation Agent.
A dialogue between two individuals is seen as a game in which each individual has objectives and a set of legal moves which can be used to obtain those objectives. The moves are illocutions, and the objectives are matters such as persuading the other player of the truth of a proposition.
In [9] a set of dialogue moves is defined. For each move, they give what they call rationality rules and dialogue rules. The rationality rules specify the preconditions for playing the move. The dialogue rules specify the moves the other player can make next, and so specify the *protocol* under which the dialogue takes place.

assert(p) where p is a propositional formula

- **Rationality:** the player uses its [1]AS to check if there is an acceptable argument for the fact p.
- **dialogue** the other player can respond with:
    - *accept(p)*
    - *assert(-p)*
    - *challenge(p)*

[1] Argumentation System

assert(S) where S is a set of formulas representing the support of an argument. Note that in DC, players can only assert one propositional formula.

- **rationality** the player uses the AS to check if the related argument is acceptable
- **dialogue** the other player can play
    - *accept(S)*
    - *assert(-p)*
    - *challenge(p) where* $p \in S$
    - Informally, this means that the player can accept the whole support or challenge/deny an element of the support

accept(p) p is a propositional formula

- **rationality** the player uses the AS to check if there is an acceptable argument for p.
- **dialogue** the other player can play any allowed moves.

accept(S) S is a set of propositional formulas

- **rationality** the player uses his AS to check if each element of S is supported by an acceptable argument.
- **dialogue** the other player can play any allowed move

challenge(p) where p is a propositional formula

- **rationality Ø**
- **dialogue** the other player can only *assert(S)* where *S* is an argument supporting *p*.

As we will see in section 3, the used protocol uses the same primitives as those presented above but with a slight modification to the dialogue rules.

2.2. Argument Interchange Format (AIF)

Besides specifying what the primitives of the communication protocol and what the dialogue rules are, our agent interaction model is not complete without defining the representation format for the exchanged messages.
As discussed in the introduction, one of the main concerns of the representation format is that it be a standard thus allowing for interoperability and easy development. Viewing from this perspective, the AIF [7] looks like the perfect candidate.

Iyad Rahwan and Chris Reed [11] give a good description of the core AIF and the extensions they propose to help it deal with inferential schemes as those described by Walton [12].

As presented in [11] the core AIF has two types of nodes: *information nodes* (or *I-nodes*) and *scheme nodes* (or *S-nodes*). Information nodes are used to represent *propositional* information contained in an argument, such as a claim, premise, data, etc. S-nodes capture the application of *schemes* (i.e. patterns of reasoning). Such schemes may be domain independent patterns of reasoning, which resemble rules of inference in deductive logics but broadened to include non-deductive inference.

The schemes themselves are classified into the types *rule of inference scheme*, *conflict scheme*, and *preference scheme*. The AIF thus provides an ontology for expressing schemes and instances of schemes, and constrains the latter to the domain of the former via the function uses.
The present ontology has three different types of scheme nodes: *rule of inference application nodes* (or *RA-nodes*), *preference application nodes* (or *PA-nodes*) and *conflict application nodes* (or *CA-nodes*).

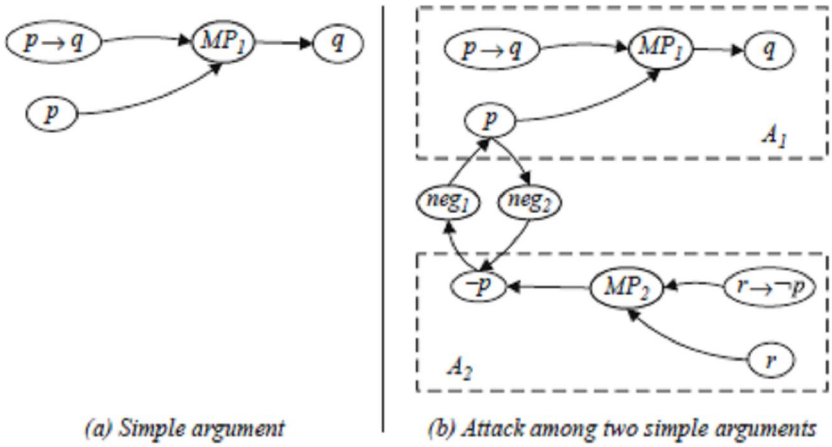

Fig. 1 Examples of simple arguments; S-Nodes denoted with a thicker border

A *simple* argument can be represented by linking premises to a conclusion like in figure 1.

The extensions to the core AIF [7] brought by Rahwan and Reed relate to representing argument schemes and were implemented in the ArgDF RDF code. They consider the set of schemes S as themselves nodes in the argument network and they introduce a new class of nodes, called *forms* (or *F-nodes*).
The following figure shows a clear example of the way the extensions work.

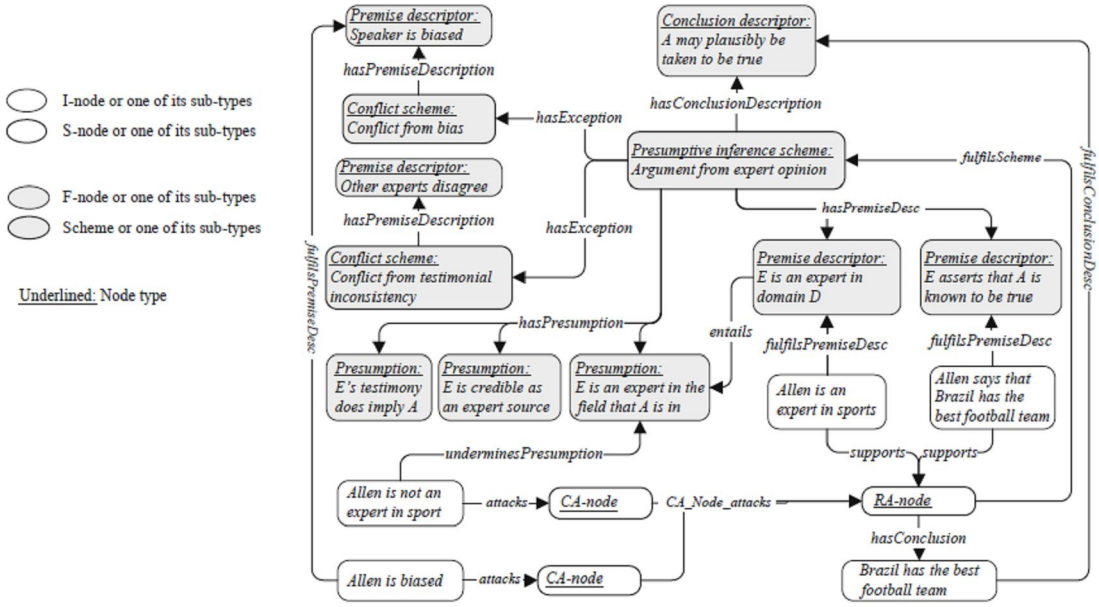

Fig. 2 An argument network showing an argument from expert opinion opinion, two counter-arguments undermining a presumption and an exception, and the descriptions of the schemes used by the argument and attackers. *A:* Brazil has the best football team: Allen is a sports expert and he says so; *B:* But Allen is biased, and he is not an expert in sports!

Even of more importance to the proposed AIF usage within the Argumentation Agent are the description logic (DL) characterizations of the components for each type of scheme that can be defined in the AIF. That is, another AIF-based ontology is presented which captures schemes as classes of arguments explicitly. The AIF model is reified by interpreting schemes as classes and S-nodes as instances of those classes; in this case, the semantics of the "*uses*" edge can be interpreted as "*instance – o f* ".
As described in [11], at the highest level, three concepts are identified: *statements* that can be made (that correspond to AIF I-nodes), *schemes* that describe arguments made up of statements (that correspond to AIF S-nodes) and *authors* of those statements and arguments (formerly just properties in AIF). All these concepts are disjoint. As with the ArgDF reification of AIF, different specializations of scheme are identified for example the rule scheme (which describes the class of arguments), conflict scheme, preference scheme.
Each of these schemes can then be further classified.
To capture the structural relationships between different schemes, their components should first be classified. This is done by classifying their premises, conclusions, assumptions and exceptions into different *classes of statements*. This is a very important step as we find, because it allows a very fine-tuned description of the scheme used. The individual statements making up the scheme can initially have text descriptors that convey the actual meaning, but our intended extensions of the model see them being replaced with structured statements from yet another ontology, one of the agents' domain ontologies. Such a construct is expected to allow the automatic creation and understanding of arguments by agents, given a topic o discussion and an inference engine. We will discuss more about the intended extensions in section 4.
With all the above in place, Rahwan and Reed provide an example of the Argument from Position to Know scheme described using the above mentioned reification of the AIF.

$$PositionToHaveKnowledgeStmnt \sqsubseteq DeclarativeStatement$$

formDescription : "E is in position to know whether A is true (false)"

$$KnowledgeAssertionStmnt \sqsubseteq DeclarativeStatement$$

formDescription : "E asserts that A is true(false)"

$$KnowledgePositionStmnt \sqsubseteq DeclarativeStatement$$

formDescription : "A may plausibly be taken to be true(false)"

$$LackOfReliabilityStmnt \sqsubseteq DeclarativeStatement$$

formDescription : "E is not a reliable source"

$$ArgFromPositionToKnow \equiv (PresumptiveArgument \sqcap \exists hasConclusion.KnowledgePositionStmnt \sqcap \exists hasPremise.PositionToHaveKnowledgeStmnt \sqcap \exists hasPremise.KnowledgeAssertionStmnt)$$

$$ArgFromPositionToKnow \sqsubseteq \exists hasException.LackOfReliabilityStmnt$$

Fig. 3 Example description of the Argument from Position to Know using the DL AIF reification by Rahwan and Reed

Having presented the influential work that lies at the basis of this paper we now go on to describing the proposed agent architecture and communication protocol.

3. Agent Architecture and Protocol Description

This section presents the argumentation agent architecture by presenting a class diagram for the JADE agent implementation and a sequence diagram for the intended protocol and then explaining them in detail.

3.1. JADE Argumentation Agent Architecture

We begin by showing the class diagram of the proposed architecture.

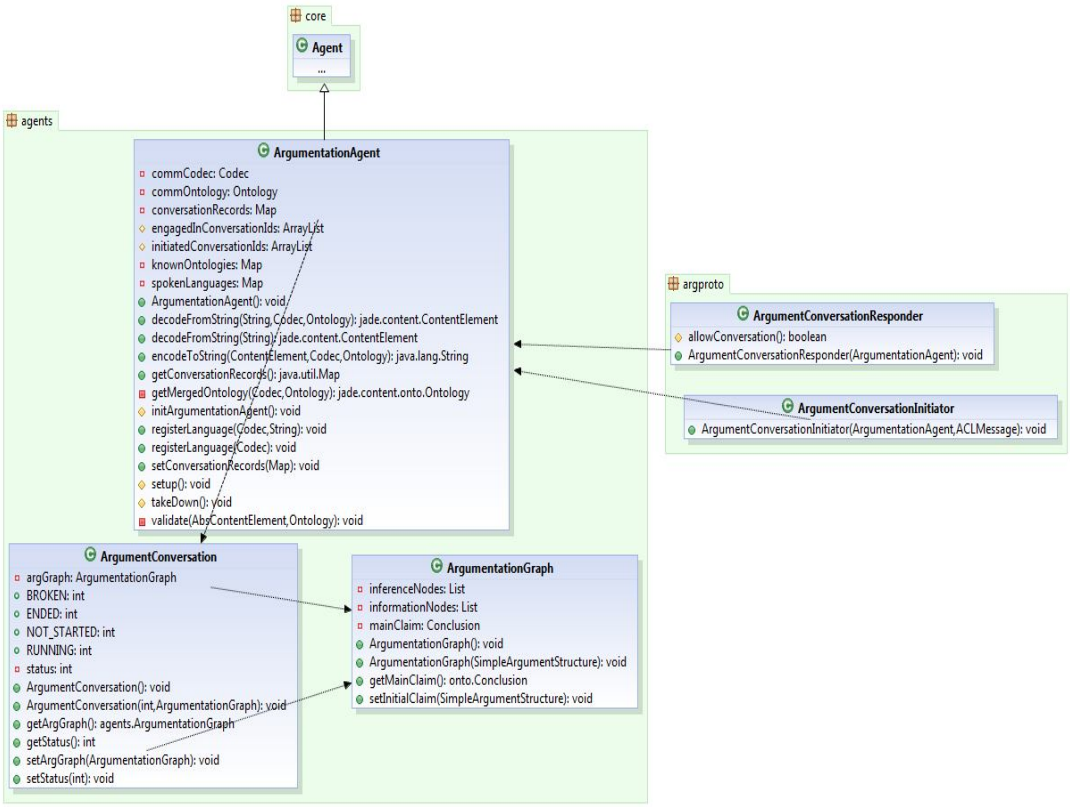

Fig. 4 Class diagram of the Argumentation Agent Architecture

Figure 4 shows the mentioned class diagram. We can immediately see that the ArgumentationAgent class is an extension of the JADE Agent base class.

The standard way to initialize the agent is by the setup() method of the JADE agent class. This method is overridden in the ArgumentationAgent to print a hello message and register the communication language (*commCodec* member) to FIPA-SL and the communication ontology (*commOntology* member) to AIFOntology.

Aside from these, the argumentation agent has containers for the agents' "spoken languages" and known ontologies (the agents' domain ontologies). While the default message content remains a text representation of a statement (as described in section 2), these fields provide the means necessary to deliver content as "sentences" from a given domain ontology.

The argumentation agent has methods that allow it to register any amount of encoding languages (that extend the jade.content.lang.Codec abstract class) and domain ontologies (that extend the jade.content.onto.Ontology class).

The *encodeToString()* and *decodeFromString()* methods are based on the JADE jade.content.lang.Codec facilities that allow the argumentation agent to encode and decode statements from the domain ontology supplied to the methods.
The encoded strings are then attached as the text value of the corresponding ArgDF (AIF reification as described in [11]) statements that make up a chosen scheme for the next argument to be sent to a counterpart agent.

The argumentation agent keeps track of all the argumentation conversations it initiated or is part of by means of the *conversationRecords* mapping. Unique conversation identifiers are mapped to structures of type *ArgumentConversation.* An ArgumentConversation object holds information about the status of that conversation (not started, ongoing, finished and broken) and, most importantly, it keeps an ArgumentationGraph reference which incrementally builds the entire argumentation network of the conversation.

Each argumentation agent will have to register at least one ArgumentConversationResponder behavior that will allow it to answer to incoming conversation requests.
Each time the agent will want to start a new conversation, it will first create the appropriate tracking structures described above and it will register an ArgumentConversationInitiator behavior to start the argumentation process.

3.2. Argumentation Agent Communication protocol

As mentioned earlier, the communication protocol (allowed primitives and correct sequence of messages) is inspired by the argumentation dialogue model introduced in [9].
Figure 5 shows the sequence diagram of the proposed protocol.

The initiator starts the conversation by sending an ARG_INIT_CONV request to the responder. The request has a timeout attached. Either if the requests times out or the responder denies the conversation, the conversation will end.
If the responder accepts the conversation (by replying with an ARG_ACK_CONV) we enter into the main part of the conversation.
The initiator starts by sending an *assert(p)* message containing the initial claim (formulated as an argument following a scheme and encoded using the ArgDF statements).
Following an assert message, the responder can either *accept p* or *challenge p* by replying with an argument *q* that contradicts either one of the premises of *p* or its conclusion altogether.
Upon receiving an *accept* message, the initiator can *assert* another argument, he can *challenge* a previous argument sent by the responder or he can signal the intent of *ending the conversation*.
When receiving a *challenge* message, the initiator will usually try to support its previous claim by challenging the received argument. If it cannot do so, the initiator will *accept* the argument put forward in the received *challenge* message.
At this point, the protocol allows the agents to change roles, that is, the responder will become the main proposer of claims, while the initiator will try to respond to them, all the while still trying to defend his initial claims.
A change of roles can also occur if one of the agents sends an *end conversation request* and the other one will not accept it, but instead keep asserting or challenging existing arguments in the network. The conversation will stop when one of the agents will accept an *end conversation request.*

To allow for the mentioned role switching ability of the protocol, the dialogue rules introduced in [9] were slightly changed.
Thus, an *assert(p)* statement is only used to introduce a new, unchallenged argument *p* in the network. Also, a *challenge(p)* statement will only be answered by an *accept(p)* statement (in which case the roles will change) or a *challenge(q)* statement, where *q* is an argument attacking a premise of *p* or its conclusion (by proving the converse).

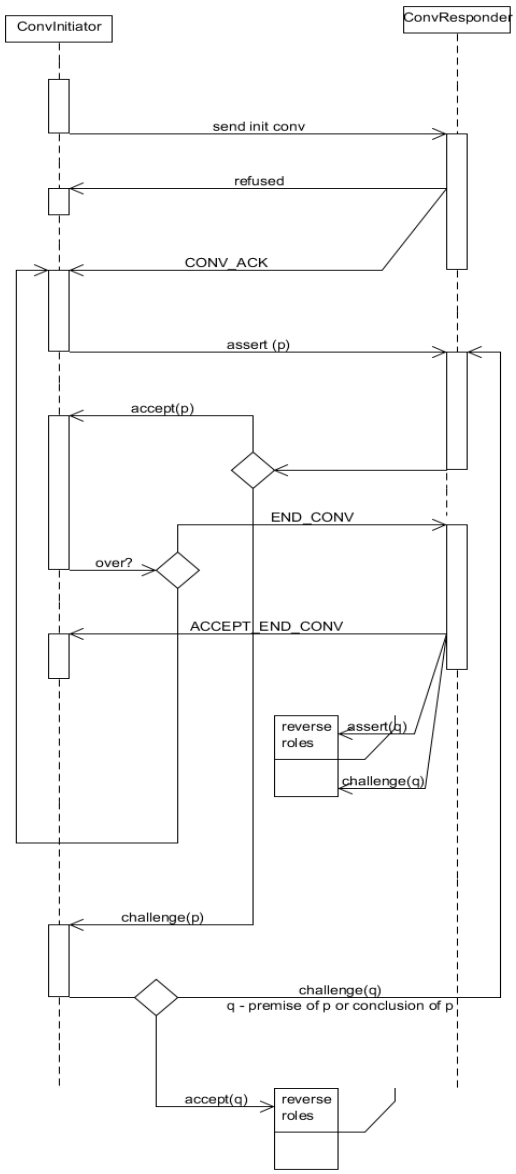

Fig. 5 Sequence diagram for the argumentation protocol

4. Conclusions and possible extensions

In this paper we have presented the architecture of an argumentation agent based on the JADE agent programming platform and the protocol it uses to engage in argumentation conversations. The main advantage of our proposal is its great flexibility and representational strength. By using the AIF ontology to represent both the internal argument network that an agent builds as well as the content used in the protocol's messages, we have laid the groundwork for an extensible architecture.

The first next step will be to deliver shortly a proof of concept project that will feature agents implemented using the current architecture and that reason using simple deductive schemes such as Modus Ponens (which will be explicitly described using the ArgDF statements introduced in section 2.2). A sample use case may be the well-known AI example of modeling the flying abilities of birds and penguins, and reasoning about whether a particular penguin *opus* can fly.
The resulting network would then have to be similar to the following figure.

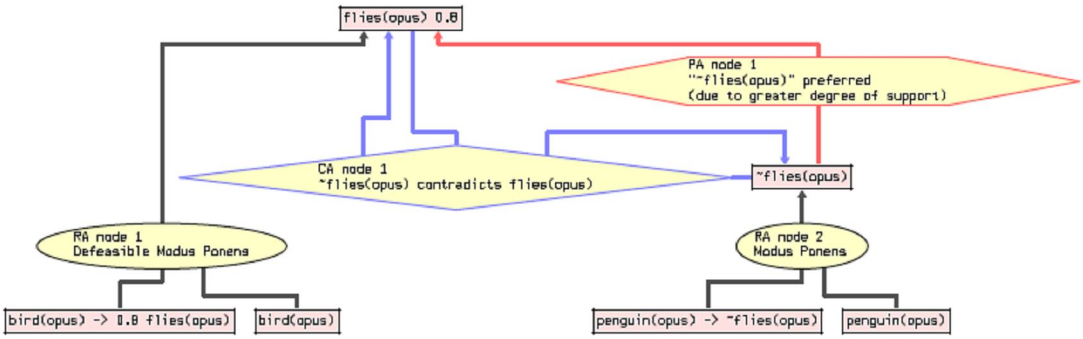

Fig. 6 Modeling arguments "pro" and "con" flying abilities in birds: a concrete example of an argument network.

As to the extensions of the model themselves, the main future work will be done in properly defining rule, conflict and preference schemes using the constructs of the ArgDF and letting the argumentation agent instantiate well defined arguments with their help.

Another important addition will be to couple the argumentation agent with an inference engine that can automatically understand an argument given the domain ontology and the used argumentation scheme and generate a response accordingly. This is a very hard problem and it is expected that it will receive much research effort in the future.

One last future addition is the possibility to view argument networks in external visualization tools. Since support for AIF has recently been implemented in many of them, this will come as natural extension of our argumentation agents' capabilities.

References

[1] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*, pages 291–316. MIT Press, Cambridge, USA, 1997.

[2] FIPA. Communicative Act Library Specification. Standard SC00037J, IEEE Foundation for Intelligent Physical Agents, 3 December 2002.

[3] G. W. A. Rowe, C. A. Reed, and J. Katzav. Araucaria: Marking up argument. In European Conference on Computing and Philosophy, 2003.

[4] truthmapping. http://www.truthmapping.com, 2006.

[5] T. V. Gelder. A reason!able approach to critical thinking. Principal Matters: The Journal for Australasian Secondary School Leaders, 34(6), 2002.

[6] M. Bachler, S. B. Shum, D. D. Roure, D. Michaelides, and K. Page. Ontological mediation of meeting structure: Argumentation, annotation, and navigation. In Proceedings of the 1st International Workshop on Hypermedia and the Semantic Web (HTSW2003), Nottingham, UK, 2003

[7] Chesnevar, C. Mcginnis, J. Modgil, S. Rahwan, I. Reed, C. Simari, G. South, M. Vreeswijk, G. Willmott, S. Towards an argument interchange format. Cambridge University Press. 2006, VOL 21; NUMB 4, pages 293-316

[8] D. N. Walton and E. C. W. Krabbe. Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. SUNY Press, Albany, NY, USA, 1995.

[9] L. Amgoud, N. Maudet, and S. Parsons. Modelling dialogues using argumentation. In E. Durfee, editor, *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, pages 31–38, Boston, MA, USA, 2000. IEEE Press.

[10] J. MacKenzie. Question-begging in non-cumulativesystems. *Journal of philosophical logic*, 8:117–133, 1979.

[11] I. Rahwan and C. Reed. The Argument Interchange Format. Argumentation in Artificial Intelligence, 2009 - Springer

[12] D.Walton. *Argumentation Schemes for Presumptive Reasoning*. Erlbaum, Mahwah NJ, 1996