Towards an Argument Interchange Format for Multi-Agent Systems

Steven Willmott[1], Gerard Vreeswijk[2], Carlos Chesñevar[3], Matthew South[4], Jarred McGinnis[5], Sanjay Modgil[4], Iyad Rahwan[6,5], Chris Reed[7], and Guillermo Simari[8]

[1] Universitat Politècnica de Catalunya, Catalunya, Spain
[2] Universiteit Utrecht, The Netherlands
[3] Universitat de Lleida, Catalunya, Spain
[4] Cancer Research UK, UK
[5] University of Edinburgh, UK
[6] British University in Dubai, UAE
[7] University of Dundee, UK
[8] Universidad Nacional del Sur, Argentina

Joint subimission ArgMAS 2006
See (`http://x-opennet.org/aif`) for previous versions

Abstract. This document describes a strawman specification for an Argument Interchange Format (AIF) that might be used for data exchange between Argumentation tools or communication in Multi-Agent Systems (MAS). The document started life as a skeleton for contributions from participants in the Technical Forum Group meeting in Budapest in September 2005, receiving also input from third parties. The results were subsequentely improved and added to by online discussion to form a more substantial. In its current form, this document is intended to be a strawman model which serves as a point of discussion for the community rather than an attempt at a definitive, all encompassing model. The hope is that it could provide a useful input to ArgMAS discussion in paricular on the utility of common Argumentation Interchange Formats, what form they might take and a potential research / development agenda to help realise them.

1 Introduction and Background

Argumentation is a verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader, by putting forward a constellation of propositions intended to justify (or refute) the standpoint before a rational judge [22, page 5]. The theory of argumentation is a rich, interdisciplinary area of research lying across philosophy, communication studies, linguistics, and psychology. Its techniques and results have found a wide range of applications in both theoretical and practical branches of artificial intelligence and computer science as outlined in various recent reviews [2, 3, 15, 18]. These applications range from specifying semantics for logic programs [4], to natural language text generation [5], to supporting legal reasoning [1], to decision-support for multi-party human decision-making [7] and conflict resolution [20].

In recent years, argumentation theory has been gaining increasing interest in the multi-agent systems (MAS) research community [16, 17]. On one hand, argumentation-based techniques can be used to specify *autonomous agent reasoning*, such as belief revision and decision-making under uncertainty and non-standard preference policies. On the other hand, argumentation can also be used as a vehicle for facilitating *multi-agent interaction*, because argumentation naturally provides tools for designing, implementing and analysing sophisticated forms of interaction among rational agents. Argumentation has made solid contributions to the theory and practice of multi-agent dialogues.

While these efforts have made great progress there remain major barriers to the development and practical deployment of Argumentation systems. One of these barriers is the lack of a shared, agreed notation or "Interchange Format" for argumentation and arguments. The potential benefits of such a format include:

– Providing a convergence point for discussing the syntax and semantics of argumentation-related agent interaction.
– Provide a common basis for discussing and comparing Argumentation scenarios.
– Enabling the development of a variety of compatible tools/systems which share the same argumentation input/output formats.
– Facilitating the development of agents capable of interaction via argumentation using a shared formalism.

While argumentation mark-up languages such as Araucaria,[9] Compendium[10] and ASCE[11] (see [9] for example) already exist they are primarily a means to enable user to structure arguments through diagramatic linkage of natural language sentences. These mark-up languages are not designed to process formal logical statements such as those used within multi-agent systems. As a result, the aim of the Argumentation Interchange Format (AIF) workshop hosted in Budpest, Hungary in September 2005 was to sketch out a strawman document that presents an attempt to consolidate, where possible, the work that has already been done in argumentation mark-up languages and multi-agent systems frameworks. It is hoped that this effort will provide a convergence point for theoretical and practical work in this area, and in particular facilitate:

1. Argument interchange between agents within a particular multi-agent framework.
2. Argument interchange between agents across separate multi-agent frameworks.

[9] `http://araucaria.computing.dundee.ac.uk/`
[10] `http://www.compendiuminstitute.org/tools/compendium.htm`
[11] `http://www.adelard.co.uk/software/asce/`

3. Inspection/manipulation of agent arguments through argument visualization tools.
4. Interchange between argumentation visualization tools.

The remainder of this document provides a first-cut model for such a format in order that it might form a discussion point in the community.

2 Overall Approach

An Argumentation Interchange Format, like any other data representation, requires a well defined *syntax* and *semantics*. The syntax is required as a concrete representation of statements relating to arguments, and the semantics conveys the meaning of statements made using the syntax. However, beyond this basic requirement, there are a wide range of approaches which could be taken for defining both syntax and semantics. In particular, semantics may be explicit (using some previous formal notation with its own syntax and semantics) or implicit (hard coded into a piece of software which subsequently behaves in a given way for each combination of inputs), machine readable or targeted at a human audience (written notes for human consumption), formal or informal, etc. Further questions arise as to whether there should be one single AIF format defined, whether variations should be allowed for, how extensions should be dealt with, etc. Given this range of possibilities the approach taken in this document adheres to the following overall principles:

– *Machine readable syntax*: AIF representations are specifically targeted at machine read/write operations rather than human level documentation. While using formats which are human readable is desirable (for example for debugging purposes) the primary aim of the format is data interchange between software systems.
– *Explicit and (where possible) machine processable semantics*: The semantics of AIF statements are to be stated explicitly in specification documents, such that they may be implemented by multiple tool/system providers. Secondly, where possible, the nature of the semantic definition should enable the implementation of processing tools such as reasoners (for example using some existing logical framework).
– *Unified abstract model, multiple reifications*: the AIF should be defined in terms of: 1) An *abstract model* defining the concepts which could be expressed in an AIF and their relationship to one other, and 2) a set of concrete *reifications / concrete syntaxes* which instantiate these concepts in a particular syntactic formalism (such as XML, Lisp-like S-expressions, etc.). Using this even if different computational environments require different styles of Syntax, interoperability may still be facilitated by similarities at the abstract level.
– *Core concepts, multiple extensions*: recognizing that different applications may require statements about a wide array of different argumentation related concepts, the AIF will be structured as a set of core concepts (those likely to

be common to many applications) and extensions (those which are specialist to particular domains or types of applications). It is anticipated that: A) the core will evolve over time as consensus changes on what is central and applications generate experience, and that B) extensions could be generated by any user of the AIF and, if they turn out to be particularly useful, shared amongst large groups of users (potentially also being merged into the core).

3 Abstract Model / Core Ontology

The foundation for the AIF model is a set of definitions for high-level concepts related to argumentation which may need to be represented in the proposed format. These concepts are gathered into three main groups:

1. *Arguments and Argument Networks*: the core ontology for argument entities and relations between argument entities with the purpose of reification in an AIF (see Section 3.2).
2. *Communication*: the core ontology for items which relate to the interchange of arguments between two or more participants in an environment, including *locutions* and *protocols* (see Section 3.4).
3. *Context*: the core ontology for items associated with environments in which argumentation may take place. These include *participants* in argument exchanges (*agents*), *theories* contained in the environment that are used for argumentation, and other aspects which may affect the meaning of arguments/communication of arguments (see Section 3.5).

In the next subsections an overview of the above concepts is given. Definitions are drawn from existing theories when possible, but may diverge where alignment between theories is needed. Items unique to argumentation (such as the notion of an "argument" itself) are naturally treated in greater depth than items for which more general definitions are already available (such as the notion of an "agent" for example). The relationships between these groups of concepts are shown in Fig. 1.

3.1 The Notion of Argument

Before proceeding with these definitions, it is worth noting that we will not take a position on the precise definition of the notion of "argument" itself, even though later sections do provide structures for describing argument. The reason for this is that initially we found it too difficult to select a single definition acceptable to all. We contend that progress on such a definition might be better made once some consensus is reached on the necessary lower level concepts. A useful starting point for understanding philosophical notions of arguments can however be found in David Hitchcock's input to the original AIF meeting.[12]

[12] `http://www.x-opennet.org/aif/Inputs/aif2005_david_hitchcock_1.pdf`

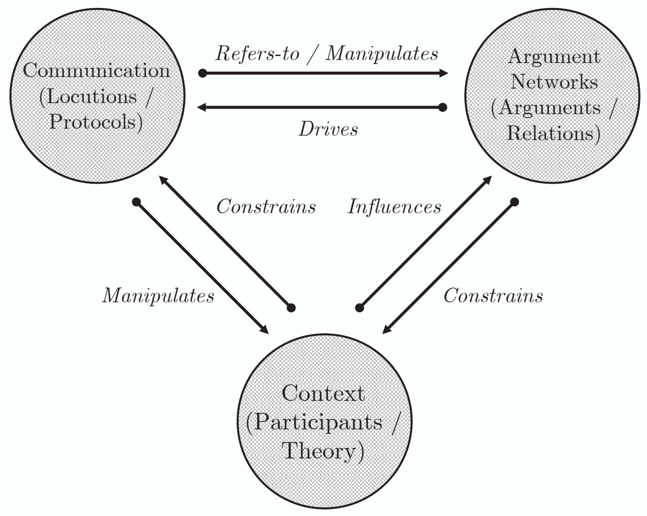
Communication (Locutions / Protocols)
Refers-to / Manipulates
Drives
Argument Networks (Arguments / Relations)
Constrains
Influences
Manipulates
Constrains
Context (Participants / Theory)

Fig. 1. Overview diagram of main groups of concepts defined by the AIF Core Ontology

3.2 Arguments / Argument Networks

The following section defines the top level concepts to be considered for an ontology of arguments and relationships between arguments.

Concepts and Relations: The starting point of this section is the assumption that argument entities can be represented as nodes in a directed graph (di-graph). This di-graph is informally called an *argument network* (AN). An example of an AN is displayed in Fig. 3. This figure will be described later, in Sec. 3.3. The rational for not to restrict ourselves to directed acyclic graphs (DAGs) or even trees is that argumentation formalisms vary to a great extent. A number of formalisms allow for cycles where others forbid them explicitly. One of our basic assumptions is that the core ontology should cater for these differences, and should be able to capture extreme cases.

Nodes: There are two kinds of nodes, namely, *information nodes* (I-nodes) and scheme application nodes or *scheme nodes* (S-nodes) for short (see Fig. 2). Note that one alternative for "scheme node" could be "application node". However, the meaning of "application" is not precise, neglecting the scheme connotation.

Whereas I-nodes relate to content and represent claims that depend on the domain of discourse, S-nodes are applications of *schemes*. Such schemes may be considered as domain-independent patterns of reasoning (that resemble rules of inference in deductive logics but broadened to non-deductive logics and not

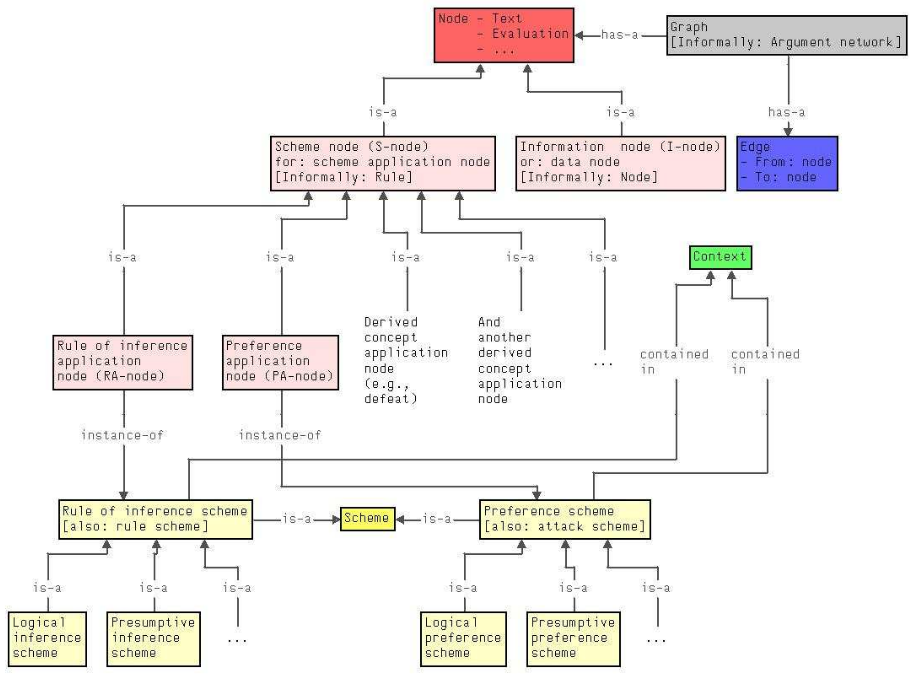
Node - Text
- Evaluation
- ...
has-a
Graph
[Informally: Argument network]
is-a
is-a
has-a
Scheme node (S-node)
for: scheme application node
[Informally: Rule]
Information node (I-node)
or: data node
[Informally: Node]
Edge
- From: node
- To: node
is-a
is-a
is-a
is-a
is-a
Context
Derived concept application node (e.g., defeat)
And another derived concept application node
...
contained in
contained in
Rule of inference application node (RA-node)
Preference application node (PA-node)
instance-of
instance-of
Rule of inference scheme [also: rule scheme]
is-a
Scheme
is-a
Preference scheme [also: attack scheme]
is-a
is-a
is-a
is-a
is-a
is-a
Logical inference scheme
Presumptive inference scheme
...
Logical preference scheme
Presumptive preference scheme
...

Fig. 2. Concepts and relations for an ontology of arguments

restricted to classical logical inference). The present ontology deals with two different types of schemes, namely *inference schemes* and *attack schemes*. Potentially scheme types could exist, such as evaluation schemes and scenario schemes, which will not be addressed here.

If a scheme application node is an application of an inference scheme it is called a *rule of inference application node* (RA-node). If a scheme application node is an application of a preference scheme it is called a *preference application node* (PA-node). Informally, RA-nodes can be seen as applications of rules of inference while PA-nodes can be seen as applications of (possibly abstract) criteria of preference among evaluated nodes.

Node Attributes: Nodes may possess different attributes such as “title,” “text,” “creator,” “type” (decision, action, goal, belief), “creation date,” “evaluation” (or “strength,” or “conditional evaluation table”), “acceptability,” and “polarity” (values either “pro” or “con”). These attributes may vary and are not part of the core ontology. The term “conditional evaluation table” is inspired by its Bayesian analogon named “conditional probability table” (CPT). Most attributes are proper, that is, essential to the node itself, while others are derived. In this example, all attributes except "acceptability" are proper. It is imaginable that a derived attribute such as "acceptability" may be obtained from node-specific attributes through calculation. In this case, "acceptability" may be obtained from "evaluation" through mechanical inference.[13]

Edges: Let us analyze the notion of support. In the context of a graph representing argument-based concepts and relations, a node A is said to *support* node B if and only if an edge runs from A to B. This rather broad notion of support turns out to be remarkably convenient in discussions on argument ontology. Alternative terminology, more akin to graph-theory is "children of".[14]

1. Every node (i.e., every I-node and every S-node) can be supported by zero or more S-nodes.
2. Every S-node can be supported by zero or more I-nodes.

Edges do not need to be explicitly marked, labelled, or otherwise supplied with semantical pointers. A very practical example showing this would be an "edge table" representing edges between nodes. Besides an OID (object identifier) column, such an edge table does not need more than two columns: a `from_oid` field, denoting the OID of the source node, and a `to_oid` field, denoting the OID of the sink node.

If desired, edge types can be inferred from the nodes they connect. Basically there are two types of edges, namely *scheme edges* and *data edges*. Scheme edges emanate from S-nodes and are meant to support conclusions. These conclusions may either be I-nodes or S-nodes. Data edges emanate from I-nodes, necessarily end in S-nodes, and are meant to supply data, or information, to scheme applications. In this way, one may speak of I-to-S edges ("information," or "data" supplying edges), S-to-I edges ("conclusion" edges) and S-to-S edges ("warrant" edges). Table 1 summarizes the relations associated with the semantics of support. Notice that I-to-I edges are forbidden, as will be discussed further on in this section.

To distinguish scheme edges from data edges in diagrams, edges that emanate from S-nodes may be supplied with a closed arrowhead at the end, while edges that emanate from I-nodes may be supplied with an open arrowhead at the end. Edges fall into different categories, such as *support edges* (that are associated or "colored" by the scheme of the S-node they are connected to; for S-to-S edges, the nodes that they emanate from), *inference edges* (those edges that are connected to an RA-node, shown in black in Fig. 3), and *attack edges* (edges

[13] There are voices that advocate to drop derived node attributes altogether, for different algorithms may assign different statuses to arguments within one and the same argument network.

[14] Note however that the term support could be misleading when applied to preference application nodes, as preference application is intuitively associated with concepts such as negation, counterargument and preference. In such cases it may help to think of *negative support*.

	to *I-node*	to *RA-node*	to *PA-node*
from *I-node*		data/information used in applying an inference	data/information used in applying a preference
from *RA-node*	inferring a conclusion in the form of a claim	inferring a conclusion in the form of a scheme application	inferring a conclusion in the form of a preference application
from *PA-node*	applying preferences among information (goals, beliefs, ..)	applying preferences among inference applications	meta-preferences: applying preferences among preference applications

Table 1. Semantics of support.

that are connected to an PA-node, shown in red in Fig. 3).[15] Marking edges and applying arrowheads to edges is not part of the ontology but only meant to help human beings in its interpretation.

Constructions that are not permitted: The ontology is flexible enough to allow for exceptional constructions. Still, it does not account for a number of artifacts. The following list shows a number of constructions that are not accommodated for in the present ontology:

1. I-nodes cannot be linked to other I-nodes. The reason for this restriction is that I-nodes cannot be connected without explaining why the connection is being made. There is always a reason, scheme, justification, inference, or rationale behind a relation between two or more I-nodes.
2. S-nodes may not be employed as I-nodes. Notice that it is difficult to find a compelling example that would justify the use of an S-node as an I-node. A possible example could be *"But previously you said that items that look red generally are red, so in the same way I say here that items that look like an apple generally are an apple"*. In these cases, it seems that it is not really a scheme application that is being used as an I-node-like premise, but rather something slightly different. Also, rather than using an S-node as an I-node, it seems more plausible to re-apply the scheme used for that S-node to create a new S-node.

Derived concepts: Concepts from an extension ontology, in particular concepts such as *rebut*, *undercut*, *defend*, and *defeat* can in principle be derived from the concepts in the diagram that is displayed in Fig. 2. Thus, an argument qualified with derived concepts can in principle be described in terms of basic concepts in a mechanical manner. Nevertheless, such derived concepts may still

[15] Note that in the color printed version of the document different colors are visible for edges for clarity – however, they are not essential to intepretation.

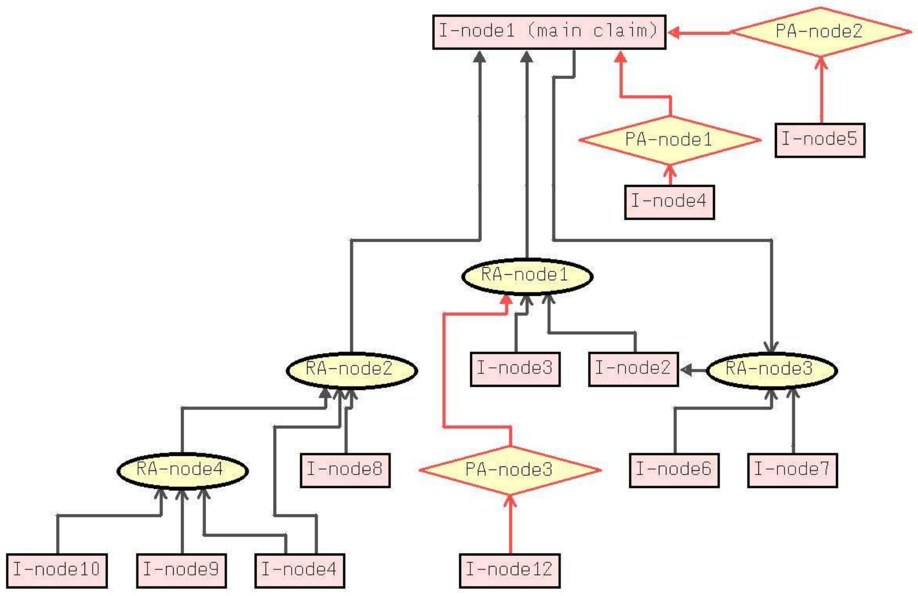

Fig. 3. Sample argument network.

have an important place that should be respected by their inclusion in an extension ontology that we might call "derived concepts" (see further discussions in Section 5).

3.3 Examples

This section presents three examples: an abstract example that shows most of the features of the ontology, a translation of Toulmin's scheme, and a simple concrete example.

Abstract example of an argument network: An abstract example of an argument network is displayed in Fig. 3. This network contains eleven I-nodes, namely I-node1 , . . . , I-node11 and six rule application nodes, namely RA-node1, RA-node2, RA-node3, RA-node4, PA-node1, and PA-node2. This abstract example is meant to demonstrate the flexibility of the core ontology, stretching the limits of the model. Obviously, most existing argument formalisms would not support the constructions shown in this example. Some observations that can be drawn from the diagram:

1. The main claim is supported by two inference applications and two attack applications.

2. Scheme-to-conclusion (SC) edges are drawn with an arrowhead, while premise-to-scheme (PS) edges are drawn without arrowheads. This is for two reasons. The first reason is to distinguish PS from SC edges. The second reason is to disambiguate direction when two S-nodes are connected by an SC edge (notice the SC-edge from RA-node4 to RA-node2). The arrowhead distinction is for diagrammatic purposes only, and it has no added value for representation and interchange formats.
3. I-node4 shows that our model allows for multiple node references. Thus, nodes may be referred to more than once. In particular, an argument network (AN) need not be a tree.
4. The two inference applications

$$\begin{aligned} \text{I-node2, I-node3} &-(\text{RA-node1})\rightarrow \text{I-node1 (main claim)} \\ \text{I-node1, I-node6, I-node7} &-(\text{RA-node3})\rightarrow \text{I-node2} \end{aligned}$$

   show that cycles in theory may occur.
5. If I-nodes are attacked, then the premises connected to the intermediate PA-node are called *rebutters*. For example, I-node1 is rebutted by I-node4 and I-node-5 through PA-node-1 and PA-node-2, respectively. These are two independent rebutters.
6. If RA-nodes are attacked, then the premises connected to the intermediate PA-node are called *undercutters*. For example, RA-node1 is undercut by I-node12 through PA-node3. In general, every type of node may be attacked, including attack nodes themselves. The diagram does not contain an instance of the latter.

Argumentation *à la* Toulmin. Example: Toulmin's scheme as depicted in (Eq. 1) is constituted of six essential elements, namely data (D), warrant (W), backing (B), qualifier (Q), rebuttal (R) and claim (C). A (somewhat liberal) translation is displayed in Fig. 4. The shadow-encircled nodes together relate to the original backing B.

$$\begin{tikzcd}[row sep=small] D \arrow[r] & Q, C \\ \text{since } W \arrow[u, dash] & \text{unless } R \arrow[u, dash] \\ B \arrow[u, dash] \end{tikzcd} \tag{1}$$

Notice that in Fig. 4, R (rebuttal) attacks C (claim) rather than W (warrant). It is not clear from *"The Uses of Argument"* [21] whether R should attack C or W. Since an attack on C is called a rebuttal, and since an attack on W is called an undercutter in our terminology, we have chosen the one which is consistent with it. Nevertheless, R can reasonably be taken to attack C, to support not-C, to attack W, or to attack an implicit warrant (the dots). This document does not advocate a mechanism for translation but merely that any of those translations should be representable in the present ontology.

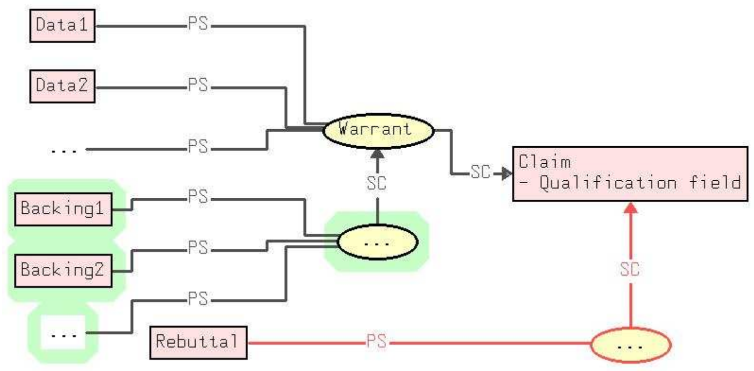

Fig. 4. Toulmin scheme.

A concrete and simple example: In Fig. 5 we show a concrete and simple example of an argument network for handling the well-known AI example of modelling the flying abilities of birds and penguins, and reasoning about whether a particular penguin `opus` can fly. In this case there are two arguments, one for `fly(opus)` and one for `~fly(opus)`.

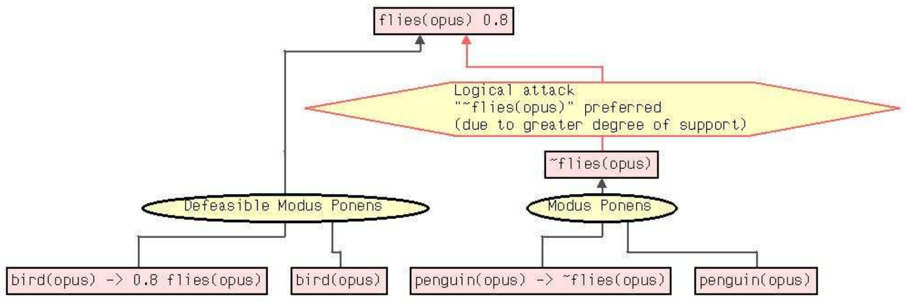

Fig. 5. Concrete example of an argument network.

The argument for `~fly(opus)` is composed of one scheme-application, namely *Modus Ponens* (MP). A simplistic version of MP reads as follows: *if there are two information nodes* `A(x)` *and* `A(x)->B(x)` *then conclusively infer* `B(x)`. The argument for `fly(opus)` is composed of one scheme-application, namely *defeasible Modus Ponens* (dMP). A simplistic version of dMP reads as follows: *If there are two information nodes* `A(x)` *and* `A(x)-(qualifier)->B(x)` *then defeasibly* *infer* `B(x)`. The conflicting nodes `fly(opus)` and `~fly(opus)` are related by a PA-node that says that the argument against `fly(opus)` is conclusive and therefore preferred over the argument for `fly(opus)`. This PA-node is an instance of a more general scheme saying that deductive arguments always win out over non-deductive arguments.

3.4 Communication: Locution / Protocols

The second group of concepts identified in discussions are those which concern communication in the context of argumentation, for example, concepts which capture:

– The utterance of a statement containing an argument or argument network by an agent.
– A sequence of legal statements making reference to arguments/argument networks which could be made by a set of agents in order to make a decision or reach some other goal.

In turn, as with arguments / argument networks, communication also takes place in a context –elements of which may affect the interpretation of statements (such as references to the participants in a dialogue, the ontologies applying, the semantic models adopted etc.). Presentation here is initially split into two parts:

– *Locutions*: individual words, phrases or expressions uttered by an agent.
– *Interaction Protocols*: sequences of locutions involving one or more (usually at least two) agents and usually designed to achieve a specific goal (such as reaching an agreement or giving information).

Hence locutions form the basic building blocks of protocols. It is important to note however that there are different "schools of thought" on how the semantics for locutions and protocols should be defined in terms of one another. One approach, such as FIPA ACL [6], holds that semantics are attributed to individual locutions –and the semantics of a protocol are a compound of the semantics of individual locutions. Another approach holds that the semantics of locutions vary depending on their context (e.g. the commitments made thus far) and hence their place in a particular protocol [11].

Locutions: A rich literature exists on locutions of various types and their semantics. In terms of general agent communication, languages such as FIPA-ACL and KQML define sets of general locutions such as *inform*, *request*, *query*, *tell* and so on –each with an associated formal logical semantics. However, while these languages may provide useful resources, it is also clear from more specific argumentation literature that the types of locutions which occur frequently are more specific / different to those found in FIPA-ACL / KQML. Examples include: *assert*, *accept*, *challenge*, *question*, *concede*, and *prefer*.

While different authors use different labels for different locutions, there seem to often be similarities in semantics. Work such as that by McBurney, Parsons and Wooldridge [14], McBurney and Parsons [13], Maudet and Chaib-draa [10] and Mcburney, Hitchcock, and Parsons [12] provides a starting point for potentially determining a limited number of locutions which could form the core of an AIF, others potentially being added as extensions. In this setting, at a more general level however an AIF core ontology should usefully define the notions of:

– *Locution*: the notion of a locution, and its associated properties, which might include (taken from the FIPA-ACL message structure specification [6]):[16]
  - *Sender*: the agent uttering a locution (note that a distinction could also be made between the sender who makes and utterance and the originator(s) – an agent or group of agents responsible for generating the utterance.)
  - *Receiver* or *Receivers*: Agents "hearing" an utterance (distinctions could be made between intended recipients, those intentionally made aware of the message but not the intended recipients and those who unintentionally become aware of an utterance).
  - *Ontologies*: the ontologies which hold and define elements of the content.
  - *Language*: the content language used in the content part of the message (which should itself have a formal semantics).
  - *Protocol*: the protocol a locution is part of.
  - *Content*: the object of the locution.
  - Message management elements: items such as a *message-identifier*, a *conversation-identifier*, *in-reply-to field* etc.

– *Individual Locutions*: potentially a set of subclasses of the class of locutions which capture individual locutions such as those listed at the beginning of this section.

Interaction Protocols: It is possible to construct comprehensive standards of language usage for computational systems that are widely used and relatively precise. This is the case for programming language standards (such as PROLOG, dialects of ADA, *etc*). By contrast, in areas where standardization of more abstract concepts is required, consensus appears to be much harder to achieve, because abstract concepts are difficult to pin down uniquely in a simple way. In this circumstance it is often expedient to define precisely a core standard, containing only those elements essential to getting the job done, and then allow extensions to this core in a controlled (but perhaps less precise) way. An example of this form of standardization is the Process Interchange Format (PIF) which is a standard for describing processes. The PIF core contains a small number of very generic concepts at the heart of that standard and then allows those with specific process description needs to meet their own requirements by building on that core.

[16] Note that additionally one could add a slot for *semantics* which points to the defined formal semantics for the locution.

$$\begin{array}{rcl}
Model & := & \{Clause, \ldots\} \\
Clause & := & Role :: Def \\
Role & := & a(Type, Id) \\
Def & := & Role \mid Message \mid Def\ then\ Def \mid Def\ or\ Def \\
Message & := & M \Rightarrow Role \mid M \Rightarrow Role \leftarrow C \mid M \Leftarrow Role \mid C \leftarrow M \Leftarrow Role \\
C & := & Constant \mid P(Term, \ldots) \mid \neg C \mid C \wedge C \mid C \vee C \\
Type & := & Term \\
Id & := & Constant \mid Variable \\
M & := & Term \\
Term & := & Constant \mid Variable \mid P(Term, \ldots) \\
Constant & := & \text{lower case character sequence or number} \\
Variable & := & \text{upper case character sequence or number}
\end{array}$$

Fig. 6. LCC syntax

The definition of a interaction protocol language as part of an argument interchange format provides a number of advantages. If the language can be used for computation then the standard is, effectively, a programming standard and history suggests that such standards tend to be durable because they connect to practice (or fail to connect and then die cleanly). If it is also declarative – and hence independent of current fashion in low level implementation languages or basic communications protocols– then it can support formal analysis and verification more readily. In addition, the use of a high level language arguably facilitates human readability. For software engineers there is a natural notion of pattern in the design of protocols and this is one approach to extension from a core protocol syntax to a (more interesting) set of extensions via patterns.

Protocols are an area where traditional computer science helps supply standards. For example, Figure 6 defines the syntax of the Lightweight Coordination Calculus (LCC) that uses a combination of traditional specification drawn from CCS and logic programming (for details on LCC see [19]). An interaction model in LCC is a set of clauses, each of which defines how a role in the interaction must be performed. Roles are described by the type of role and an identifier for the individual agent undertaking that role. The definition of performance of a role is constructed using combinations of the sequence operator (‘*then*’) or choice operator (‘*or*’) to connect messages and changes of role. Messages are either outgoing to another agent in a given role (‘$\Rightarrow$’) or incoming from another agent in a given role (‘$\Leftarrow$’). Message input/output or change of role can be governed by a constraint defined using the normal logical operators for conjunction, disjunction and negation. Notice that there is no commitment to the system of logic through which constraints are solved –on the contrary, we would expect different agents to operate different constraint solvers. Hence the standardization in LCC is on the generic language for describing interaction (only) and in this sense it is “core”. It also has the added benefit of having a style of description that is close to computation –in this case quite close to logic programming (despite the process operators) where we already have a successful ISO standard.

3.5 Context: General Context / Participants / Theory

The third group of concepts in the ontology is that of elements which form the context in which argumentation takes place. In keeping with the distinction already made between concepts for communication and those for arguments / argument networks, concepts related to context may also be usefully grouped into these two areas.

Communication Context: Here, context captures information relevant to argument-based dialogues. These include:

– *Participants:* We may require references to agents taking place in the dialogue, possibly including:
  1. Participant ID: an identifier for a participant.
  2. Participant role: the role of the participant in relation to the dialogue (e.g. pro, con, persuader, buyer, seller, etc.). This may influence the way dialogue proceeds.
– *Dialogue topic:* This refers to the main issue under discussion (e.g. the question under enquiry, or resource under negotiation).
– *Dialogue type:* a reference to the type of the dialogue (e.g. persuasion, negotiation [23]). This can be simply a name, or it can be a pointer to more elaborate dialogue typology.
– *Background theory:* This includes statements that participants agree upon (e.g. legal rules), and which may be used to construct arguments within the dialogue.
– *Commitment stores:* This is a data structure that allows agents to add and remove commitments during their dialogues [8].
– *Commitment rules:* These are rules that specify how dialogue participants may modify the content of commitment stores.

Argument Network Context: Here, context captures information relevant to the interpretation and processing of the argument network.

– *Argumentation theory rules:* These are the rules that specify the way arguments are constructed and interpreted. In a way, they represent the underlying formal argumentation theory. These include:
  1. Inference rules: These can be thought of as the specifications of the types of inference application nodes that can be used in the argument network.
  2. Preference rules: Similarly, these can be thought of as the specifications of the types of preference application nodes that can be used in the argument network.

- *Background theory:* This includes statements taken for granted (e.g. legal rules), and which may be used to interpret or process arguments.
- *Domain ontologies:* One could add references to ontologies that may be used to interpret argument networks. For example, suppose an argument network represents claims and justifications of the medical properties of a particular drug. In order to process these arguments automatically, we may benefit from a specialized medical drug ontology while interpreting these arguments.

4 Reifications

Reifications of the concepts defined in the AIF are *concretizations from abstract to more concrete definitions*. In particular the primary use of reifications in AIF is to define concrete syntaxes which can be unambiguously serialized and de-serialized for transmission between two communicating participants exchanging arguments or between two software tools using the AIF:

- More than one reification may exist.
- Two different reifications may not be interoperable. That is, serializers for one reification may produce output which is not readable by parsers for another.
- While individual reifications will each aim to capture the semantics of the concepts defined in the AIF ontologies, they may also be influenced by the semantics of the encoding language used. Hence minor semantic differences as well as syntactic differences may arise.

A simple example of what is meant by a reification can be seen in the AIF input document by Willmott, Fox and Reed to the original AIF event.[17]

5 Conclusions and Open Issues

As described in the introduction, the development of an AIF is a highly challenging endeavor and this document is intended as a discussion starter and not a fully fledged proposal. Further, as noted in Section 3.2, the current model may well not capture all types of argumentation that are of interest. Specific significant open issues which arose during discussion included:

1. Currently no distinction is being made for AIF formalisms which might be used in GUI/Tool import-export type application and those which might be used in agent-to-agent communication. While the core concepts may be the same it remains an open issue as to whether one format can really adequately cover both cases.

[17] `http://x-opennet.org/aif/Inputs/aif2005_steven_willmott_2.pdf`

2. Given the potential richness of the communication concepts ontology it remains an open issue as to how close to generic Agent Communication Languages (ACLs – such as FIPA-ACL, KQML etc.) AIF definitions may get. This affects possible re-use of ACL concepts and/or overlap with them and/or worries about tractability issues which affected ACL semantics also affecting the semantics of concepts defined here.
3. How should the community of users around the AIF organize themselves to agree on core concepts and extensions?
4. How should reifications be generated in detail from high level concepts (e.g. development of specific RDF / XML schemas or other syntax forms?

A longer version of this document, initial inputs, previous versions and a discussion forum for feedback can be found on the AIF website at `http://x-opennet.org/aif/`.

5.1 Acknowledgments

Support is also gratefully acknowledged from Agentlink III [18] European Commission and the ASPIC (FP6-IST-002307)[19] research funded projects. Additional inputs and contributions are also gratefully acknowledged from all of the following: Leila Amgoud, Trevor Bench-Capon, Jamal Bentahar, Ivan Bratko, Martin Caminada, Sylvie Doutre, John Fox, Dan Grecu, David Hitchcock, Tsakou Ioanna, Paul Krause, Nicolas Maudet, Peter McBurney, Maxime Morge, Martin Mozina, Simon Parsons, Henri Prade, Henry Prakken, Chris Reed, Dave Robertson, Michael Rovatsos, Carles Sierra, and Michael Wooldridge.

While efforts have been made to reach a consensus on the content of this document, it is important to note that it remains the integration of a wide range of inputs, hence the final result *may not necessarily reflect the opinion of everybody who contributed* – authorship or being listed as contributor does not necessarily imply complete agreement with the text.

References

1. T. J. M. Bench-Capon. Argument in artificial intelligence and law. *Artificial Intelligence and Law*, 5(4):249–261, 1997.
2. D. Carbogim, D. Robertson, and J. Lee. Argument-based applications to knowledge engineering. *Knowledge Engineering Review*, 15(2):119–149, 2000.
3. C. I. Chesñevar, A. Maguitman, and R. P. Loui. Logical models of arguments. *ACM Computing Surveys*, 32(4):337–383, 2000.
4. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and n-person games. *Artifical Intelligence*, 77:321–357, 1995.
5. M. Elhadad. Using argumentation in text generation. *Journal of Pragmatics*, 24:189–220, 1995.

[18] `http://www.agentlink.org`
[19] `http://www.argumentation.org`

6. FIPA. Communicative Act Library Specification. Technical Report XC00037H, Foundation for Intelligent Physical Agents, 10 August 2001.
7. T. F. Gordon and N. Karacapilidis. The Zeno argumentation framework. In *Proceedings of the Sixth International Conference on AI and Law*, pages 10–18, New York, NY, USA, 1997. ACM Press.
8. C. L. Hamblin. *Fallacies*. Methuen, London, UK, 1970.
9. P. A. Kirschner, S. J. B. Schum, and C. S. Carr, editors. *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making*. Springer Verlag, London, 2003.
10. N. Maudet and B. Chaib-draa. Commitment-based and dialogue-game based protocols: new trends in agent communication languages. *The Knowledge Engineering Review*, 17(2):157–179, June 2002.
11. N. Maudet and B. Chaib-draa. Commitment-based and dialogue-game based protocols – new trends in agent communication language. *Knowledge Engineering Review*, 17(2):157–179, 2003.
12. P. McBurney, D. Hitchcock, and S. Parsons. The eight-fold way of deliberation dialogue. *Intelligent Systems (In press)*, 2005.
13. P. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information, Special Issue on Logic and Games*, 11(3):315–334, 2002.
14. P. McBurney, S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, pages 402–409. ACM Press, July 2002.
15. H. Prakken and G. A. W. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer Academic Publishers, 2002.
16. I. Rahwan. (Editor) Special Issue on Argumentation in Multi-Agent Systems. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 11(2):115–206, September 2005.
17. I. Rahwan, P. Moraitis, and C. Reed, editors. *Argumentation in Multi-Agent Systems: First International Workshop, ArgMAS 2004, New York, NY, USA, July 19, 2004, Revised Selected and Invited Papers*, volume 3366 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin, Germany, 2005.
18. C. Reed and T. J. Norman, editors. *Argumentation Machines: New Frontiers in Argument and Computation*, volume 9 of *Argumentation Library*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.
19. D. Robertson. Multi-agent coordination as distributed logic programming. In *International Conference on Logic Programming*, pages 416–430, Sant-Malo, France, 2004.
20. K. Sycara. The PERSUADER. In D. Shapiro, editor, *The Encyclopedia of Artificial Intelligence*. John Wiley & Sons, January 1992.
21. S. Toulmin. *The Uses of Arguments*. Cambridge University Press, 1958.
22. F. H. van Eemeren, R. F. Grootendorst, and F. S. Henkemans. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Applications*. Lawrence Erlbaum Associates, Hillsdale NJ, USA, 1996.
23. D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Press, Albany NY, USA, 1995.