Chapter 19
The Argument Interchange Format

Iyad Rahwan and Chris Reed

1 Introduction

While significant progress has been made 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. One of these barriers is the lack of a shared, agreed notation or "interchange format" for argumentation and arguments. In the last years a number of different argument mark-up languages have been proposed in the context of tools developed for argument visualisation and construction (see [10] for a review). Thus, for example, the Assurance and Safety Case Environment (ASCE)[1] is a graphical and narrative authoring tool for developing and managing assurance cases, safety cases and other complex project documentation. ASCE relies on an ontology for *arguments about safety* based on *claims*, *arguments* and *evidence* [8]. Another mark-up language was developed for Compendium,[2] a semantic hypertext concept mapping tool. The Compendium argument ontology enables construction of networks, in which nodes represent *issues*, *positions* and *arguments*.

The analysis and study of human argument has also prompted the development of specialised argument mark-up languages and tools. Two particularly relevant developments in this direction are *ClaiMaker* [5] and AML [18]. *ClaiMaker* and related technologies [5] provide a set of tools for individuals or distributed communities to publish and contest ideas and arguments, as is required in contested domains such as research literatures, intelligence analysis, or public debate. This system is based on the *ScholOnto* ontology [4], which can express a number of basic reasoning

Iyad Rahwan
British University in Dubai, UAE & University of Edinburgh, UK, e-mail: irahwan@acm.org

Chris Reed
University of Dundee, UK e-mail: chris@computing.dundee.ac.uk

[1] http://www.adelard.co.uk/software/asce/

[2] http://www.compendiuminstitute.org/tools/compendium.htm

I. Rahwan, G. R. Simari (eds.), *Argumentation in Artificial Intelligence,*
DOI 10.1007/978-0-387-98197-0_19,

schemes (causality, support) and relationships between concepts found in scholarly discourse (e.g. similarity of ideas, taxonomies of concepts, etc.). The argument-markup language (AML) used by the Araucaria system [18] is an XML-based language designed for the markup of analysed human argument. The syntax of AML is specified in a Document Type Definition (DTD) which imposes structural constraints on the form of valid AML documents. AML was primarily produced for use in the Araucaria tool, though has more recently been adopted elsewhere.

These various attempts at providing argument mark-up languages share two major limitations. Firstly, each particular language is designed for use with a specific tool (usually for the purpose of facilitating argument visualisation) rather than for facilitating inter-operability of arguments among a variety of tools. As a consequence, the semantics of arguments specified using these languages is tightly coupled with particular schemes to be interpreted in a specific tool and according to a specific underlying theory. Thus, for example, arguments in the Compendium concept mapping tool are to be interpreted in relation to a rigorous theory of issue-based information systems. Clearly, in order to enable true interoperability of arguments and argument structures we need an argument description language that can be extended beyond a particular argumentation theory, enabling us to accommodate a variety of argumentation theories and schemes. Another limitation of the above argument mark-up languages is that they are primarily aimed at enabling users to structure arguments through diagrammatic linkage of natural language sentences. Hence, these mark-up languages are not designed to process formal logical statements such as those used within multi-agent systems. For example, AML imposes structural limitations on well formed arguments, but provides no semantic model. Such a semantic model is an important requirement in order to enable the automatic processing of argument structures by heterogeneous software agents.

In order to address these limitations, a group of researchers interested in 'argument and computation' gathered for a workshop[3] whose aim was to sketch an *Argumentation Interchange Format (AIF)* which consolidates –where possible– the work in argumentation mark-up languages and multi-agent system frameworks by focusing on two main aims:

- to facilitate the development of (closed or open) multi-agent systems capable of argumentation-based reasoning and interaction using a shared formalism;
- to facilitate data interchange among tools for argument manipulation and argument visualization.

This article describes and analyzes the main components of the draft specification for AIF. It must be remarked that AIF as it stands represents a consensus 'abstract model' established by researchers across fields of argumentation, artificial intelligence and multi-agent systems. In its current form, this specification is intended as a starting point for further discussion and elaboration by the community, rather than an attempt at a definitive, all encompassing model. In order to demonstrate the power of the proposed approach, we describe use cases which show how AIF fits

[3] AgentLink Technical Forum Group meeting, Budapest, Hungary, September 2005.

into some argument-based tools and applications. We also illustrate a number of concrete realisations or 'reifications' of the proposed abstract model.

2 The Core AIF

In this section, we briefly describe the first AIF draft specification, as reported in more detail elsewhere [6] and subsequently formalised in [16, 14].

The core AIF has two types of nodes: *information nodes* (or *I-nodes*) and *scheme nodes* (or *S-nodes*). These are represented by two disjoint sets, $\mathcal{N}_I \subset \mathcal{N}$ and $\mathcal{N}_S \subset \mathcal{N}$, respectively. 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 belong to a class, $\mathcal{S}$, and are classified into the types: *rule of inference scheme*, *conflict scheme*, and *preference scheme*. We denote these using the disjoint sets $\mathcal{S}^R$, $\mathcal{S}^C$ and $\mathcal{S}^P$, respectively. The predicate $(\mathsf{uses} : \mathcal{N}_S \times \mathcal{S})$ is used to express the fact that a particular scheme node uses (or instantiates) a particular 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, i.e., $\forall n \in \mathcal{N}_S, \exists s \in \mathcal{S}$ such that $\mathsf{uses}(n, s)$.

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*). These are represented as three disjoint sets: $\mathcal{N}_S^{RA} \subseteq \mathcal{N}_S$, $\mathcal{N}_S^{PA} \subseteq \mathcal{N}_S$, and $\mathcal{N}_S^{CA} \subseteq \mathcal{N}_S$, respectively. The word 'application' on each of these types was introduced in the AIF as a reminder that these nodes function as instances, not classes, of possibly generic inference rules. Intuitively, $\mathcal{N}_S^{RA}$ captures nodes that represent (possibly non-deductive) rules of inference, $\mathcal{N}_S^{CA}$ captures applications of criteria (declarative specifications) defining conflict (e.g. among a proposition and its negation, etc.), and $\mathcal{N}_S^{PA}$ are applications of (possibly abstract) criteria of preference among evaluated nodes.

The AIF specification does not type its edges. The (informal) semantics of edges can be inferred from the types of nodes they connect. One of the restrictions is that no outgoing edge from an I-node can be linked directly to another I-node. This ensures that the type of any relationship between two pieces of information must be specified explicitly via an intermediate S-node.

Definition 19.1. (Argument Network) An *argument network* Φ is a graph consisting of:

- a set $\mathcal{N} = \mathcal{N}_I \cup \mathcal{N}_S$ of vertices (or *nodes*); and
- a binary relation $\xrightarrow{edge}$: $\mathcal{N} \times \mathcal{N}$ representing edges.

where $\nexists(i, j) \in \xrightarrow{edge}$ where both $i \in \mathcal{N}_I$ and $j \in \mathcal{N}_I$

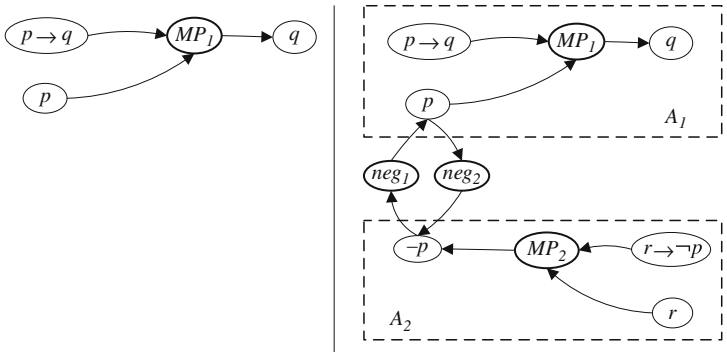

Fig. 19.1 Examples of simple arguments; S-Nodes denoted with a thicker border

A *simple* argument can be represented by linking premises to a conclusion.

Definition 19.2. (Simple Argument) A *simple argument*, in network Φ and schemes $\mathcal{S}$, is a tuple $\langle P, \tau, c \rangle$ where:

- $P \subseteq \mathcal{N}_I$ is a set of nodes denoting premises;
- $\tau \in \mathcal{N}_S^{RA}$ is a rule of inference application node;
- $c \in \mathcal{N}_I$ is a node denoting the conclusion;

such that $\tau \xrightarrow{edge} c$, $\mathsf{uses}(\tau, s)$ where $s \in \mathcal{S}$, and $\forall p \in P$ we have $p \xrightarrow{edge} \tau$.

Following is a description of a simple argument in propositional logic, depicted in Figure 19.1(a).

Example 19.1. **(Simple Argument)**

The tuple $A_1 = \langle \{p, p \rightarrow q\}, MP_1, q \rangle$ is a simple argument in propositional language $\mathcal{L}$, where p, $(p \rightarrow q) \in \mathcal{N}_I$ are nodes representing premises, and $q \in \mathcal{N}_I$ is a node representing the conclusion. In between them, the node $MP_1 \in \mathcal{N}_S^{RA}$ is a rule of inference application node (i.e., RA-node) that uses the *modus ponens* natural deduction scheme, which can be formally written as follows: $\mathsf{uses}(MP_1, \forall A, B \in \mathcal{L} \ \frac{A \quad A \rightarrow B}{B})$.

An attack or conflict from one information or scheme node to another information or scheme node is captured through a CA-node, which captures the type of conflict. The attacker is linked to the CA-node, and the CA-node is subsequently linked to the attacked node. Note that since edges are directed, each CA-node captures attack in one direction. Symmetric attack would require two CA-nodes, one in each direction. The following example describes a conflict between two simple arguments (see Figure 19.1(b)).

Example 19.2. **(Simple Arguments in Conflict)**

Recall the simple argument $A_1 = \langle \{p, p \rightarrow q\}, MP_1, q \rangle$. And consider another simple argument $A_2 = \langle \{r, r \rightarrow \neg p\}, MP_2, \neg p \rangle$. Argument A_2 undermines A_1 by supporting the negation of the latter's premise. This (symmetric) propositional conflict is captured through two CA-nodes: neg_1 and neg_2, both of which insantiate a conflict scheme based on propositional contraries.

3 An Extended AIF in RDF

In this section, we present a brief description of an implementation of an extended AIF ontology which may be used as a seed for a variety of Semantic Web argument annotation tools. The ontology is described in detail in a recent joint paper with other colleagues [16]. It enables the annotation of arguments using RDF, and is based on the AIF, extended with Walton's account of argumentation schemes [22].

3.1 Representing Argument Schemes

Recall that schemes are *forms* of argument, representing stereotypical ways of drawing inferences from particular patterns of premises to conclusions. We consider the set of schemes $\mathcal{S}$ as themselves nodes in the argument network. And we introduce a new class of nodes, called *forms* (or *F-nodes*), captured in the set $\mathcal{N}_F \subseteq \mathcal{N}$. Two distinct types of forms are presented: *premise descriptors* and *conclusion descriptors*, denoted by $\mathcal{N}_F^{Prem} \subseteq \mathcal{N}_F$ and $\mathcal{N}_F^{Conc} \subseteq \mathcal{N}_F$, respectively. As can be seen in Figure 19.2, we can now explicitly link each node in the actual argument (the four unshaded nodes at the bottom right) to the form node it instantiates (the four shaded nodes at the top right).[4] Notice that here, we expressed the predicate 'uses' with the edge $\xrightarrow{fulfilsScheme}: \mathcal{N}_S \times \mathcal{S}$.

Since each critical question corresponds either to a presumption or an exception, we provide explicit descriptions of the presumptions and exceptions associated with each scheme. To express the scheme's presumptions, we add a new type of F-node called *presumption*, represented by the set $\mathcal{N}_F^{Pres} \subseteq \mathcal{N}_F$, and linked to the scheme via a new edge type, $\xrightarrow{hasPresumption}: \mathcal{S} \times \mathcal{N}_F^{Pres}$. This is shown in the three (shaded) presumption nodes at the bottom left of Figure 19.2. As for representing exceptions, the AIF offers a more expressive possibility. In just the same way that stereotypical patterns of the passage of deductive, inductive and presumptive inference can be captured as rule of inference schemes, so too can the stereotypical ways of characterising conflict be captured as conflict schemes. Conflict, like inference, has some patterns that are reminiscent of deduction in their absolutism (such as the conflict between a proposition and its complement), as well as others that are reminiscent

[4] To improve readability, we will start using typed edges. All typed edges will take the form $\xrightarrow{type}$, where *type* is the type of edge, and $\xrightarrow{type} \subseteq \xrightarrow{edge}$.

of non-deductive inference in their heuristic nature (such as the conflict between two courses of action with incompatible resource allocations). Thus, exceptions can most accurately be presented as conflict scheme descriptions (see top left of Figure 19.2).

Finally, in Walton's account of schemes, some presumptions may be implicitly or explicitly *entailed* by a premise. While the truth of a premise may be questioned directly, questioning associated with the underlying presumptions can be more specific, capturing the nuances expressed in Walton's characterisation. This relationship, between is captured explicitly using a predicate ($\xrightarrow{entails}: \mathcal{N}_F^{Prem} \times \mathcal{N}_F^{Pres}$).

Definition 19.3. (Presumptive Inference Scheme Description) A *presumptive inference scheme description* is a tuple $\langle PD, \alpha, cd, \Psi, \Gamma, \xrightarrow{entails} \rangle$ where:

- $PD \subseteq \mathcal{N}_F^{Prem}$ is a set of *premise descriptors*;
- $\alpha \in \mathcal{S}^R$ is the scheme;
- $cd \in \mathcal{N}_F^{Conc}$ is a *conclusion descriptor*.
- $\Psi \subseteq \mathcal{N}_F^{Pres}$ is a set of *presumption descriptors*;
- $\Gamma \subseteq \mathcal{S}^C$ is a set of *exceptions*; and
- $\xrightarrow{entails} \subseteq \mathcal{N}_F^{Prem} \times \mathcal{N}_F^{Pres}$

such that:

- $\alpha \xrightarrow{hasConcDesc} cd$;
- $\forall pd \in PD$ we have $\alpha \xrightarrow{hasPremiseDesc} pd$;
- $\forall \psi \in \Psi$ we have $\alpha \xrightarrow{hasPresumption} \psi$;
- $\forall \gamma \in \Gamma$ we have $\alpha \xrightarrow{hasException} \gamma$;

With the description of the scheme in place, we can now show how argument structures can be linked to scheme structures. In particular, we define a presumptive argument, which is an extension of the definition of a simple argument.

Definition 19.4. (Presumptive Argument) A *presumptive argument* based on presumptive inference scheme description $\langle PD, \alpha, cd, \Psi, \Gamma, \xrightarrow{entails} \rangle$ is a tuple $\langle P, \tau, c \rangle$ where:

- $P \subseteq \mathcal{N}_I$ is a set of nodes denoting premises;
- $\tau \in \mathcal{N}_S^{RA}$ is a rule of inference application node;
- $c \in \mathcal{N}_I$ is a node denoting the conclusion;

such that:

- $\tau \xrightarrow{edge} c$; $\forall p \in P$ we have $p \xrightarrow{edge} \tau$;
- $\tau \xrightarrow{fulfilsScheme} \alpha$; $c \xrightarrow{fulfilsConclusionDesc} cd$; and
- $\xrightarrow{fulfilsPremiseDesc} \subseteq P \times PD$ corresponds to a one-to-one correspondence from P to PD.

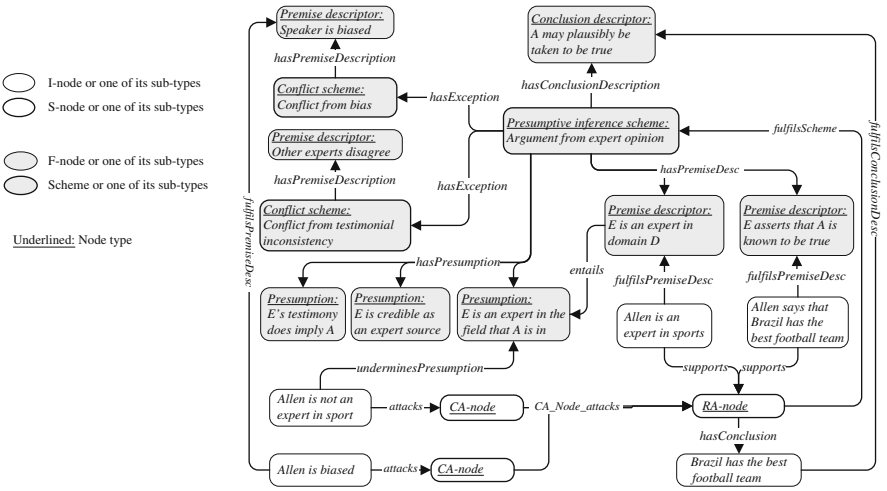

Fig. 19.2 An argument network showing an argument from expert opinion, two counter-arguments undermining a presumption and an exception, and the descriptions of the schemes used by the argument and attackers. *A:* Brazil has the best football team: Allen is a sports expert and he says so; *B:* But Allen is biased, and he is not an expert in sports!

3.2 Implementation in ArgDF

We implemented our extended ontology using RDF and RDFS [2], and call the resulting ontology AIF-RDF. In summary, we view elements of arguments and schemes (e.g. premises, conclusions) as RDF resources, and connect them using binary predicates as described earlier.

ArgDF[5] is a Semantic Web-based system that uses the AIF-RDF ontology. The Sesame RDF repository offers the central features needed by the system, namely: (i) uploading RDF and RDFS single statements or complete files; (ii) deleting RDF statements; (iii) querying the repository using the Semantic Web query language RQL; and (iv) returning RDF query results in a variety of computer processable formats including XML, HTML or RDF.

Creating New Arguments: The system presents the available schemes, and allows the user to choose the scheme to which the argument belongs. Details of the selected scheme are then retrieved from the repository, and the form of the argument is displayed to the user, who then creates the conclusion followed by the premises.

Support/Attack of Existing Expressions: The expressions (i.e. premises or conclusions) in the repository can be displayed, supported or attacked. When a user chooses to support an existing premise through a new argument/scheme, this premise will be both a premise in one argument, and a conclusion in another. Thus, the system enables argument *chaining*. If the user chooses to *attack* an expression, on the other hand, s/he will be redirected to choose an appropriate conflict scheme,

[5] ArgDF is a proof-of-concept prototype and can be accessed at: `http://www.argdf.org`

and create a new argument whose conclusion is linked to the existing conclusion via a conflict application node (as in Example 19.2).

Searching through Arguments: The system enables users to search existing arguments, by specifying text found in the premises or the conclusion, the type of relationship between these two (i.e. support or attack), and the scheme(s) used. For example, one can search for arguments, based on expert opinion, *against* the '*war on Iraq*,' and mentioning '*weapons of mass destruction*' in their premises. An RQL query is generated in the background.

Linking Existing Premises to a New Argument: While creating premises supporting a given conclusion through a new argument, the user can *re-use* existing premises from the system. This premise thus contributes to multiple arguments in a *divergent* structure. This functionality can be useful, for example, in Web-based applications that allow users to use existing Web content (e.g. a news article, a legal document) to support new or existing claims.

Attacking Arguments through Implicit Assumptions: With our account of presumptions and exceptions, it becomes possible to construct an automatic mechanism for *presuming*. ArgDF allows the user to inspect an existing argument, allowing the exploration of the hidden assumptions (i.e. presumptions and exceptions) by which its inference is warranted. This leads the way for possible implicit attacks on the argument through pointing out an exception, or through undermining one of its presumptions (as shown in Figure 19.2). This is exactly the role that Walton envisaged for his critical questions [22]. Thus, ArgDF exploits knowledge about implicit assumptions in order to enable richer interaction between the user and the arguments.

Creation of New Schemes: The user can create new schemes through the interface of ArgDF without having to modify the ontology. This feature enables a variety of user-created schemes to be incorporated, thus offering flexibility not found in any other argument-support system.

4 The AIF in Description Logic

In ArgDF, the actual arguments are specified by instantiating nodes, while actual schemes are created by instantiating the "scheme" class. Then, argument instances (and their constituent parts) are linked to scheme instances (and their part descriptors) in order to show what scheme the argument follows.

From the above, it is clear that ArgDF's reification of the AIF causes some redundancy at the instance level. Both arguments and schemes are described with explicit structure at the instance level. As a result, the property "*fulfilsScheme*" does not capture the fact that a S-node represents an instantiation of some generic *class of arguments* (i.e. scheme). Having such relationship expressed explicitly can enable reasoning about the classification of schemes.

In this section, we present another AIF-based ontology, 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 – of*".

We formalise the new ontology using Description Logics (DLs) [1], a family of logical formalisms that have initially been designed for the representation of conceptual knowledge in Artificial Intelligence. DL knowledge representation languages provide means for expressing knowledge about concepts composing a terminology (TBox), as well as knowledge about concrete facts (i.e. objects instantiating the concepts) which form a world description (ABox). Since Description Logics are provided with a formal syntax and formal model-theoretic semantics, sound and complete reasoning algorithms can be formulated. Our summary here of AIF in OWL-DL draws upon [15].

4.1 The ontology

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.

$Scheme \sqsubseteq Thing$ $Statement \sqsubseteq Thing$ $Author \sqsubseteq Thing$
$Author \sqsubseteq \neg Scheme$ $Author \sqsubseteq \neg Statement$ $Statement \sqsubseteq \neg Scheme$

As with the ArgDF reification of AIF, different specialisations of scheme are identified; for example the rule scheme (which describes the class of arguments), conflict scheme, preference scheme etc.

$RuleScheme \sqsubseteq Scheme$ $ConflictScheme \sqsubseteq Scheme$ $PreferenceScheme \sqsubseteq Scheme$

Each of these schemes can be further classified. For example, a rule scheme may be further specialised to capture deductive or presumptive arguments. The same can be done with different types of conflicts, preferences, and so on.

$DeductiveArgument \sqsubseteq RuleScheme$ $LogicalConflict \sqsubseteq ConflictScheme$
$InductiveArgument \sqsubseteq RuleScheme$ $PresumptivePreference \sqsubseteq PreferenceScheme$
$PresumptiveArgument \sqsubseteq RuleScheme$ $LogicalPreference \sqsubseteq PreferenceScheme$

A number of properties (or *roles* in DL terminology) are defined, which can be used to refer to additional information about instances of the ontology, such as authors of arguments, the creation date of a scheme, and so on. The domains and ranges of these properties are restricted appropriately and described below.

$\top \sqsubseteq \forall creationDate.Date$ $\top \sqsubseteq \forall creationDate^{-}.Scheme$
$\top \sqsubseteq \forall argTitle.String$ $\top \sqsubseteq \forall argTitle^{-}.RuleScheme$
$\top \sqsubseteq \forall authorName.String$ $\top \sqsubseteq \forall authorName^{-}.Author$
$Scheme \sqsubseteq \forall hasAuthor.Author$ $Scheme \sqsubseteq = 1 creationDate$
$RuleScheme \sqsubseteq = 1 argTitle$

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*. For example, at the highest level, we may classify statements as declarative, comparative or imperative, etc.

$DeclarativeStatement \sqsubseteq Statement$ $\quad ImperativeStatement \sqsubseteq Statement$
$ComparativeStatement \sqsubseteq Statement$ $\quad \ldots$

Actual statement instances have a property that describes their textual content.

$\top \sqsubseteq \forall claimText.String$ $\quad \top \sqsubseteq \forall claimText^{-}.Statement$

When defining a particular RuleScheme (i.e. class of arguments), we capture the relationship between each scheme and its components. Each argument has exactly one conclusion and at least one premise (which are, themselves, instances of class "Statement"). Furthermore, presumptive arguments may have assumptions and exceptions.

$RuleScheme \sqsubseteq \forall hasConclusion.Statement$ $\quad RuleScheme \sqsubseteq \geq 1 hasPremise$
$RuleScheme \sqsubseteq = 1 hasConclusion$ $\quad PresumptiveArgument \sqsubseteq \forall hasAssumption.Statement$
$RuleScheme \sqsubseteq \forall hasPremise.Statement$ $\quad PresumptiveArgument \sqsubseteq \forall hasException.Statement$

4.2 Example

With this in place, it becomes possible to further classify the above statement types to cater for a variety of schemes. For example, to capture the scheme for "Argument from Position to Know," the following classes of declarative statements need to be defined (each class is listed with its property `formDescription`[6] that describes its typical form).

$PositionToHaveKnowledgeStmnt \sqsubseteq DeclarativeStatement$

`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"

Now it is possible to fully describe the scheme for "Argument from Position to Know." Following are the necessary and sufficient conditions for an instance to be classified as an argument from position to know.

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

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

Other argument schemes (e.g. argument from analogy, argument from sign, etc.) can be defined in the same way.

[6] formDescription is an annotation property in OWL-DL. Annotation properties are used to add meta-data about classes.

4.3 Representing Conflicts Among Arguments

Conflict among arguments are captured through different specialisations of *ConflictScheme* such as *GeneralConflict* and *ExceptionConflict*.

$$ExceptionConflict \sqsubseteq ConflictScheme \qquad GeneralConflict \sqsubseteq ConflictScheme$$

GeneralConflict instances capture simple symmetric and asymmetric attacks among arguments while *ExceptionConflict* instances represent exceptions to rules of inference. The definition of *ConflictScheme* and *Statement* classes have been extended to include the appropriate restrictions on properties used to represent attacks among different arguments.

$$ConflictScheme \sqsubseteq \forall confAttacks.(Statement \sqcup RuleScheme) \qquad Statement \sqsubseteq \forall attacks.ConflictScheme$$
$$ConflictScheme \sqsubseteq \forall isAttacked.Statement \qquad Statement \sqsubseteq \forall confIsAttacked.ConflictScheme$$
$$ConflictScheme \sqsubseteq \forall underminesAssumption.Statement$$

Figures 19.3(a) to 19.3(d) illustrate how instances of *ConflictScheme* and the related properties are used to represent four different types of conflicts among arguments, namely, asymmetric attacks (a), symmetric attacks (b), undermining assumptions (c) and attacking by supporting existing exceptions (d).

In these figures, argument instances are denoted by Arg_n, premises are denoted by PX_n, conclusions by CX, assumptions by $AsmX_n$, exceptions by $ExcpX_n$ and instances of general conflict and exception conflict as GC_n and EC_1 respectively where X = {A, B, C, ...} and *n* represents the set of natural numbers {1,2,3,...}.

5 Reasoning over Argument Structures

In this section, we describe two ways in which the expressive power of Description Logic and its support for reasoning can be used to enhance user interaction with arguments. The features discussed here were implemented in a pilot system called *Avicenna*, utilizing the DL-compatible Web Ontology Language (OWL) .

5.1 Automatic Classification of Schemes and Arguments

In this section, we describe the general inference pattern behind classification of argument schemes (and their instances). This inference is based on the statement hierarchy and the conditions defined on each scheme. Two examples of this inference are also provided.

Consider two specialisations (sub-classes) of *PresumptiveArgument* : *PresScheme*1 and *PresScheme*2. An instance of the first scheme, *PresScheme*1, might have an instance of *CA* class as its conclusion and premises from classes $(PA_1, PA_2, ..., PA_n)$, where classes *CA* and $(PA_1, PA_2, ..., PA_n)$ are specialisations of the class *Statement*. Similarly, *PresScheme*2 has members of *CB* class as its

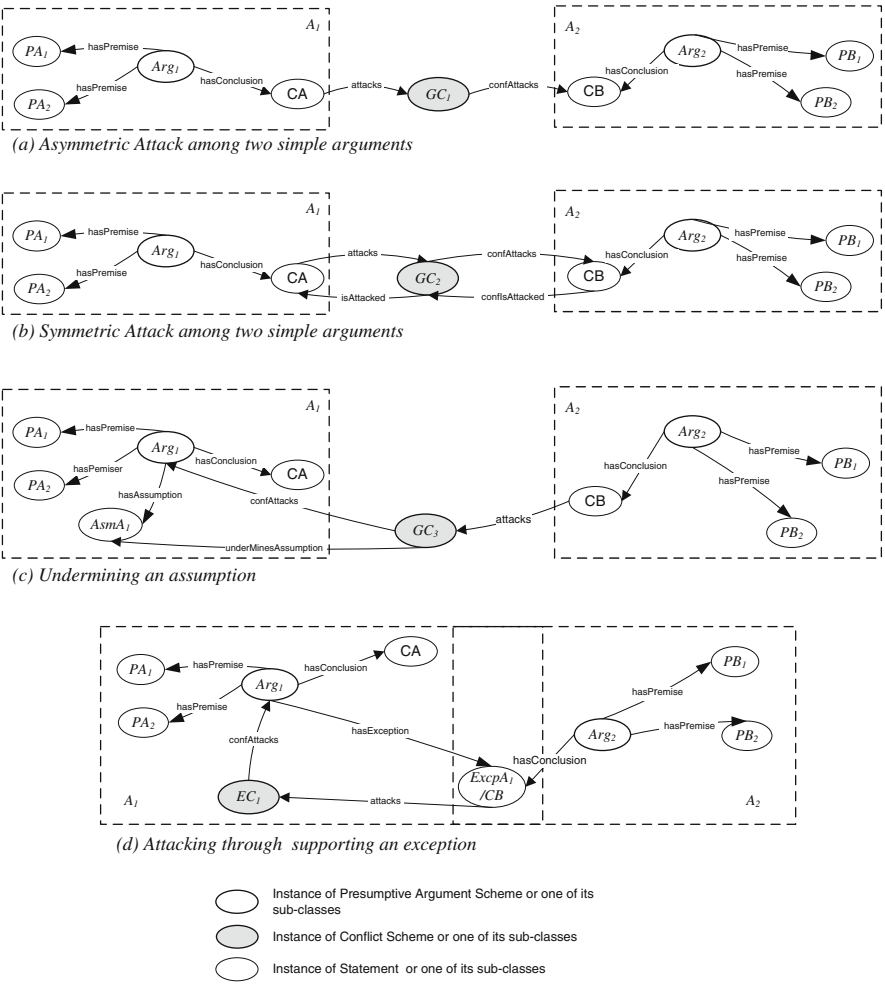

Fig. 19.3 Representation of different types of attack among arguments

conclusion and its premises are from classes $(PB_1, PB_2, ..., PB_m$ where CB and $(PB_1, PB_2, ..., PB_m)$ are specialisations of $Statement$ and $m >= n$. Let us assume that a relationship exists between CA and CB, that they are either referring to the same class or else that the latter is a specialisation of the former, i.e., $(CB \equiv CA) \vee (CB \sqsubseteq CA)$.

We also assume a relationship exists among the premises of these two schemes in a way that for every premise class of $PresScheme1$, there is a corresponding premise class in $PresScheme2$ that is either equal to or is a specialisation of the premise class in $PresScheme1$ (the opposite does not hold as we have allowed that $PresScheme2$ could have greater number of premises than $PresScheme1$), i.e. $\forall x \in 1, 2, ...m, \forall y \in 1, 2, ..., n, (PB_x \equiv PA_y) \vee (PB_x \sqsubseteq PA_y)$.

The necessary and sufficient conditions on *PresScheme*1 and *PresScheme*2 are defined as:

$$PresScheme1 \equiv (PresumptiveArgument \sqcap \exists hasConclusion.CA \sqcap \exists hasPremise.PA1 \sqcap \exists hasPremise.PA2 \sqcap \exists hasPremise.(...) \sqcap \exists hasPremise.PAn)$$

$$PresScheme2 \equiv (PresumptiveArgument \sqcap \exists hasConclusion.CB \sqcap \exists hasPremise.PB1 \sqcap \exists hasPremise.PB2 \sqcap \exists hasPremise.(...) \sqcap \exists hasPremise.PBm)$$

Considering the statement hierarchy and the necessary and sufficient conditions defined on each class, *PresScheme*2 is inferred by the description logic reasoner as the sub-class of *PresScheme*1 in case the number of premises in *PresScheme*2 is greater than number of premises in *PresScheme*1 (i.e. $m > n$). In case the number of premises are the same (i.e $m = n$), and at least one of the premises of *PresScheme*2 is a specialisation of a premise in *PresScheme*1 and/or the conclusion *CB* is a specialisation of *CA*, *PresScheme*2 is also inferred as the sub-class of *PresScheme*1.

Following the above explanation, due to the hierarchy of specialisation among different descriptors of scheme components (statements) as well as the necessary and sufficient conditions defined on each scheme, it is possible to infer the classification hierarchy among schemes.

An interesting example is offered by the specialisation relationship that can be inferred between "Fear Appeal Argument" and "Argument from Negative Consequences".

Scheme 1 ***Argument From Negative Consequences***

- Premise: *If A is brought about, bad consequences will plausibly occur.*
- Conclusion: *A should not be brought about.*
- ***Critical Questions***

  1. *How strong is the probability or plausibility that these cited consequences will (may, might, must) occur?*
  2. *What evidence, if any, supported the claim that these consequences will (may, might, must) occur if A is brought about?*
  3. *Are there consequences of the opposite value that ought to be taken into account?*

Scheme 2 ***Fear Appeal Argument***

- Fearful situation premise: *Here is a situation that is fearful to you.*
- Conditional premise: *If you carry out A, then the negative consequences portrayed in this fearful situation will happen to you.*
- Conclusion: *You should not carry out A.*
- ***Critical Questions***

  1. *Should the situation represented really be fearful to me, or is it an irrational fear that is appealed to?*
  2. *If I don't carry out A, will that stop the negative consequences from happening?*

3. If I do carry out A, how likely is it that the negative consequences will happen?

The necessary and sufficient conditions of the "Argument from Negative Consequences" are detailed as:

$$ArgNegatvieConseq \equiv (PresumptiveArgument \sqcap \exists hasConclusion.ForbiddenActionStmnt \sqcap \exists hasPremise.BadConsequenceStmnt)$$

Likewise, for "Fear Appeal Argument":

$$FearAppealArg \equiv (PresumptiveArgument \sqcap \exists hasConclusion.ForbiddenActionStmnt \sqcap \exists hasPremise.FearfulSituationStmnt \sqcap \exists hasPremise.FearedBadConsequenceStmnt)$$

The statements are defined below. Note that the "Feared Bad Consequence" statement is a specialization of "Bad Consequence" statement, since it limits the bad consequence to those portrayed in the fearful situation.

BadConsequenceStmnt $\sqsubseteq$ *DeclarativeStatement*

`formDescription` : "If A is brought about, bad consequences will plausibly occur"

ForbiddenActionStmnt $\sqsubseteq$ *DeclarativeStatement*

`formDescription` : "A should not be brought about"

FearfulSituationStmnt $\sqsubseteq$ *DeclarativeStatement*

`formDescription` : "Here is a situation that is fearful to you"

FearedBadConsequenceStmnt $\sqsubseteq$ *BadConsequenceStmnt*

`formDescription` : "If you carry out A, then the negative consequences portrayed in this fearful situation will happen to you"

As a result of classification of schemes into hierarchies, instances belonging to a certain scheme class will also be inferred to belong to all its super-classes. For example, if the user queries to return all instances of "Argument from Negative Consequences," the instances of all specializations of the scheme, such as all argument instances from "Fear Appeal Arguments" are also returned.

5.2 Inferring Critical Questions

In this section we describe the general inference pattern behind inference of critical questions from an argumentation scheme's super-classes and provide an example.

In the previous section we described an assumption about two specialisations of *PresumptiveArgument*, *PresScheme*1 and *PresScheme*2 and the fact that *PresScheme*2 was inferred to be the sub-class of *PresScheme*1. Each of these schemes might have different assumptions and exceptions defined on their classes. For example, *PresScheme*1 has *AsmA*1 and *AsmA*2 as its assumptions and *ExcA*1 as its exception. *PresScheme*2 has *AsmB*1 and *ExcB*1 as its assumption and exception respectively. *AsmA*1, *AsmA*2, *AsmB*1, *ExcA*1 and *ExcB*1 are specialisations of *Statement* class. The the necessary conditions defined on classes *PresScheme*1 and *PresScheme*2 are:

$$PresScheme1 \sqsubseteq \exists hasAssumption.AsmA1$$
$$PresScheme1 \sqsubseteq \exists hasAssumption.AsmA2$$
$$PresScheme1 \sqsubseteq \exists hasException.ExcA1$$

$PresScheme2 \sqsubseteq \exists hasAssumption.AsmB1$
$PresScheme2 \sqsubseteq \exists hasException.ExcB1$

Since *PresScheme*2 has been inferred by the reasoner as the specialization (sub-class) of *PresScheme*1, a query to the system to return all assumptions and exceptions of *PresScheme*2, is able to return all those explicitly defined on the scheme class (i.e. *AsmB*1 and *ExcB*1) as well as those defined on any of its super-classes (in this case: *AsmA*1, *AsmA*2 and *ExcA*1).

Since the schemes are classified by the reasoner into a hierarchy, if certain assumptions or exceptions are not explicitly stated for a specific scheme but are defined on any of its super-classes, the system is able to infer and add those assumptions and exceptions to instances of that specific scheme class. Since critical questions enable evaluation of an argument, inferring additional questions for each scheme will enhance the analysis process.

Consider the critical questions for "Fear Appeal Argument" and "Argument from Negative Consequences" given in the previous section. These critical questions are represented in the ontology through the following statements:

$IrrationalFearAppealStmnt \sqsubseteq DeclarativeStatement$
`formDescription` : "It is an irrational fear that is appealed to"
$PreventionOfBadConsequenceStmnt \sqsubseteq DeclarativeStatement$
`formDescription` : "If A is not carried out, this will stop the negative consequences from happening"
$OppositeConsequencesStmnt \sqsubseteq DeclarativeStatement$
`formDescription` : "There are consequences of the opposite value that ought to be taken into account"
$StrongConsequenceProbabilityStmnt \sqsubseteq DeclarativeStatement$
`formDescription` : "There is a strong probability that the cited consequences will occur"
$ConsequenceBackUpEvidenceStmnt \sqsubseteq DeclarativeStatement$
`formDescription` : "There is evidence that supports the claim that these consequences will occur if A is brought about."

The necessary conditions on "Argument from Negative Consequences" that define these critical questions are:

$ArgNegatvieConseq \sqsubseteq \exists hasException.OppositeConsequencesStmnt$
$ArgNegatvieConseq \sqsubseteq \exists hasAssumption.StrongConsequenceProbabilityStmnt$
$ArgNegatvieConseq \sqsubseteq \exists hasAssumption.ConsequenceBackUpEvidenceStmnt$

Likewise, the necessary conditions on "Fear Appeal Argument" are:

$FearAppealArg \sqsubseteq \exists hasException.IrrationalFearAppealStmnt$
$FearAppealArg \sqsubseteq \exists hasAssumption.PreventionOfBadConsequenceStmnt$
$FearAppealArg \sqsubseteq \exists hasAssumption.StrongConsequenceProbabilityStmnt$

"Fear Appeal Argument" is classified as a sub-class of "Argument from Negative Consequences." The critical questions 2 and 3 of "Argument from Negative Consequences" have not been explicitly defined on "Fear Appeal Argument", but can be inferred through reasoning.

6 Current Issues and Future Directions

The AIF will come into its own as it demonstrates that it can be used to build bridges between applications, and, perhaps, between theories. Work with the Araucaria diagramming tool [18] has demonstrated (at least in the specific area of linguistic analysis) how carefully designed representation can support analysts working in different traditions, and to a certain extent can help reuse across domains. At the time of writing, Araucaria has in the region of 10,000 users. Some few of these submit analyses using a number of different analytical techniques (Toulmin schema, argumentation schemes, Wigmore charts, etc.) to a centralised corpus. Though there are analysed arguments from an enormous range of domains, one that is particularly interesting is the legal domain. Wigmore charts were designed specifically for analysis of cases and are rarely used in other domains. The Toulmin-schema is rooted in legal analysis though is now much more widely used. Walton's approach to argumentation schemes has generic application, though one that encompasses use in law [23]. These various degrees of specificity to the legal domain counterbalance the number of analysts working in each tradition (relatively few using Wigmore, many more using Walton argumentation schemes). As a result, the part of the corpus that might be said to encompass examples from the legal domain has a theoretically diverse basis. Despite this diversity, that part of the corpus has been successfully used in an unrelated project investigating discourse markers in legal argumentation [13]. What has made this possible is the underlying unifying representation.

This example shows in microcosm what the AIF is trying to do right across computational uses of argument. Though it is still early days, there are a number of systems, tools and techniques that are working, planning or considering implementation to support AIF. We provide a brief overview of a number of them here to give an indication of the range of potential applications, and the types of role that AIF might play.

Argkit and Dungine. Argkit[7] is designed to be a reusable, plug and play codebase for developing and linking together applications that use argument, and particularly those that have a requirement for processing abstract argument [20]. In a compelling demonstration, South has shown how the Dungine component, which performs computations according to Dung acceptability semantics, can be connected to Araucaria to compute acceptability of real arguments on the fly. Though there are theoretical challenges with connecting models of abstract and concrete argumentation, this proof of concept demonstrator shows how the two areas of research might be harmonised. Argkit plans have scheduled integration of AIF as a way to support such integration more broadly across other sources of both concrete and abstract argumentation.

Araucaria. The analysis tool, Araucaria [18], has a large user base, but is now suffering from limitations of its underlying representation and increasingly dated interface and interaction metaphors. A large-scale rewrite is underway, which provides

[7] http://www.argkit.org

AIF support: an early alpha is available with reusable code modules for processing AIF resources.

Rationale. Rationale [21] is a highly polished commercial product for argument visualisation in a primarily educational context, which has been recently complemented by a new product from the same company, Austhink Software Pty Ltd[8], providing related functionality targeted at a commercial context. Rationale, in particular, has explored interaction with resources that Araucaria produces and vice versa. From Austhink's commercial point of view, the cost of developing an AIF component (even if low) would need to be offset against value; that value will only be clear when there is a critical mass of other systems and environments that can work with AIF. This is perhaps an inevitable part of the relationship between academic and commercial sides of research in argumentation.

Compendium. Compendium is similar to Rationale in a number of ways in that it is a polished tool with its origins in a research programme but now mature with a wide user-bases supported by the Compendium Institute[9], run by the Open University. Compendium focuses not just on arguments, but on a wide range of semantic types (issues, decisions, etc.). It has been used with other tools, such as Araucaria, which can provide embedded support for building the fine-grained structure of arguments which form components of Compendium maps. Compendium has also made use of Araucaria's argumentation scheme representation, by automatically importing the various "*schemesets*" that capture the definitions of schemes offered by various authors. The import makes those same definitions available as templates to Compendium users[10]. Richer integration with more detailed models of these schemes would be made possible by AIF import along these same lines.

Cohere. Cohere is an ambitious project that aims to bring Compendium-like flexibility in 'sense-making' to a broad online audience. Though it supports a very broad range of semantic relationships between componets, it has a particular focus on those that might be considered argumentative. As a part of its mission to be "an idea management tool"[11], Buckingham Shum states that, "A key priority is to provide Argument Interchange Format compatibility" [3] to provide smooth interoperability with both other tools in the space (such as those listed here) and also to provide Cohere with structured access to additional argument resources.

Carneades. The Carneades system is both a framework for reasoning about arguments and a system that implements that framework. Carneades is sited squarely within an AI & Law context [9], and has already worked to integrate with the Legal Knowledge Interchange Format, LKIF. At its core, however, lies argumentation-based representation and reasoning at both concrete and abstract levels (though, interestingly in regards to the latter not using Dung's popular approach [7]). As a result, the Carneades work is exploring the possibility of using AIF as a mechanism for exporting and importing argument structures from other systems.

---

[8] http://www.austhink.com

[9] http://compendium.open.ac.uk/institute

[10] http://compendium.open.ac.uk/compendium-arg-schemes.html

[11] http://kmi.open.ac.uk/technologies/cohere

ArguGRID. ArguGRID is a large EU project funded under FP7. Though its goals cover a broad spectrum of activity, models of argumentation lie at its centre. The project aims to use argumentation to provide semantically rich processes for negotiating services across grid networks – see e.g. [11]. Drawing heavily upon abstract argumentation models, it needs AIF to develop far enough that it provides strong support for abstract argumentation before AIF can play an important role. In the meantime, AIF remains on the roadmap of development for ArguGRID systems.

AAC. The Arguing Agents Competition is a new collaboration that aims to provide an open, competitive environment in which agents can compete in their ability to argue successfully [24]. It aims to be similar in spirit to the leading example set by the Trading Agent Competition[12]. As the AAC is currently under development, it has been designed to use AIF from the outset. This represents the first significant test case for AIF's suitability for autonomous reasoning (as opposed to human-in-the-loop processing).

InterLoc. Interloc is an online educational environment for structuring pedagogic discourse and debate [17]. Its rich representations of argument, and strongly typed dialogue games are well suited to what AIF can offer. Initial explorations are under way to explore potential uses of AIF in the Interloc project, but there is a significant challenge: Interloc focuses heavily upon the design and execution of a number of sophisticated and intricate dialogue games for structuring interactions online. AIF in its simplest form, presented here, does not support dialogue at all.

6.1 Dialogue in AIF

Though discussed at the initial AgentLink meeting in Budapest, argumentation dialogue was deprioritised against the more basic requirement of being able to represent "monologue": it is first necessary to be able to handle relatively static knowledge structures (such as those represented in abstract argumentation frameworks) before going on to tackle how those structures are updated and modified. The problem is that very many systems and use cases for the AIF involve dialogue.

The first step in introducing dialogue into the AIF is presented in [12], which shows how protocol application steps can be introduced into the AIF framework. [19] goes on to show how both dialogue game descriptions, and the actual dialogues they govern can be represented in a common extension to AIF, called AIF^{+}. The aim is to allow specifications of dialogue games to be represented in a way that is analogous to argument scheme representations, and to allow instantiated dialogues to be analagous to argument instantiations. The challenge lies in ensuring that the instantiations of dialogues are connected in appropriate ways to instantiations of arguments, according to the rules of the appropriate dialogue game. Both [12] and [19] are, however, rather preliminary, and many issues remain to be resolved.

[12] http://www.sics.se/tac

7 Conclusion

The current AIF specification and its reifications mark a starting point. As experience with AIF grows, and different systems and research programmes make call upon it, the specification will inevitably shift to accommodate the broadening demands. Extensions to handle dialogue represent and early example of this broadening – albeit one that has been anticipated from the outset.

Unfortunately, this shifting poses two distinct problems: one in the short term, and one in the longer term. The first problem is one of bootstrapping. With a number of teams working to implement slightly different reifications of the AIF, tracking versions to ensure at least some compatibility is becoming tricky. To some extent, the core of AIF is stabilising, and as reference implementations become available, code reuse will become more common, and compatibility will be improved. The second, related problem, concerns the process of solidification by which the AIF settles into stability. It is important that this solidification not happen too early: the AIF must support the theories and systems that are being developed right across the community. But on the other hand, it must also not happen to late, or we risk fragmentation and a loss of coherence, which is the raison d'être of the specification. Currently, the balance is being successfully struck informally through personal networks and regular communication. If AIF starts to scale, a control system that is less lightweight may be necessary to maintain stability of at least a common core.

Finally, [16] present a vision of a World Wide Argument Web of interconnected arguments and debates, founded upon the AIF. Though it leaves many questions unanswered it does present a challenging goal for development not only of AIF components but also large-scale infrastructure for supporting online argumentation. The WWAW vision has the potential to draw together a number of different initiatives in the space allowing each to find a wider audience and more practical utility than they might individually, which would be a true measure of success for the AIF.

Acknowledgements The authors are grateful to Steve Willmott, Peter McBurney, and AgentLink III for initiating and organising the Technical Forum "Towards a Standard Agent-to-Agent Argumentation Interchange Format," and to all those who contributed to the initial AIF specification that it produced.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, Cambridge, UK, 2003.
2. D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation REC-rdf-schema-20040210, World Wide Web Consortium (W3C), February 2004.
3. S. Buckingham Shum. Cohere: Towards Web 2.0 argumentation. In P. Besnard, S. Doutre, and A. Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, pages 97–108. IOS Press, Amsterdam, The Netherlands, 2008.

4. S. Buckingham Shum, E. Motta, and J. Domingue. ScholOnto: An ontology-based digital library server for research documents and discourse. *International Journal of Digital Libraries*, 3(3):237–248, 2000.
5. S. Buckingham Shum, V. Uren, G. Li, B. Sereno, and C. Mancini. Modelling naturalistic argumentation in research literatures: Representation and interaction design issues. *International Journal of Intelligent Systems, Special Issue on Computational Modelling of Naturalistic Argumentation*, 22(1):17–47, 2007.
6. C. I. Chesñevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott. Towards an argument interchange format. *The Knowledge Engineering Review*, 21(4):293–316, 2007.
7. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.
8. L. Emmet and G. Cleland. Graphical notations, narratives and persuasion: a pliant systems approach to hypertext tool design. In *HYPERTEXT 2002, Proceedings of the 13th ACM Conference on Hypertext and Hypermedia, June 11-15, 2002, University of Maryland, College Park, MD, USA*, pages 55–64, New York, USA, 2002. ACM Press.
9. T. F. Gordon, H. Prakken, and D. Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15):875–896, 2007.
10. 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.
11. P.-A. Matt, F. Toni, T. Stournaras, and D. Dimitrelos. Argumentation-based agents for eprocurement. In *AAMAS '08*, pages 71–74, 2008.
12. S. Modgil and J. McGinnis. Towards characterising argumentation based dialogue in the argument interchange format. In I. Rahwan and P. Moraitis, editors, *Proceedings of the 4th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS)*, volume 5384 of *Lecture Notes in Computer Science*. Springer Verlag, 2008. to appear.
13. M. F. Moens, E. Boiy, R. M. Palau, and C. Reed. Automatic detection of arguments in legal texts. In *Proceedings of the International Conference on AI & Law (ICAIL-2007)*, 2007.
14. I. Rahwan. Mass argumentation and the Semantic Web. *Journal of Web Semantics*, 6(1):29–37, 2008.
15. I. Rahwan and B. Banihashemi. Arguments in OWL: A progress report. In P. Besnard, S. Doutre, and A. Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, pages 297–310, Amsterdam, Nethrelands, 2008. IOS Press.
16. I. Rahwan, F. Zablith, and C. Reed. Laying the foundations for a world wide argument web. *Artificial Intelligence*, 171(10–15):897–921, 2007.
17. A. Ravenscroft. Promoting thinking and conceptual change with digital dialogue games. *Journal of Computer Assisted Learning*, 23(6):453–465, 2007.
18. C. Reed and G. Rowe. Araucaria: Software for argument analysis. *International Journal of AI Tools*, 14(3–4):961–980, 2004.
19. C. Reed, S. Wells, J. Devereux, and G. Rowe. AIF+: Dialogue in the Argument Interchange Format. In P. Besnard, S. Doutre, and A. Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, pages 311–323. IOS Press, Amsterdam, The Netherlands, 2008.
20. M. South, G. Vreeswijk, and J. Fox. Dungine: A Java Dung reasoner. In P. Besnard, S. Doutre, and A. Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, pages 360–368, Amsterdam, Nethrelands, 2008. IOS Press.
21. T. van Gelder. The rationale for rationale. *Law, Probability and Risk*, 6(1–4):23–42, 2007.
22. D. Walton. *Argumentation Schemes for Presumptive Reasoning*. Erlbaum, Mahwah NJ, 1996.
23. D. Walton. *Legal Argumentation and Evidence*. Penn State Press, University Park, PA, 2002.
24. T. Yuan, J. Schulze, J. D. C., and Reed. Towards an arguing agents competition: Building on Argumento. In *Working Notes of the 8th Workshop on Computational Models of Natural Argument (CMNA-2008)*, 2008.