The Knowledge Engineering Review, Vol. 26:4, 487–511. © Cambridge University Press, 2011
doi:10.1017/S0269888911000191

Representing and classifying arguments on the Semantic Web

IYAD RAHWAN[1,2], BITA BANIHASHEMI[3], CHRIS REED[4], DOUGLAS WALTON[5] and SHERIEF ABDALLAH[3,6]

[1]*Computing & Information Science, Masdar Institute of Science and Technology, Abu Dhabi, UAE;*
[2]*Technology & Development Program, Massachusetts Institute of Technology, Cambridge, MA, USA;*
e-mail: irahwan@masdar.ac.ae;
[3]*Faculty of Engineering & IT, British University in Dubai, Dubai, UAE;*
e-mail: bita@emirates.net.ae;
[4]*School of Computing, University of Dundee, Dundee, UK;*
e-mail: creed@computing.dundee.ac.uk;
[5]*Centre for Research in Reasoning, Argumentation & Rhetoric, University of Windsor, Windsor, Ontario, Canada;*
e-mail: dwalton@uwindsor.ca;
[6]*School of Informatics, University of Edinburgh, Edinburgh, UK;*
e-mail: sherief.abdallah@buid.ac.ae

Abstract

Until recently, little work has been dedicated to the representation and interchange of informal, semi-structured arguments of the type found in natural language prose and dialogue. To redress this, the research community recently initiated work towards an Argument Interchange Format (AIF). The AIF aims to facilitate the exchange of semi-structured arguments among different argument analysis and argumentation-support tools. In this paper, we present a Description Logic ontology for annotating arguments, based on a new reification of the AIF and founded in Walton's theory of argumentation schemes. We demonstrate how this ontology enables a new kind of automated reasoning over argument structures, which complements classical reasoning about argument acceptability. In particular, Web Ontology Language reasoning enables significantly enhanced querying of arguments through automatic scheme classifications, instance classification, inference of indirect support in chained argument structures, and inference of critical questions. We present the implementation of a pilot Web-based system for authoring and querying argument structures using the proposed ontology.

1 Introduction

Arguments are presented every day on the Web, in discussion forums, blogs, news sites, etc. As such, the Web acts as an enabler of large-scale argumentation, where different views are presented, challenged, and evaluated by contributors and readers. However, these methods do not capture the explicit structure of argumentative viewpoints. This makes the task of evaluating, comparing and identifying the relationships among arguments difficult.

Recently, various software tools have been developed to assist users in analyzing textual arguments (e.g. *Araucaria*; Reed & Rowe, 2004), for helping users construct well-organized textual arguments (e.g. *Parmenides*; Atkinson *et al*., 2006), and for organizing debates on the Web (e.g. Cohere; Shum, 2008; *Truthmapping*[1], *Debatepedia*[2]). However, little integration exists between such systems, mainly due to the lack of a unified ontology for describing argument structures.

[1] http://www.truthmapping.com
[2] http://wiki.idebate.org/index.php/

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

Recently, the 'computational modelling of argument' research community initiated work towards an Argument Interchange Format (AIF; Chesňevar *et al.*, 2006). The AIF aims to facilitate the exchange of semi-structured arguments among different argument analysis and argumentation-support tools.

To demonstrate how the AIF recommendations can be implemented concretely, we recently proposed the first AIF-based ontology (Rahwan *et al.*, 2007) based on Resource Description Framework Schema (RDFS; Brickley & Guha, 2004). A pilot system named ArgDF was implemented, through which users can create arguments using different schemes and can query arguments using a Semantic Web query language. Users can also attack or support parts of existing arguments, or use existing parts of an argument in the creation of new arguments. As such, ArgDF is an open platform not only for representing arguments, but also for building interlinked and dynamic argument networks on the Semantic Web. This initial public-domain tool was intended to seed what it is hoped will become a rich suite of sophisticated applications for authoring, linking, navigating, searching, and evaluating arguments on the Web.

In this paper[3], we extend our previous work on supporting argumentation on the Semantic Web. In particular, the paper advances the state of the art in computational modelling of argumentation in three ways. First, it presents the first ontology of argumentation schemes in a Description Logic (DL; Baader *et al.*, 2003) using as a starting point the AIF specification (Chesňevar *et al.*, 2006). To our knowledge, this ontology is the most expressive formal ontology of argument, based on Walton's general theory of argumentation schemes (Walton, 1996).

The second main contribution of this paper is in showing, for the first time, how DL inference techniques can be used to reason about arguments, ranging from automatic argument classification to reasoning about chained argument structures. To our knowledge, this kind of reasoning has never been applied to argumentation schemes, and provides a complement to the classical reasoning about argument acceptability (Baroni & Giacomin, 2007).

The third main contribution of this paper is providing the first implementation of an OWL (Web Ontology Language)-based system for argumentation support on the Semantic Web. Using OWL enables significantly enhanced querying and interaction with argument structures. These aspects are brought together in a Web-based pilot system called *Avicenna*.

We emphasize that our aim here was not to present an extensive and complete ontology of argumentation schemes. This task is beyond the scope of any single paper, and is a topic under development in its own right (Walton, 1996; Walton *et al.*, 2008). Instead, our aim is to show how a new specification of the AIF top-level ontology enables some new kinds of automated reasoning over schemes, once these are specialized further.

2 Mass argumentation tools on Web 2.0

There are a number of Web 2.0 applications designed to support large-scale argumentation on the World Wide Web[4]. In this section, we discuss some of these tools and provide a concise assessment.

2.1 Existing tools

Debatepedia[5] synthesizes two structuring elements: a 'wiki' technology and a 'logic tree' debate methodology. The wiki enables collaborative content management on the World Wide Web by users (a significant feature of Wikis is the ease with which pages are created, edited, and linked). The logic tree is a pro/con hierarchy, in which the main debate topic is located at the root.

[3] This paper is a revised and extended version of a paper that appeared in the proceedings of COMMA 2008 (Rahwan & Banihashemi, 2008).

[4] Web 2.0 refers to the second phase of applications on the World Wide Web. Though ill-defined, it is typically taken to focus on user contribution and collaboration by tagging data (e.g. Social bookmarking), editing data (e.g. Wikis), mass publishing (e.g. Weblogs), and Web feeds (e.g. RSS).

[5] `http://wiki.idebate.org/index.php/`

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

Arguments are added in free text format and do not follow any specific structure. Users support (or attack) arguments by adding their own arguments under pro/yes (or con/no) sections of each question. Arguments can be re-used across debates; however, each debate tree is treated in isolation. Basic keyword search is provided.

Debategraph[6] adds more structure to arguments by using elements representing different argumentative constructs (e.g. *issues* open to debate, the *positions* taken, *arguments* attacking or supporting these positions and *repertoire* of possible measures and alternative policies). It provides a visual tool for visualizing and navigating a tree structure of these elements, and has been well-publicised recently by launching discussions through popular media such as the BBC[7] and the White House[8]. While the popularity of this tool is a very positive development, it does not provide a distinction among different types of arguments.

Cohere[9] (Shum, 2008) is another Web-based argumentation tool that is intended to allow students and researchers to make personal and collective sense of problems. Users can create (or re-use) *Ideas* and link them by means of different *Connection* types. All connections are broadly classified as positive or negative. Ideas play one or more *Roles* within a given association. Both connections and roles can be extended by users. Cohere offers an attractive visualization that supports browsing and searching of an argument network.

Truthmapping[10] is a public argumentation support system, which exhibits an advanced argument structure; it distinguishes between premises and conclusions of an argument. Users can agree with or attack existing arguments (via critiques) and the creator of the argument can add a single rebuttal to each critique. Arguments can be chained (although supporting claims is restricted to the same team members) and can contain hyperlinks to other Websites or to premises or conclusions of other arguments. A state map visually summarizes the overall user rating of different parts of an argument. Basic keyword search is provided over categories and topics. Truthmapping distinguishes between deductive and inductive types of arguments.

Parmenides (Atkinson *et al*., 2006) is an example of highly structured argument-based deliberation support systems (ADSS). Parmenides is based on a formal model of argumentation and a specific inference scheme for justifying and critically evaluating the adoption of an action. After a position is put forward, the system provides a forms-based, questionnaire interface to obtain views from the user, and a fixed set of possible attacks that can be made. Once the original position has been subjected to their critique, another sequence of forms enables them to propose positions of their own; again in a way which will lead them to construct their position in the same form of the argumentation scheme. A main drawback of Parmendies, similar to most ADSS, is that it is aimed for a specific domain (in this case, reasoning about action). Thus, it uses a particular scheme for arguing about action, along with its associated critical questions.

2.2 Assessing Web 2.0 tools

Many existing Web 2.0 systems tackle the issue of capturing some structural attributes of arguments, as well as the details of interactions among arguments. By doing so, they not only facilitate evaluation and search of arguments, but also enable far better visualization and navigation of arguments by users or automated tools. Moreover, such structure improves groups' abilities to reach consensus and make higher quality decisions (Farnham *et al*., 2000). It could also simplify the automated support for the argumentation process (e.g. discovering inconsistencies or synergies among disputants; Rahwan *et al*., 2007).

But current implementations suffer from a number of limitations. Firstly, most of them sacrifice either structure or scalability. Highly structured systems (such as Parmenides) are intended for

[6] http://debategraph.org/

[7] http://news.bbc.co.uk/1/hi/technology/7827112.stm

[8] http://www.whitehouse.gov/blog/Open-Government-Brainstorm-Collaboration-in-Action/

[9] http://cohere.open.ac.uk

[10] http://www.truthmapping.com

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

smaller domains and are based on specific reasoning patterns instead of general theories of argumentation, while highly scalable systems (such as Debatepedia) present very simple structures of arguments and argument networks, limiting automated querying and analysis of argument repositories.

Another drawback of Web 2.0 argumentation systems such as Truthmapping is that while arguments are structured and possibly hyper-linked, these links carry no explicit semantics. This limitation hinders the possibility of using meta-data about arguments to enhance the automated search and evaluation.

Lastly, current tools are domain-, audience- and task-specific, with no common ontology, making it impossible to provide robust services (such as query answering) that utilize arguments from multiple mass argumentation systems (Rahwan *et al.*, 2007; Rahwan, 2008).

2.3 Argumentation and the Semantic Web

The key feature of Semantic Web technologies is that they represent Web information in standard, machine-processable formats. In the current context, semantic markup enables us to explicitly annotate arguments and their different components in order to process (or reason with) those annotations.

Semantic Web technologies can present a solution to the integration among mass argumentation tools through two key features. Firstly, a unified argument description ontology could act as an inter-lingua between the different tools and resources. Secondly, if a standard ontology of arguments cannot be achieved, then ontology mapping tools (Kalfoglou & Schorlemmer, 2003) can potentially provide means for the automatic translation of a variety of argument annotation languages (Rahwan, 2008).

Extended markup language (XML), located at the lower layer of the Semantic Web technologies, introduces structure and syntactic interoperability. The provided structure can be made machine-accessible through DTDs and XML Schema. An XML interchange language called the 'Argument Markup Language' (AML) has been proposed for structuring arguments by annotating premises and conclusions in XML. AML is used in Araucaria (Reed & Rowe, 2004) which is a stand-alone application tool for analyzing and diagramming arguments in which arguments can be authored (diagrammed) using alternative sets of argumentation schemes such as those provided by Walton (1996), Perelman and Olbrechts-Tyteca (1969), and Katzav and Reed (2004). This tool also provides the facility to design one's own argumentation schemes. Once arguments have been analyzed, they can be uploaded to AraucariaDB, which is an online repository of arguments. It provides a search engine[11], which allows advanced searches based on combination of different parameters such as argumentation schemes, argument creation date range, argument analyst or source, etc. Araucaria enables search over online argument repositories using XPath queries.

XML-based AMLs share a limitation: standard XML does not provide any means of talking about the meaning of the data. As a consequence, the semantics of arguments specified in these languages is tightly coupled with particular schemes to be interpreted in a specific tool and according to a specific underlying theory.

Unlike XML, Semantic Web ontology languages such as RDF Schema (RDFS; Brickley & Guha, 2004) and OWL (McGuinness & van Harmelen, 2004) can offer a unified ontology for describing and annotating arguments as they contain machine-processable *semantics*. RDFS is a vocabulary for describing properties and classes of RDF-based resources, with semantics for hierarchies of such properties and classes. OWL adds more vocabulary for describing properties and classes to RDFS and enables reasoning about asserted concepts to infer new concepts.

DiscourseDB[12] is an argumentation system based on Semantic Wiki (Völkel *et al.*, 2006) technology. This system collects the opinions of the world's journalists and commentators about ongoing political events and issues. Using this tool, users can post arguments and other users can

[11] http://araucaria.computing.dundee.ac.uk/doku.php

[12] http://discoursedb.org/wiki/Main_Page

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

have 'for', 'against' or 'mixed' positions on those arguments. It provides the facility to export content into OWL/RDF format for use by other Semantic Web applications.

In addition to simpler keyword searches, this system offers a semantic search module in which users can use Semantic Media Wiki's query language to write queries. The only type of inference available over the argument network is sub-class hierarchy among topics (e.g. a query to list instances of a specific topic theme returns instances under all the topics that are sub-classes of that specific topic theme). Another drawback with DiscourseDB is that the arguments posted are in free format text and do not follow any specific structures or adhere to any explicit schemes. Moreover, it is not possible to form complex structures of multiple inter-connected arguments (e.g. convergent, divergent, etc.). Capturing argument structures as well as the details of different interactions among arguments is essential for evaluation and querying of arguments by users or automated tools.

In our earlier work, we presented ArgDF (Rahwan *et al.*, 2007), a pilot system based on an RDFS ontology that models the AIF specification (Chesňevar *et al.*, 2006) and extends it to include Walton's account of argumentation schemes (Walton, 1996; see Section 3 for a brief overview). In ArgDF, users can author new arguments that adhere to any of the available schemes; they can also attack or support existing arguments (although the process of support is constrained in some ways). Users can also extend the underlying ontology by adding new argumentation schemes; the new schemes are added as new instances of the scheme-related concepts (classes) in the ontology. A semantic-based keyword search facility is also offered by the system that returns the supporting/attacking arguments of a claim containing a specific keyword.

ArgDF implements an interchange format and is based on open standards, and therefore resolves many problems related to current Web-based argumentation systems. However, the underlying ontology of ArgDF suffers from number of limitations (discussed below), both in terms of design specification and the ontology language used.

A core argumentation ontology developed in OWL is reported by Verheij (2005). He suggests that each argumentation format should use the argumentation core ontology as its starting point and provide a translation back into the core ontology. In this case, translations between argumentation formats are optional and can be developed whenever considered useful. The core ontology is meant to provide the glue. At the time of writing, no Web-based system has been reported that utilizes this ontology.

3 The World Wide Argument Web

Motivated by the limitations in current Web-based argumentation systems, we proposed the theoretical and the software foundations of a *World Wide Argument Web (WWAW)*: a large-scale Web of structured and inter-connected arguments (Rahwan *et al.*, 2007). Here, following the same principles as WWAW, we present an OWL ontology that reflects the argumentation domain in a more comprehensive way than the original ontology underlying ArgDF. This ontology is based on a new reification of the AIF (Chesňevar *et al.*, 2006). In addition, OWL offers richer features than RDFS and presents the potential for automated inference over argument structures, such as inference based on DL (Baader *et al.*, 2003). As an example, reasoning can be used to infer the classification of hierarchy of argumentation schemes.

The AIF is a strong candidate for forming the foundation of a WWAW, and a brief summary is included here of how the AIF handles arguments, relations between argument components, and the thorny issues of representing conflict and representing argumentation schemes.

3.1 Argument network and nodes

The AIF represents a core ontology of argument-related concepts. Its specification can be extended to capture different argumentation formalisms and schemes. The AIF core ontology assumes that argument entities can be represented as nodes in a directed graph called an *argument network*.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

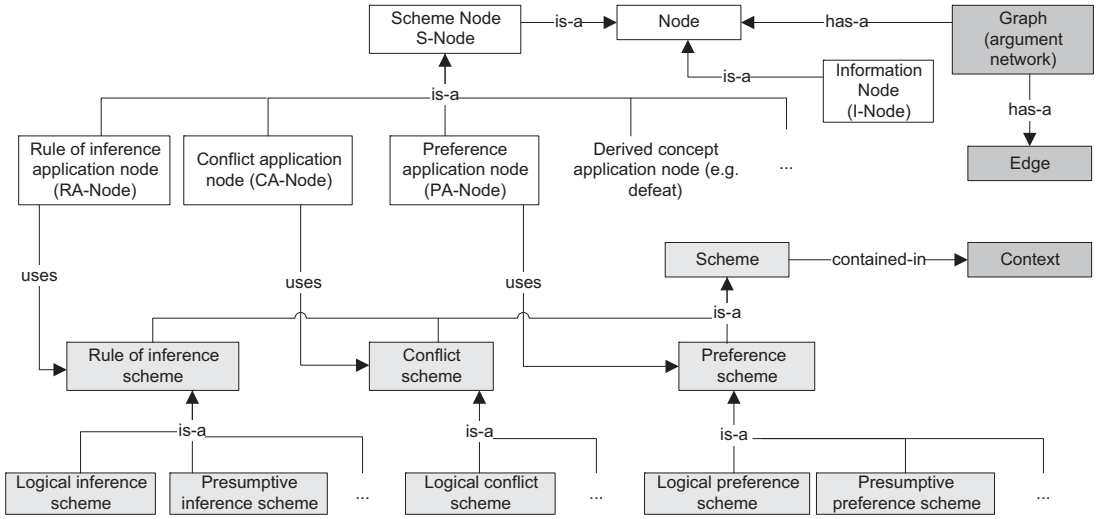


Figure 1 Original AIF Ontology

A node can also have a number of internal attributes, denoting things such as title, creator, creation date, certainty degree, acceptability status, etc. Figure 1 depicts the original AIF ontology reported by Chesňevar *et al.* (2006).

Information nodes relate to content and are used to represent *passive* information contained in an argument, such as a claim, premise or data that depend on domain of discourse. On the other hand, S-nodes capture the application of schemes (i.e. patterns of reasoning). Such schemes may be considered as 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 of schemes and can be classified further into: *rule of inference scheme*, *conflict scheme*, and *preference scheme*, etc.

The AIF specializes S-nodes further into three (disjoint) types of scheme nodes, namely *rule of inference application nodes* (RA-node), *preference application nodes* (PA-node), and *conflict application nodes* (CA-node). The word 'application' on each of these types was introduced in the AIF to emphasize the fact that these nodes function as instances, not classes, of generic inference rules. Intuitively, RA-nodes capture nodes that represent (possibly non-deductive) rules of inference, CA-nodes capture applications of criteria (declarative specifications) defining conflict (e.g. among a proposition and its negation, etc.), and PA-nodes are applications of (possibly abstract) criteria of preference among evaluated nodes.

3.2 Edges in the argument network

The argument network contains edges that connect different nodes. For example, an edge named 'uses' connects an S-node to the scheme it exploits. The AIF core specification does not type its edges. Edge semantics can be inferred from the types of nodes they connect. There are two types of edges: the *scheme edges* that emanate from S-nodes and are meant to support conclusions that follow from the S-node (these conclusions may either be I-nodes or S-nodes); and the *data edges* that emanate from I-nodes ending in S-nodes and are meant to supply data, or information to scheme applications. One of the restrictions imposed by the AIF is that no outgoing edge from an I-node can be connected directly to another I-node. This ensures that the relationship between two pieces of information must be specified explicitly via an intermediate S-node.

A simple argument in propositional logic is depicted in Figure 2(a). The S-nodes are distinguished from I-nodes graphically by drawing the former with a slightly thicker border. The node marked MP_1 denotes an application of the modus ponens inference rule.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

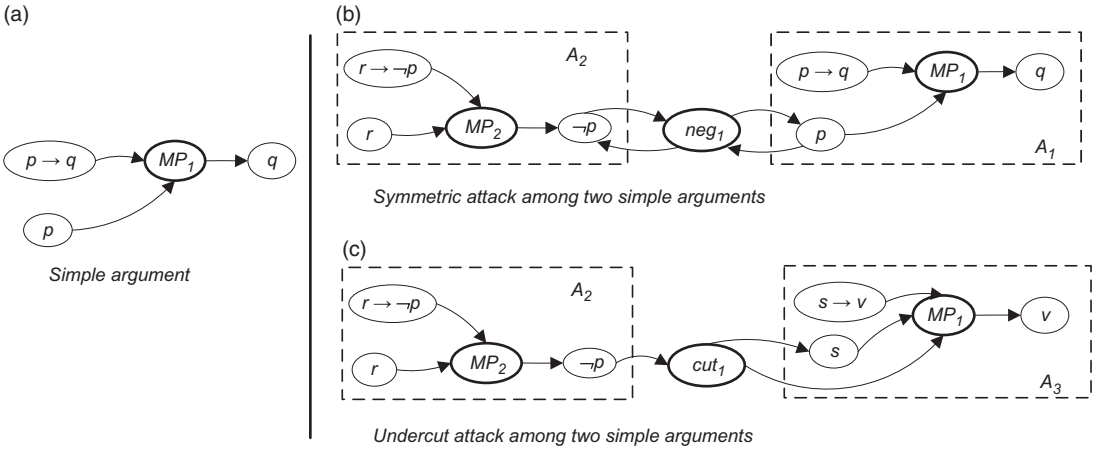


Figure 2 Examples of simple arguments and conflicts among them

3.3 Representing conflict in Argument Interchange Format

An attack or a conflict from one information or scheme node to another is captured through a CA-node, which captures the type of conflict. An asymmetric attack represents a state where one node (e.g. I-node) attacks another node (e.g. I-node) through a CA-node. On the other hand, in symmetric attacks, two nodes (e.g. I-nodes) attack each other simultaneously through a CA-node. Figure 2(b) depicts a symmetric conflict between two simple arguments (commonly known as a rebuttal in the literature). The node marked neg_1 denotes conflict as propositional negation.

Figure 2(c) illustrates a situation where a rule of inference node (RA-node) is attacked by an I-node through a CA-node. An attack on an inference application is often referred to as an undercut (Pollock, 1987)[13]. The node cut_1 represents conflict as an undercut.

3.4 Representing argumentation schemes in Argument Interchange Format

Argumentation schemes are forms of argument that capture stereotypical patterns of reasoning. They might represent the deductive or inductive forms of argument as well as forms of argument that are presumptive in nature (Reed & Walton, 2005). These schemes are referred to as *presumptive* inference patterns, in the sense that if the premises are true, then the conclusion may *presumably* be taken to be true.

Structures and taxonomies of schemes have been analyzed and proposed by many theorists, such as Perelman and Olbrechts-Tyteca (1969), van Eemeren and Grootendorst (1992), and Katzav and Reed (2004), but it is Walton's exposition (Walton, 1996) that has been most influential in computational work.

Each *Walton scheme* has a name, conclusion, set of premises and a set of critical questions. Critical questions enable contenders to identify the weaknesses of an argument based on the particular scheme, and potentially attack the argument. Here is an example:

SCHEME 1. *Argument from Position to Know*

- Position to know premise: *E is in a position to know whether A is true (false).*
- Assertion Premise: *E asserts that A is true (false).*
- Conclusion: *A may plausibly be taken to be true (false).*

[13] In some literature, asymmetric attacks by a CA-node on an I-node are also referred to as undercuts; for example, as explained by Prakken and Sartor (1997), an argument *A* undercuts another argument *B* if *A* proves(claims) what was assumed unprovable by *B*.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

- *Critical questions*
  1. Knowledge: *Is E in a position to know whether A is true (false)?*
  2. Trustworthiness: *Is E an honest (trustworthy, reliable) source?*
  3. Opinion: *Did E assert that A is true (false)?*

Many types of different schemes are explained by Walton (2006); examples of which include: *Argument from Sign*, *Argument from Analogy*, *Argument from Expert Opinion*, etc. Actual arguments are *instances* of schemes.

ARGUMENT 1. *Instance of Argument from Position to Know*

- Premise: Allen is in a position to know whether Brazil has the best football team.
- Premise: Allen says Brazil has the best football team.
- Conclusion: Brazil has the best football team.

It is possible that premises may not always be stated, in which case it is said that a given premise is *implicit* (Walton 2006). One of the benefits of argument classification is that it enables analysts to uncover the hidden premises behind an argument, once the scheme has been identified.

The *critical questions* help to evaluate arguments by serving as a means to inspect arguments based on a particular argumentation scheme. As discussed by Gordon *et al.* (2007), critical questions are not all alike. Some questions may refer to *assumptions* required for the inference to go through, while others may refer to *exceptions* to the rule, and correspond to Toulmin's *rebuttal* (Toulmin, 1958). The contemporary view is that the main difference between assumptions and exceptions lies in the *burden of proof*. The proponent of the argument has the burden of proof to answer questions about assumptions, while with exceptions the burden shifts to the questioner.

A notable aspect of schemes, receiving relatively little attention in the literature, is that they do not merely describe a *flat* ontology of arguments. Consider the following scheme.

SCHEME 2. *Argument from Expert Opinion*

- Expertise premise: *Source E is an expert in domain D containing proposition A.*
- Assertion premise: *E asserts that A is true (false).*
- Conclusion: *A may plausibly be taken to be true (false).*
- *Critical questions*
  1. Expertise: *How credible is expert E?*
  2. Trustworthiness: *Is E reliable?*
  3. Consistency: *Is A consistent with the testimony of other experts?*
  4. Backup evidence: *Is A supported by evidence?*

It is clear that this scheme *specializes* the scheme for argument from position to know. Apart from the fact that both schemes share the conclusion and the assertion premise, the statement 'Source *E* is an expert in domain *D* containing proposition *A*' is clearly a specialization of the statement that '*E* is in a position to know (things about *A*)'. Having expertise in a field causes one to be in a position to know things in that field[14].

Thus, schemes themselves have a hierarchical ontological structure, based on a classification of their constituent premises and conclusions. Capturing such structures (and in general, capturing the hierarchical ontological structure of different argumentation schemes) presents a new opportunity to enhance analysis and querying of arguments in argument networks.

Let us now consider how schemes may be formalized in the AIF. The initial AIF specification separates the classification of nodes from the classification of schemes (see Figure 1). Both nodes and schemes are independently classified upper-level concepts. S-nodes are classified into nodes that capture

[14] Indeed, there may be other reasons to be in a position to know *A*. For example, if *E* is taken to refer to society as a whole, then the argument from position to know becomes 'argument from popular opinion'.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

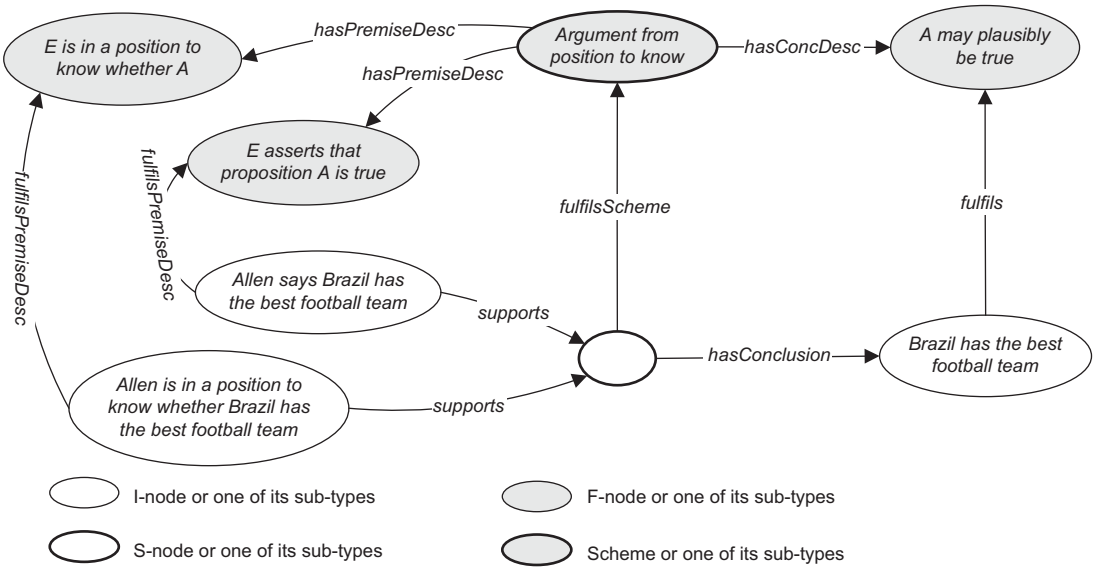


Figure 3 An argument network linking instances of argument and scheme components

inference, conflict, preference, etc. Likewise, schemes are classified into similar sub-schemes such as inference schemes, conflict schemes, and so on. S-nodes are linked to schemes via a special edge *uses*.

It should be noted that the original AIF represents an 'abstract model', allowing a number of different concrete reifications to be made. The reification of the AIF in ArgDF ontology defines two types of classes for representing schemes and nodes. Moreover, Rahwan *et al*. (2007) introduced a new type of class, *Form node (F-node)*, to capture the generic form of statements (e.g. assumptions, premises) that constitute presumptive arguments. For example, *PremiseDescriptor* is a sub-class of F-node that captures the generic form of premises used in arguments.

In ArgDF, the actual arguments are specified by instantiating node types, 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.

Figure 3 shows an argument network for 'an argument from position to know' using the underlying ontology of ArgDF. Here, each node in the actual argument (unshaded nodes) is explicitly linked, via a special-purpose property, to the form node it instantiates (shaded nodes). These special-purpose proprieties (e.g. *fulfilsScheme*) are particular reifications of the '*uses*' relation (between S-nodes and schemes) in the original AIF specification.

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 an 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 fact, the ontology presented here captures this relationship explicitly, presenting a simpler and more natural ontology of arguments. 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*'. The design of the new ontology is discussed in detail in Section 4.

4 A new argumentation ontology in Description Logic

DLs (Baader *et al*., 2003) are a family of logical formalisms that have initially been designed for the representation of conceptual knowledge in Artificial Intelligence (see the Appendix for a brief review). 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

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

instantiating the concepts), which form a world description (ABox). Since DLs have formal syntax and formal model-theoretic semantics, various reasoning algorithms can be formulated, and we use some of these in this paper.

The new formalization of the AIF argumentation ontology is expressed using the Ontology Language OWL (McGuinness & van Harmelen, 2004) in DL notation. The ontology is designed using a particular dialect of OWL, called OWL-DL, which is equivalent to logic $\mathcal{SHOIN}(\mathcal{D})$ (Baader, *et al.*, 2003). Although very expressive[15], $\mathcal{SHOIN}(\mathcal{D})$ is still decidable, and comes with efficient reasoning support.

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 specializations 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 specialized to capture deductive or presumptive arguments. The same can be done with different types of conflicts, preferences, and so on.

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

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:

$$\top \sqsubseteq \forall creationDate.Date$$
$$\top \sqsubseteq \forall creationDate^{-}.Scheme$$
$$\top \sqsubseteq \forall argTitle.String$$

[15] $\mathcal{SHOIN}(\mathcal{D})$ allows expression of basic DL, transitive roles, nominals, role hierarchy, inverse roles, and number restrictions.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

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

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.

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

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

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

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:

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

4.2 Examples

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`[16] 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"

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

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

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:

$$\begin{aligned}&\mathit{ArgFromPositionToKnow} \equiv (\mathit{PresumptiveArgument} \sqcap\\ &\quad \exists \mathit{hasConclusion}.\mathit{KnowledgePositionStmnt} \sqcap\\ &\quad \exists \mathit{hasPremise}.\mathit{PositionToHaveKnowledgeStmnt} \sqcap\\ &\quad \exists \mathit{hasPremise}.\mathit{KnowledgeAssertionStmnt})\\ &\mathit{ArgFromPositionToKnow} \sqsubseteq \exists \mathit{hasException}.\mathit{LackOfReliabilityStmnt}\end{aligned}$$

Now, for the 'Appeal to Expert Opinion' scheme, we only need to define one additional premise type, since both the conclusion and the assertion premise are identical to those of 'Argument from Position to Know':

$\mathit{FieldExpertiseStmnt} \sqsubseteq \mathit{PositionToHaveKnowledgeStmnt}$
`formDescription`: "source E is an expert in subject domain D containing proposition A"

Similarly, one of the exceptions of this scheme is identical to 'Argument from Position to Know'. The remaining assumptions and exception are added as follows:

$\mathit{ExpertiseInconsistencyStmnt} \sqsubseteq \mathit{DeclarativeStatement}$
`formDescription`: "A is not consistent with other experts assertions"
$\mathit{CredibilityOfSourceStmnt} \sqsubseteq \mathit{DeclarativeStatement}$
`formDescription`: "E is credible as an expert source"
$\mathit{ExpertiseBackUpEvidenceStmnt} \sqsubseteq \mathit{DeclarativeStatement}$
`formDescription`: "E's assertion is based on evidence"

Likewise, the necessary and sufficient conditions of 'Appeal to Expert Opinion' are:

$$\begin{aligned}&\mathit{AppToExpertOpinion} \equiv (\mathit{PresumptiveArgument} \sqcap\\ &\quad \exists \mathit{hasConclusion}.\mathit{KnowledgePositionStmnt} \sqcap\\ &\quad \exists \mathit{hasPremise}.\mathit{FieldExpertiseStmnt} \sqcap\\ &\quad \exists \mathit{hasPremise}.\mathit{KnowledgeAssertionStmnt})\\ &\mathit{AppToExpertOpinion} \sqsubseteq \exists \mathit{hasException}.\mathit{LackOfReliabilityStmnt}\\ &\mathit{AppToExpertOpinion} \sqsubseteq \exists \mathit{hasException}.\mathit{ExpertiseInconsistencyStmnt}\\ &\mathit{AppToExpertOpinion} \sqsubseteq \exists \mathit{hasAssumption}.\mathit{CredibilityOfSourceStmnt}\\ &\mathit{AppToExpertOpinion} \sqsubseteq \exists \mathit{hasAssumption}.\mathit{ExpertiseBackUpEvidenceStmnt}\end{aligned}$$

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

4.3 Capturing support and conflict among arguments

Arguments can be chained together where a claim acts both as a premise of one argument and as a conclusion of another. A transitive property named *supports* was added to the ontology, to allow linking the supporting argument to the supported argument in a chain:

$$\mathit{RuleScheme} \sqsubseteq \forall \mathit{supports}.\mathit{RuleScheme}$$

Conflict among arguments are captured through different specializations of *ConflictScheme* such as *GeneralConflict* and *ExceptionConflict*:

$$\mathit{ExceptionConflict} \sqsubseteq \mathit{ConflictScheme}$$
$$\mathit{GeneralConflict} \sqsubseteq \mathit{ConflictScheme}$$

GeneralConflict instances capture simple symmetric and asymmetric attacks among arguments while *ExceptionConflict* instances represent exceptions to rules of inference. The definition of

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

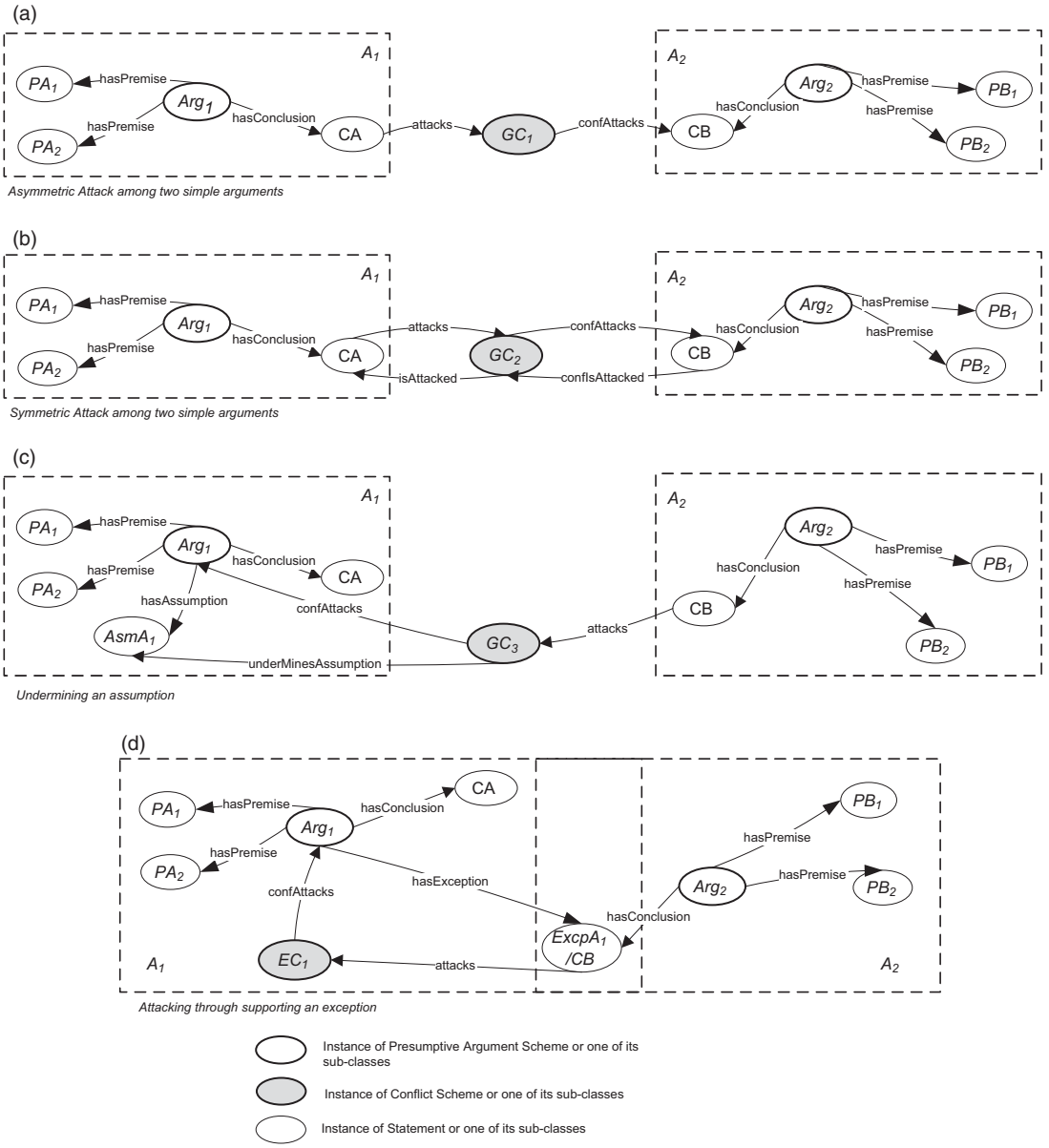


Figure 4 Representation of different types of attack among arguments

ConflictScheme and *Statement* classes have been extended to include the appropriate restrictions on properties used to represent attacks among different arguments:

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

Figures 4(a–d) illustrate how instances of conflict scheme 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

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

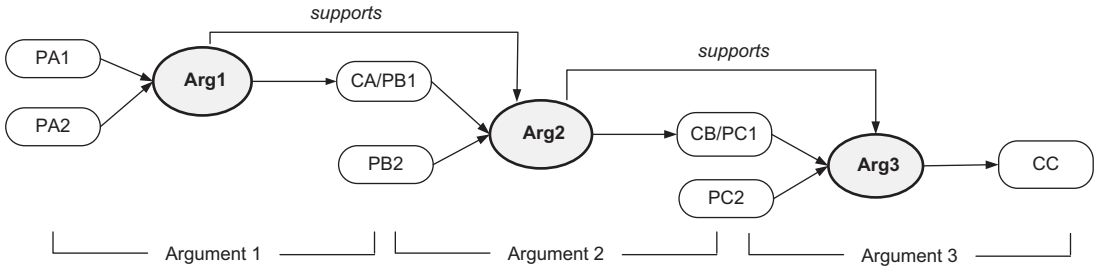


Figure 5 Support among chained arguments

and exception conflict as GC_n and EC_1, respectively, where $X = \{A, B, C, \ldots\}$ and n represents the set of natural numbers $\{1, 2, 3, \ldots\}$.

5 Web Ontology Language reasoning over argument structures

In this section, we describe a number of ways in which the expressive power of OWL and its support for reasoning can be used to enhance user interaction with arguments.

5.1 Inference of indirect support in chained arguments

One of the advantages of OWL over RDF Schema is that OWL supports inference over transitive properties. In other words, if $r(X,Y)$ and $r(Y,Z)$, then OWL reasoners can infer $r(X,Z)$. This can be used to enhance argument querying.

Figure 5 shows three arguments chained together. In Argument 1, premises PA1 and PA2 have the conclusion CA, which is used at the same time as premise PB1 of the argument 2. Premises PB1 and PB2 have the conclusion CB, which is used at the same time as premise PC1 of argument 3; PC1 and PC2 have the conclusion CC. Here, we can say that Argument 1 *indirectly* supports Argument 3. An OWL reasoner supports small and efficient queries corresponding to user requests for all arguments that directly or indirectly support some conclusion.

5.2 Automatic classification of argumentation schemes and instances

In this section, we describe the general inference pattern behind classification of argumentation 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.

Let us consider two specializations (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 $(PA_1, PA_2, \ldots, PA_n)$, where classes CA and $(PA_1, PA_2, \ldots, PA_n)$ are specializations of the class *Statement*. Similarly, *PresScheme*2 has members of CB class as its conclusion and its premises are from classes $(PB_1, PB_2, \ldots, PB_m)$ where CB and $(PB_1, PB_2, \ldots, PB_m)$ are specializations of *Statement* and $m \geqslant n$. 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 specialization of the former, that is, $(CB \equiv CA) \vee (CB \sqsubseteq CA)$.

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

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

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

$$\begin{aligned}
&PresScheme1 \equiv (PresumptiveArgument \sqcap \\
&\exists hasConclusion.CA \sqcap \\
&\exists hasPremise.PA1 \sqcap \\
&\exists hasPremise.PA2 \sqcap \\
&\exists hasPremise.(\ldots) \sqcap \\
&\exists hasPremise.PAn) \\
&PresScheme2 \equiv (PresumptiveArgument \sqcap \\
&\exists hasConclusion.CB \sqcap \\
&\exists hasPremise.PB1 \sqcap \\
&\exists hasPremise.PB2 \sqcap \\
&\exists hasPremise.(\ldots) \sqcap \\
&\exists hasPremise.PBm)
\end{aligned}$$

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 specialization of a premise in *PresScheme*1 and/or the conclusion *CB* is a specialization of *CA*, *PresScheme*2 is also inferred as the sub-class of *PresScheme*1.

Following the above explanation, due to the hierarchy of specialization 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 specialization relationship that can be inferred between 'Fear Appeal Argument' and 'Argument from Negative Consequences'.

SCHEME 3. *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 4. *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 do not 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?*

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

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

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

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.

5.3 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 specializations 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 specializations 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

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

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 Implementation

In this section, we explain the basic architecture of the implemented Web-based system *Avicenna*[17]. A comparison among different tools/technologies for building this system as well as the reasons

[17] Avicenna was a Persian polymath, physician, and Islamic philosopher (see `http://en.wikipedia.org/wiki/Avicenna`). He developed an early theory on hypothetical syllogism, which formed the basis of his early risk factor analysis. In addition to developing an early theory on propositional calculus and an

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

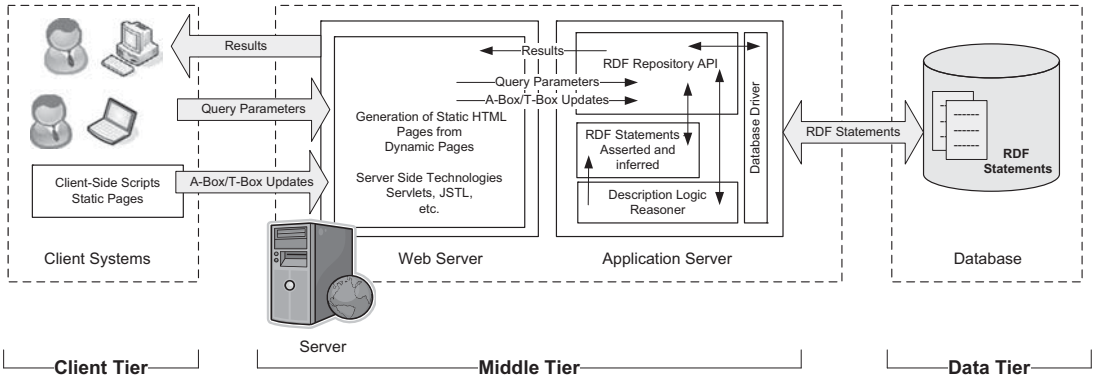


Figure 6 Basic System Architechture

for choosing each tool/technology is also provided. Moreover, the main features of this system are highlighted and briefly explained.

Avicenna's basic system architecture is illustrated in Figure 6. It consists of three main tiers: the data tier, the middle tier, and the client tier. The argumentation ontology (including both the TBox and the ABox) is stored in form of RDF statements (triples) in the back-end database, which constitutes the data tier. The middle tier is responsible for the DL reasoning and the interface to the web, over which applications in the client tier connect.

DL Semantic Web ontologies have a rich tooling environment. We used Protégé to implement the ontology, Jena as a repository supporting SPARQL queries through ARQ, and Pellet for reasoning.

6.1 Exploring available arguments

The system lists the available arguments by listing their titles. These titles are in form of hyperlinks and can be used to navigate to a page where the details of the argument (its scheme, author, conclusion, premises, and critical questions) are listed for further exploration. Figure 7 displays the details of argument 'Tipping lowers self esteem', which is an instance of 'Argument from Expert Opinion.'

The *View Arguments* navigation menu item displays recent argument threads; the user can also *Search Arguments* on the basis of keywords, structural features, and related properties (creation date, author, etc.).

6.2 Creating new arguments

New semantically annotated arguments can be authored using new claims or claims already existing as part of available arguments in the system. By selecting *Add New Argument* from the navigation menu, the user has the choice of creating a new argument that adheres to any of the existing argumentation schemes.

After a specific argumentation scheme is chosen by the user, its constituent parts (the conclusion, the premises, the assumptions, and exceptions) are extracted by running a query on the different restrictions defined on the scheme. If certain assumptions or exceptions are not explicitly defined on a scheme but are defined on any of its deduced super-classes, the description logic reasoner is able to infer them and add them to the list of asserted assumptions and exceptions.

The user is then forwarded to a page with a form containing place holders for the different parts of the argument to be filled. Beside each place holder, a brief description of the claim format

(F'note continued)
original theory on temporal modal syllogism, he also developed his own system of logic known as 'Avicennian logic' as an alternative to Aristotelian logic. Avicenna also contributed inventively to the development of inductive logic, being the first to describe the methods of agreement, difference, and concomitant variation, which are critical to inductive logic and the scientific method.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

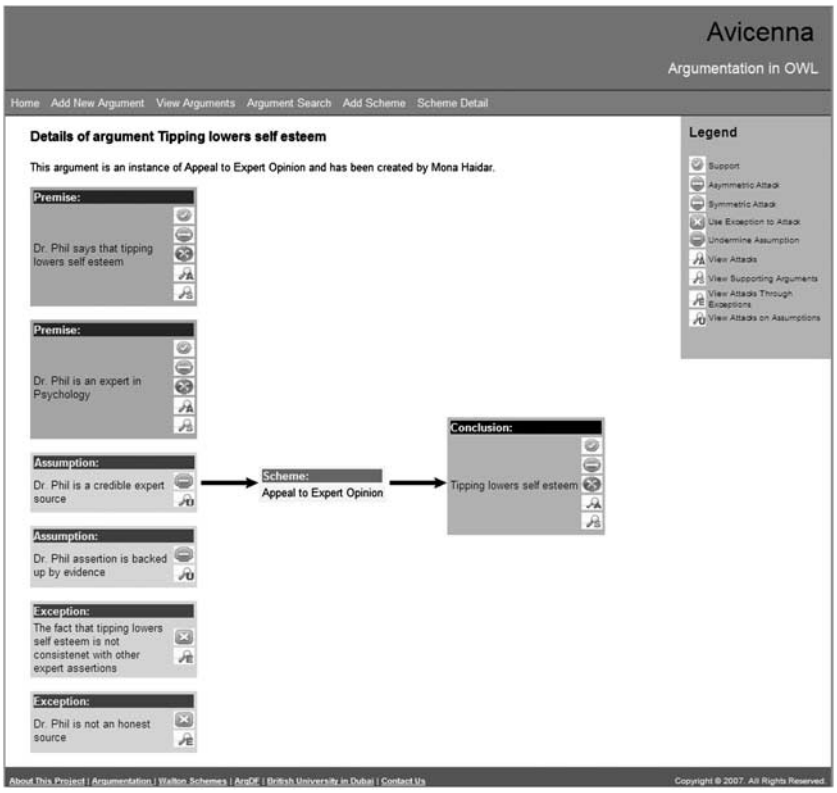


Figure 7 Argument Details

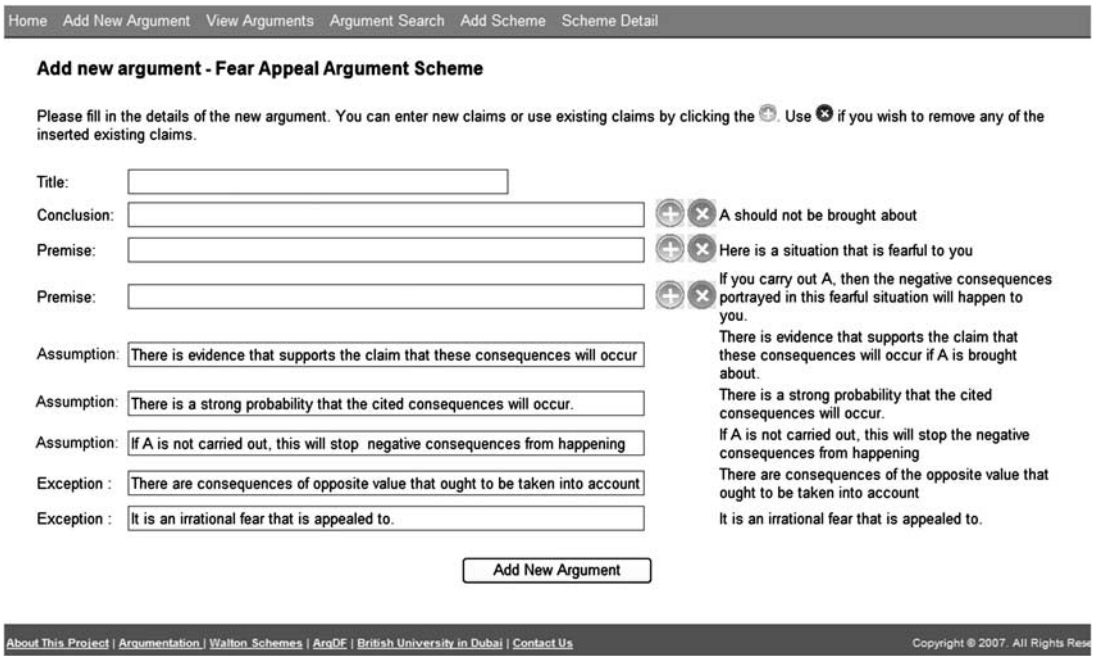


Figure 8 Adding New Arguments

is provided. The textual contents of assumptions and exceptions are already filled in respective placeholders and only require minimal change by the user. Figure 8 illustrates the page for authoring a new argument instance of 'Fear Appeal Argument'.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

The user may enter new claims or may choose to use any of the existing claims. He can access the existing claims by clicking a link to access a page that displays all the available claims in the system. This list can be searched and filtered as required. The paging technique is implemented (through code) to limit the number of claims displayed in the page at any given time.

The instances of the conclusion and the premises are created under the appropriate statement classes and linked to the argument instance through *hasConclusion* and *hasPremise* properties. Instances of assumptions and exceptions (if available) are also created under the appropriate statement classes. Instances of assumptions represent a set of implicit premises of the argument and are connected to the argument instance through *hasAssumption* property. Instances of exceptions are connected to the argument instance through instances of *ExceptionConflict* and *hasException* property; such exceptions will not undercut the presumptive argument instance unless they are supported by further statements.

6.3 Attacking and supporting existing arguments

Users can attack or support existing arguments. The available operations on each claim making up the argument are accessible through different icons on the right side of each claim as illustrated in Figure 7. Users can perform symmetric or asymmetric attacks (see Section 3.3) on a conclusion or a premise.

Users can also choose to support the conclusion or premises of an argument. As explained in Section 5.1, if the supported claim is the premise of an argument, this claim is both the conclusion of the supporting argument and the premise of the supported argument; thus creating a chain of arguments. Users can also undercut (see Section 3.3) an argument by undermining an existing assumption or supporting an existing exception of an argument. In every case, the user is required to add a new argument fulfilling the supporting, attacking, or undercutting role.

As the system allows re-using of existing claims while authoring new arguments as well as supporting and attacking different claims that are part of an existing argument, interlinked and dynamic argument networks are created—a central feature provided by the underlying ontology design.

6.4 Retrieving attacking/supporting arguments of a claim

Viewing different types of attacks on a premise or conclusion is also available through a set of icons displayed on the right-hand side of the claim. Users can view the different claims that are attacking or being attacked by the claim and then view the arguments that those claims are part of. When the user chooses to view the attacks made against a certain claim, he has the option of choosing among three different options: to view the claims that this claim attacks, or the claims that attack this claim and finally the claims that attack this claim and are attacked by it at the same time (symmetric attack). The result of choosing any of the options is displaying the list of claims that are involved in a conflict relationship with the initial claim.

Users can also query the network to view different arguments that support a certain premise or conclusion; the transitivity feature of the *supports* property is used to return all the arguments supporting the claim whether directly or indirectly (as depicted in Figure 5).

7 Towards Argument Web Mining

In this section, we briefly discuss how our ontology can form a foundation for more sophisticated applications that exploit ideas from *Semantic Web Mining*.

Web mining is the research field concerned with applying data mining techniques[18] to the data available on the world wide web. Such data includes content (e.g. text, images, and videos),

[18] Data mining is 'the non-trivial process of identifying valid, previously unknown, and potentially useful patterns' (Fayyad *et al.*, 1996).

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

structure (e.g. links across webpages), and web usage (e.g. user browsing patterns). Recently, web mining and the Semantic Web developed a benevolent relationship (the combination is called *Semantic Web Mining*; Stumme *et al.*, 2006). Web mining provides an automated mechanism for building Semantic Web ontologies and semantically annotating unstructured Web pages. The Semantic Web structure on the other hand improves web mining results by providing semantic information.

We envision a similar cooperative relationship between web mining and the argument Web, which we call the *Argument Web Mining*. Information extraction techniques (ranging from keyword extraction to natural language processing) can automatically locate facts, concepts, and structure in text, which can then be combined with our argumentation ontology in order to automatically (or semi-automatically via interaction with a knowledge engineer) extract argument instances and annotate the electronic document accordingly. Web mining can also be applied to documents that are semantically tagged in order to automatically learn rules for recognizing arguments from semantic information (e.g. an $\langle expert \rangle$ and $\langle says \rangle$ tag may trigger the recognition of an 'argument from expert opinion'). Unsupervised learning, such as k-means clustering and association rule discovery (Hand *et al.*, 2001), can help discover new argumentation schemes and augment our proposed ontology, a process that is lengthy and error prone if done manually. Mining user browsing patterns can also help in the construction of the argument Web. For example, a user browses an opinion regarding a particular product then decides to add the product to her shopping cart. This behaviour indicates that the browsed opinion is a strong argument in favour of the product.

Once 'shallow' argument structure is identified through Web mining techniques, our ontological reasoning technique can provide more fine-grained, knowledge-based classification of the arguments. Hence, our approach provides part of a very sophisticated argument retrieval on the Web.

Web mining also benefits from the Argument Web. Aside from allowing richer information retrieval (e.g. 'what are the arguments by experts against product A?'), the argument Web can provide added value to argument-enabled Web sites. For example, when a user views an opinion about a product, the argument web can be used to automatically point out the weaknesses in the opinion's argument (e.g. 'the reviewer is not an expert').

8 Discussion and conclusions

This paper is a contribution to a line of work towards an AIF. The AIF aims to facilitate the exchange of semi-structured arguments among different argument analysis and argumentation-support tools, as well as the representation of arguments on the Web in a unified format. While much prior work has been done on representing rules in XML for the applications on the Web (e.g. RIF[19] or SWRL; Horrocks *et al.*, 2005), relatively little work has been done on representing arguments in a format suitable for Web applications. The AIF provides primitives specifically aimed at representing aspects of complex argument structures, such as attack between arguments, implicit presumptions underlying explicit arguments, and capturing stereotypical classes of arguments as classified by theoretical work on argumentation schemes (Walton, 1996). The AIF is still at a very early stage when compared with formal rule interchange languages, and this paper is a contribution to bring the AIF up to speed.

We presented a Description Logic ontology for annotating arguments, based on a new reification of the AIF and founded in Walton's theory of argumentation schemes. We demonstrated how this ontology enables automated reasoning over argument structures. In particular, OWL reasoning enables significantly enhanced querying of arguments through automatic scheme classifications, instance classification, inference of indirect support in chained argument structures, and inference of critical questions. We present the implementation of a pilot Web-based system, called Avicenna, for authoring and querying argument structures using the proposed ontology.

[19] www.w3.org/2005/rules/

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

Avicenna is a candidate implementation for contributing to the task of bringing the WWAW into existence. It supports storage of arguments in persistent RDF storage where it is possible to create new arguments by introducing new claims, re-using existing claims, or introducing new attacks. While creating new arguments, the inferred critical questions on that scheme are also added to the argument instance.

Avicenna implements and uses queries in different tasks; for example: displaying the different parts of an argument instance, displaying the details of an argumentation scheme and searching for arguments in both basic and advanced modes (advanced search queries the argument network based on different parameters; in case of searching for instances of a specific argumentation scheme, inference is used to return the instances of inferred sub-classes of that argumentation scheme as well). Queries are also utilized to return attacking or supporting arguments of a given claim; searching for supporting arguments of a claim retrieves arguments that support the claim both directly and indirectly.

It is important to note that our aim here was not to present an extensive and complete ontology of argumentation schemes. This task is beyond the scope of any single paper, and is a topic under development in its own right (Walton, 1996; Walton *et al.*, 2008). Our aim here was to show some new kinds of automated reasoning, over such schemes, made possible by Semantic Web technologies. Hence, we focused on specifying the AIF top-level ontology, then specializing it through some specific schemes as serves our purpose.

Avicenna does, however, have some key limitations. Firstly, the current approach to definition of argumentation schemes (and therefore, the inference of hierarchy of schemes) is based on the necessary and sufficient conditions on *hasPremise* and *hasConclusion* properties of each scheme (as explained in detail in Section 5.2). Certain argument schemes exhibit somewhat complex structures; for example, 'Circumstantial Ad Hominem Argument' is a chain of argumentation based on combining 'Argument from Inconsistent Commitment' with the 'Direct Ad Hominem Argument' (Walton, 2006). In this scheme, an intermediate proposition forms the conclusion of one scheme (argument from inconsistent commitment), while its final conclusion is based on the conclusion of another scheme (direct ad hominem argument). It is not clear how such compound schemes should be treated, either in Avicenna, or indeed in the AIF. Considering the different types of argumentation schemes that the ontology must incorporate and the expected classification hierarchy results, one possibility is that the necessary-and-sufficient conditions on each scheme might be re-defined by using a new property, *hasPart*, that stands for both *hasPremise* and *hasConclusion* properties, and properties *hasPremise* and *hasConclusion* will become part of the necessary conditions.

It is important to note that the semantics captured by Avicenna pertains to the typology and overall structure of the arguments. This is quite distinct from (but complimentary to) the *argument acceptability* semantics studied extensively in the literature (Baroni & Giacomin, 2007). Indeed, the AIF's original purpose was only to describe the structure of arguments, leaving evaluation of argument acceptability to the various available theories.

From a technical perspective, the SPARQL query language is limited in its handling of transitive properties in that it is not possible to limit the depth of application of transitive properties. In the current system, when the user queries to view the supporting arguments of a claim, it is not possible to limit the returned results to those triples that provide indirect support up to two levels back (or any arbitrary number specified by the user).

This papers opens up many avenues for future research, the most important of which is perhaps enabling people to easily author annotated arguments. This is crucial for accumulating sizeable content suitable for further analysis and reasoning experiments. Some promising progress has been made recently by providing Web interfaces for direct manipulation of networked visualizations (e.g. as in Cohere (Shum 2008) or Debategraph) and through argument blogging (Wells *et al.*, 2009). Another important research area is automatic argument tagging through a combination of information retrieval and text analysis techniques (see Section 7 for more on this).

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

Appendix: Description Logics

Table 1 shows the syntax and semantics of common concept and role constructors. The letters A, B are used for atomic concepts and C, D for concept descriptions. For roles, the letters R and S are used and non-negative integers (in number restrictions) are denoted by n, m and individuals (i.e. instances) by a, b. An *interpretation* $\mathcal{I}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Table 1 Some Description Logic Role Constructors, Concept Constructors, and Terminological Aximos

Name	Syntax	Semantics
Concept and Role Constructors		
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Concept Intersect.	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Concept Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Concept Negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
Value Restriction	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
Exist. Quantifier	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Unqualified	$\geq nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}} \mid \geq n\}$
Number	$\leq nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}} \mid \leq n\}$
Restriction	$=nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}} \mid = n\}$
Role-value-map	$R \subseteq S$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a, b) \in R^{\mathcal{I}} \rightarrow (a, b) \in S^{\mathcal{I}}\}$
	$R = S$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a, b) \in R^{\mathcal{I}} \leftrightarrow (a, b) \in S^{\mathcal{I}}\}$
Nominal	I	$I^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ with $\mid I^{\mathcal{I}} \mid = 1$
Universal Role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Role Intersection	$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
Role Union	$R \sqcup S$	$R^{\mathcal{I}} \cup S^{\mathcal{I}}$
Role Complement	$\neg R$	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \backslash R^{\mathcal{I}}$
Role Inverse	R^{-}	$\{(b, a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}}\}$
Transitive Closure	R^{+}	$\bigcup_{n \geq 1} (R^{\mathcal{I}})^n$
Role Restriction	$R \mid c$	$R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \times C^{\mathcal{I}})$
Identity	$id(C)$	$\{(d, d) \mid d \in C^{\mathcal{I}}\}$
Teminological Axioms		
Concept Inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept Equality	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Role Inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
Role Equality	$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$

A DL knowledge base consists of a set of terminological axioms (often called *TBox*) and a set of assertional axioms or assertions (often called *ABox*). A finite set of definitions is called a *terminology* or *TBox* if the definitions are unambiguous, that is, no atomic concept occurs more than once as left-hand side.

To give examples of what can be expressed in DLs, we suppose that Person and Female are atomic concepts. Then $Person \sqcap Female$ is DL concept describing, intuitively, those persons that are female. If, in addition, we suppose that *hasChild* is an atomic role, we can form the concept $Person \sqcap \exists\, hasChild$, denoting those persons that have a child. Using the bottom concept, we can also describe those persons without a child by the concept $Person \sqcap \forall hasChild.\bot$. These examples show how we can form complex descriptions of concepts to describe classes of objects.

The *terminological axioms* make statements about how concepts or roles are related to each other. It is possible to single out definitions as specific axioms and identify terminologies as sets of definitions by which we can introduce atomic concepts as abbreviations or names for complex concepts.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

An equality whose left-hand side is an atomic concept is a *definition*. Definitions are used to introduce symbolic names for complex descriptions. For instance, by the axiom $Mother \equiv Woman \sqcap \exists hasChild.Person$, we associate to the description on the right-hand side the name *Mother*. Symbolic names may be used as abbreviations in other descriptions. If, for example, we have defined *Father* analogously to *Mother*, we can define *Parent* as $Parent \equiv Mother \sqcup Father$.

The sentence $T \sqsubseteq \forall hasParent.Person$ expresses that the range of the property *hasParent* is the class *Person* (more technically, if the property *hasParent* holds between any concept and another concept, the latter concept must be of type *Person*).

References

Atkinson, K., Bench-Capon, T. J. M. & McBurney, P. 2006. PARMENIDES: facilitating deliberation in democracies. *Artificial Intelligence and Law* **14**(4), 261–275.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. (eds). 2003. *The Description Logic Handbook*. Cambridge University Press.

Baroni, P. & Giacomin, M. 2007. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence* **171**(10–15), 675–700.

Brickley, D. & Guha, R. V. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation REC-rdf-schema-20040210, World Wide Web Consortium (W3C). `http://www.w3.org/TR/rdf-schema/`

Chesňevar, C. I., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G. & Willmott, S. 2006. Towards an Argument Interchange Format. *The Knowledge Engineering Review* **21**(4), 293–316.

Farnham, S., Chesley, H. R., McGhee, D. E., Kawal, R. & Landau, J. 2000. Structured Online Interactions: Improving the Decision-Making of Small Discussion Groups. In *Proceedings of the 2000 ACM conference on Computer Supported Cooperative Work*. ACM Press, 299–308.

Fayyad, U., Piatetsky-shapiro, G. & Smyth, P. 1996. From data mining to knowledge discovery in databases. *AI Magazine* **17**, 37–54.

Gordon, T. F., Prakken, H. & Walton, D. 2007. The Carneades model of argument and burden of proof. *Artificial Intelligence* **171**(10–15), 875–896.

Hand, D. J., Mannila, H. & Smyth, P. 2001. *Principles of Data Mining*. MIT Press.

Horrocks, I., Patel-Schneider, P. F., Bechhofer, S. & Tsarkov, D. 2005. OWL rules: a proposal and prototype implementation. *Journal of Web Semantics* **3**(1), 23–40.

Kalfoglou, Y. & Schorlemmer, M. 2003. Ontology mapping: the state of the art. *Knowledge Engineering Review* **18**(1), 1–31.

Katzav, J. & Reed, C. 2004. On argumentation schemes and the natural classification of arguments. *Argumentation* **18**(2), 239–259.

McGuinness, D. L. & van Harmelen, F. 2004. *OWL Web Ontology Language Overview*. W3C Recommendation REC-owl-features-20040210, World Wide Web Consortium (W3C). `http://www.w3.org/TR/owl-features/`

Perelman, C. & Olbrechts-Tyteca, L. 1969. *The New Rhetoric: a Treatise on Argumentation*. University of Notre Dame Press.

Pollock, J. L. 1987. Defeasible reasoning. *Cognitive Science* **11**(4), 481–518.

Prakken, H. & Sartor, G. 1997. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics* **7**(1), 25–76.

Rahwan, I. 2008. Mass Argumentation and the Semantic Web. *Journal of Web Semantics* **6**(1), 29–37.

Rahwan, I. & Banihashemi, B. 2008. Arguments in OWL: a progress report. In *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, Besnard, P., Doutre, S. & Hunter, A. (eds). IOS Press, 297–310.

Rahwan, I., Zablith, F. & Reed, C. 2007. Laying the foundations for a World Wide Argument Web. *Artificial Intelligence* **171**(10–15), 897–921.

Reed, C. & Rowe, G. 2004. Araucaria: software for argument analysis, diagramming and representation. *International Journal of AI Tools* **14**(3–4), 961–980.

Reed, C. & Walton, D. 2005. Towards a formal and implemented model of argumentation schemes in agent communication. *Autonomous Agents and Multi-Agent Systems* **11**(2), 173–188.

Shum, S. B. 2008. Cohere: towards Web 2.0 argumentation. In *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, Hunter A. (ed.). IOS Press.

Stumme, G., Hotho, A. & Berendt, B. (2006). Semantic Web mining: state of the art and future directions. *Web Semantics: Science, Services and Agents on the World Wide Web* **4**(2), 124–143.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press

Toulmin, S. E. 1958. *The Uses of Argument*. Cambridge University Press.
van Eemeren, F. H. & Grootendorst, R. F. 1992. *Argumentation, Communication and Fallacies: A Pragma-Dialectical Perspective*. Lawrence Erlbaum Associates.
Verheij, B. 2005. *An argumentation core ontology as the centerpiece of a myriad of argumentation formats*. Agentlink Technical Forum Group.
Völkel, M., Krötzsch, M., Vrandečić, D., Haller, H. & Studer, R. 2006. Semantic wikipedia. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006*. ACM Press, 585–594.
Walton, D. N. 1996. *Argumentation Schemes for Presumptive Reasoning*. Erlbaum.
Walton, D. N. 2006. *Fundamentals of Critical Argumentation*. Cambridge University Press.
Walton, D., Reed, C. & Macagno, F. 2008. *Argumentation Schemes*. Cambridge University Press.
Wells, S., Gourlay, C. & Reed, C. 2009. Argument blogging. In *Proceedings of the 9th International Workshop on Computational Models of Natural Argument (CMNA)*, Pasadena, California.

https://doi.org/10.1017/S0269888911000191 Published online by Cambridge University Press