Interchanging arguments between Carneades and AIF

Floris BEX [a], Thomas GORDON [b], John LAWRENCE [a] and Chris REED [a]
[a] *Argumentation Research Group, School of Computing, University of Dundee, UK*
[b] *Fraunhofer FOKUS, Berlin, Germany*

AbstractWe have implemented a translator that translates Carneades Argument Graphs as specified in the LKIF files of the Carneades editor to a database specification of the Argument Interchange Format and vice versa. In this paper the algorithms for this translation are presented.

Keywords. argument interchange, argument ontology, argumentation tools

1. Introduction

The Argument Interchange Format (AIF) [4] aims to facilitate the research and development of tools for argument manipulation, argument visualization and multi-agent argumentation. An abstract core ontology that encapsulates the common subject matter of the various approaches to argumentation has been proposed, which can act as a centrepiece to individual (logical, linguistic, graphical) languages for expressing arguments. Furthermore, services are available that allow for interchange between implementations of the AIF ontology and a number of argument visualisation tools, such as Rationale [13] and Araucaria [11]. Translation functions also exist [1] between the abstract specification of the AIF and the formal logical language of ASPIC$^+$ [9], formally grounding the AIF ontology. Thus, using the AIF as an interlingua arguments constructed in Rationale or Araucaria, can be evaluated using the different semantics proposed in [9].

The Carneades system is a tool for argument visualisation, evaluation and construction based on the data model of Carneades Argument Graphs (CAGs) [6]. While at present it can be considered a generally applicable argumentation support tool, it was originally developed with legal reasoning in mind and incorporates an explicit notion of proof standards in its computation of the acceptability of statements. Further evidence of Carneades' legal heritage is evident in the file format it uses, an XML serialisation of the Legal Knowledge Interchange Format (LKIF). The authors intend to extend the reach of the AIF by defining and implementing algorithms for translating Carneades Argument Graphs, as specified in the LKIF format of the Carneades editor, to AIF argument graphs as specified in the SQL specification of the AIF.

2. The Argument Interchange Format

The AIF aims to consolidate some of the defining work on (computational) argumentation [4]. It works under the assumption that a common vision and consensus on the

concepts and technologies in the field promotes the research and development of new argumentation tools and techniques. In addition to practical aspirations, such as developing a way of interchanging data between tools for argument manipulation and visualization, the AIF project also aims to develop a commonly agreed-upon core ontology that specifies the basic concepts used to express arguments and their mutual relations.

The AIF ontology places at its core a distinction between *information* and *schemes*, patterns that describe argumentative relations between information. Inference, conflict and preference are treated as genera of a more abstract class of schematic relationships [3], which greatly simplifies the ontological machinery required for handling them. Thus, inference schemes (which are akin to argumentation schemes) and conflict schemes in the AIF ontology embody the general principles expressing how it is that q is inferable from p or p is in conflict with q, respectively.

The AIF ontology further assumes that the individual entities that fulfil or instantiate generic schemes can be represented as nodes in a directed graph called an *AIF argument graph*. Accordingly, there are two types of nodes: information nodes (I-nodes) and scheme nodes (S-nodes). I-nodes are used to represent information or statements, which may serve as, for example, claims, premises, conclusions, attackers and so on. S-nodes denote applications of schemes: rule application nodes (RA-nodes) denote applications of an inference rule or scheme and conflict application nodes (CA-nodes) denote specific conflicts. In our example, RA- and CA-nodes capture the passage or the process of actually inferring q from p or conflicting p with q, respectively.

Definition 2.1 [AIF graph]
An *AIF argument graph* G is a simple digraph consisting of nodes and edges.

1. Each node has one type either I or RA or CA;
2. There are no edges between two nodes of type I;
3. Given two nodes n_1 and n_2 and an edge (n_1, n_2), we say that n_1 is a predecessor of n_2 and n_2 is a successor of n_1;
4. Nodes of type RA and CA have at least one predecessor and one successor.

I-nodes can only be connected to other I-nodes via S-nodes (2): there must be a scheme that expresses the rationale behind the relation between I-nodes. S-nodes, on the other hand, can be connected to other S-nodes directly (see Figure 1); however, there must always be I-nodes at the beginning or end of a chain of S-nodes (4). The edges in a graph are typed; Table 1 shows the default edge types for the various combinations of nodes [4]. Note that the full AIF specification also has preference schemes and preference application (PA) nodes, but that since Carneades does not have an explicit notion of preference these are not needed for current purposes. Figure 1 shows an example of an AIF argument graph, in which I-nodes are shown as rectangles and S-nodes as diamonds. Note the conflict between the information that Bob is not reliable and the inference from Bob's testimony to the conclusion, which represents an undercutter.

In Definition 2.1, the concepts from the AIF ontology are presented as a simple algebraic structure. Representing the ontology in such an abstract way allows us to compare it with other algebraic models of argument, such as the ASPIC+ framework [9,1] and Carneades Argument Graphs (section 3). However, the AIF has also been implemented in a number of languages aimed at practical applications, such as RDFs and OWL-XML.

	to I-node	*to RA-node*	*to CA-node*
from I-node		information used in applying an inference	information in conflict with successor of the CA
from RA-node	inferring a conclusion in the form of information	inferring a conclusion in the form of an inference application	inferring a conclusion in the form of a conflict application
from CA-node	predecessor of CA is in conflict with information	predecessor of CA is in conflict with inference application	predecessor of the CA is in conflict with conflict

Table 1. Edge types in the AIF

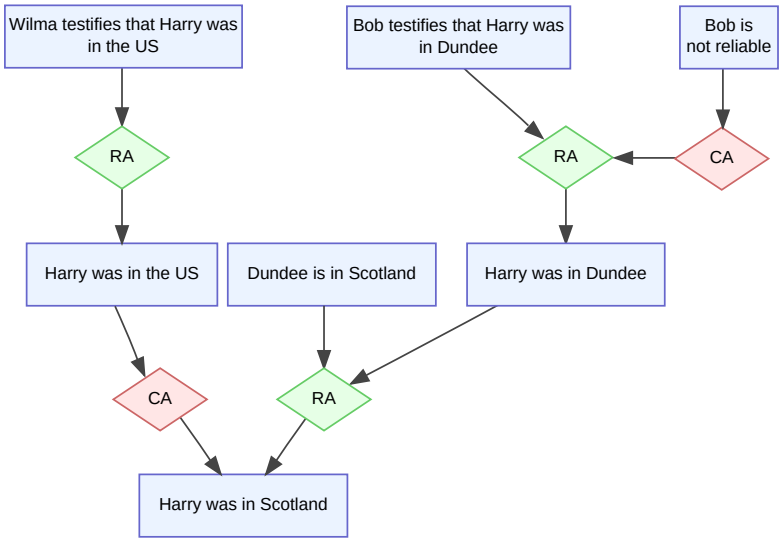

Figure 1. An AIF argument graph

Another example is AIFdb[1], a relational database definition [8] which mirrors the specification in OWL but is designed to simplify the scaling issues of the large-scale argument infrastructure of the Argument Web [10]. AIFdb allows for the storage and retrieval of AIF compliant argument structures and offers a rich array of web service interfaces allowing for interaction with low level argument components (nodes, edges, schemes), as well as modules which handle the import and export of numerous formats such as SVG, DOT, RDF-XML and the formats of the Carneades, Rationale and Araucaria tools. To facilitate import and export the database also allows for the creation of *node sets* which can be used to perform operations on a specific grouping of argument components.

3. Carneades Argument Graphs

Carneades is an open source argumentation system[2] [6] for argument construction and evaluation originally inspired by legal reasoning but more generally applicable. It works

[1] http://www.arg.dundee.ac.uk/AIFdb/

[2] http://carneades.github.com. The version of the Carneades editor used was 1.0.2, the newest version publicly available at the time of writing. A new version is currently under development

with an abstract model of argument graphs, called *Carneades Argument Graphs* (CAG) [6], which is comparable to AIF and indeed was inspired in part by AIF [5]. In a CAG there are nodes representing *statements* and nodes representing *arguments*, the latter of which instantiate argumentation schemes defined in a separate rule language.

Definition 3.1 [Carneades Argument Graph]
A *Carneades Argument Graph* CAG is a bipartite graph consisting of statements and arguments.

1. Each argument is of one type either pro or con;
2. Each argument a has zero or more premises of the form ($polarity$, $type$, $statement$) and one conclusion of the form $statement$, where
   (a) $polarity$ is either pos or neg;
   (b) $type$ is one of $ordinary$, $assumption$, $exception$;
   (c) $statement$ is the statement that serves as the premise or conclusion

The premises of pro- and con-arguments are reasons for and against the conclusion, respectively. A key feature of Carneades is that it provides a natural account of reasoning under burden of proof. To allow this burden of proof to be distributed, CAGs distinguish several types of premises, namely ordinary premises (which have to be shown to hold before the conclusion can be inferred), assumption premises (which are assumed to hold until questioned) and exception premises (which place the burden of production on the respondent). Premises can be either positive or negative, where a negative premise is the negation of the statement of the premise.

This model of argument graphs in Carneades has been relatively stable throughout the literature [6,7,5]. A new version of Carneades, which uses an updated version of the CAG data structure, is currently under development. The most important difference is that the distinction between different types of premises (ordinary, exception, assumption) has been removed and that exceptions are modelled in a more conventional way, namely as undercutters. For this paper, however, we use the version of CAGs that at the time of writing was published and publicly available and we leave the translation of this new model for future research.

Figure 2 shows a CAG version of the example AIF argument graph of Figure 1. In this visualization, statement nodes are shown with boxes and argument nodes with circles. The type of premise is shown on the link to the argument node. Pro-arguments are rendered as '+' arguments whilst con-arguments are rendered as '–' arguments.

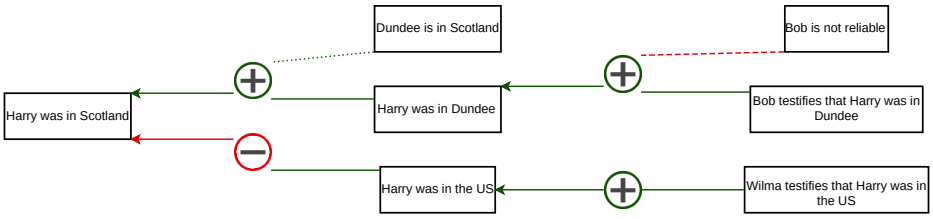

Figure 2. A CAG version of the example

Like AIF, the CAG model is specified in different ways for different purposes. For example, Carneades Argument Evaluation Structures (CAES) [7] define argument graphs

in a logical formalism, which allows for evaluation of the arguments in the graph, and the Carneades system itself produces LKIF files. LKIF, short for Legal Knowledge Interchange Format, is an XML schema developed with the aim of facilitating the interchange of legal rules and legal arguments [5]. LKIF consists of a rule language for representing argumentation schemes and legal norms and an argument language for representing CAGs[3]. It is the argument part, the part of LKIF that serialises the abstract model of CAG as XML, that we are interested in. As in Definition 3.1, an LKIF file specifies statements and arguments. In addition, it specifies some extra attributes (e.g. the title of a CAG, the id of a node) which are not directly interesting for our purposes.

4. Translating between AIF and CAG

AIFdb has a web service interface for the import and export of LKIF files. To import a file, it is posted to the database[4] and the XML is parsed using a standard parser. The translation from CAG to AIF then proceeds as in algorithm 1. It is also possible to save arguments in the AIFdb as LKIF files. The translation from AIF to CAG is shown in algorithm 2. Arrays and variables are denoted with a '*$*', where an array starts with a capital letter (i.e. *$Array*) and a variable with a lower case letter (i.e. *$var*).

Lines 1-22 in algorithm 1 translate the statements in the CAG to AIF; statements that are used positively and negatively are first put into two separate lists (line 1-10). Because the concept of negative premises does not exist in AIF, statements that appear as such are then translated by inserting the text 'it is not the case that' (lines 15, 20). Furthermore, if the negative premise is also used as a positive premise or conclusion, the algorithm inserts symmetrical conflict relations to indicate the logical conflict between a statement and its negation (lines 15-17). Lines 23-44 then handle the translation of the arguments from CAG to AIF. *Con* arguments are translated as CA-nodes (line 25) and *pro* arguments as RA-nodes (line 28). Exception premises in CAG are then translated as undercutters in AIF (lines 37-39, cf. figure 1, 2). Translating from AIF to CAG is slightly more straightforward (algorithm 2). I-nodes are translated into statements (line 2,3). S-nodes with an edge to an I-node are translated as standard arguments (lines 4-11). The predecessors of these S-nodes are then either premises (in the case of I-nodes, lines 15, 16) or exceptions (in the case of CA-node undercutters, lines 18, 19).

If we assume the original AIF graph does not have preferences, the translation from AIF to CAG and back gives us exactly the original AIF graph. Note that while both the Carneades editor and the CAES framework cannot handle cycles, CAG and LKIF can and cyclic conflict is translated as two *con*-arguments. There are situations in which the translation from CAG to AIF and back does not give us exactly the same CAG structure. Arguments with zero premises are left untranslated, as AIF cannot represent 'loose' S-nodes. Furthermore, AIF nodes do not have a 'negative' property so all I-nodes will be translated as positive premises even if they originally were negative. The information content is the same, however: where the original CAG contained a negative premise p, the CAG that's exported from AIF will have a positive premise *it is not the case that* p. Furthermore, recall that if the original CAG contains the same proposition as both a

[3]The new version of Carneades, which is under development, has a separate format for argument graphs called Carneades Argument Format (CAF)

[4]`http://www.arg.dundee.ac.uk/AIFdb_upload/`

Algorithm 1 Algorithm for translating CAG to AIF

for all argument a **do**
  add conclusion(a) to *$Pos*
  for all p = premise(a) **do**
    if polarity(p) = *pos* **then**
      add statement of p to *$Pos*
    else
      add statement of p to *$Neg*
    end if
  end for
end for
for all statement s **do**
  if s in *$Pos* **then**
    create I-node n with text of s
    if s in *$Neg* **then**
      create I-node n' with text 'It is not the case that [text of s]'
      create two CA-nodes ca_i, ca_j
      create edges (n,ca_i), (ca_i, n'), (n',ca_j), (ca_j,n)
    end if
  else
    create I-node n with text 'It is not the case that [text of s]'
  end if
end for
for all argument a **do**
  if direction(a) = con **then**
    create CA-node ca and **set** *$argS* to ca
    create edge from ca to the node corresponding to conclusion(a)
  else
    create RA-node ra and **set** *$argS* to ra
    create edge from ra to the node corresponding to conclusion(a)
  end if
  for all p = premise(a) **do**
    if polarity(p) = *pos* **then**
      set *$prem* to node corresponding to the statement of p
    else
      set *$prem* to node corresponding to the negation of the statement of p
    end if
    if type(p)= $exception$ **then**
      create new CA-node ca'
      create edge (ca', *$argS*) and create edge (*$prem*, ca')
    else
      create edge (*$prem*, *$argS*)
    end if
  end for
end for

Algorithm 2 Algorithm for translating AIF to CAG

```
for all node n do
    if type(n) = I then
        create statement s with text of n
    else if type(n) = CA or type(n) = RA then
        set $conc to successor(n)
        set $Prem to predecessors(n)
        if type($conc) = I then
            if type(n) = CA then
                create argument a, direction(a) = con
            else
                create argument a, direction(a) = pro
            end if
            create conclusion $conc for a
            for all p in $Prem do
                if type(p) = I then
                    create premise (pos, ordinary, s) for a (s is statement of p)
                else
                    set $root to predecessor(p)
                    create premise (pos, exception, s') for a (s' is statement of $root)
                end if
            end for
        end if
    end if
end for
```

positive and a negative premise this will introduce a conflict cycle in the AIF, which will be translated back into CAG as two *con*-arguments which were not in the original CAG. Finally, both assumptions and ordinary premises are translated into AIF as I-nodes so the distinction between them is effectively lost in translation.

5. Conclusions and future research

Recently, there has been an increased interest in translations from Carneades to other formal logical frameworks for argumentation (e.g. [14]). This other work, however, only defines a translation between two theoretical (logical) frameworks and does not, like the current work, define and implement algorithms for translating between abstract data structures for argument graphs (i.e. AIF graphs and CAGs). Furthermore, the translations in [14] are only meant to ensure that the same statements will be acceptable in both the CAG and the ASPIC$^+$ framework. In contrast, the objective of AIF is to make it easier to translate among different models of argument structure and to help clarify differences in how different models represent argumentative information.

A certain degree of isomorphism allows for practical interchange between Carneades and systems compatible with the Argument Web such as Araucaria and Rationale. As a result, it is, for example, possible to manipulate some of the 2,000 or so argument resources in the argument web in Carneades, and visualise the result in Rationale; it is

possible to extend the Sacco and Vanzetti case analysed in Araucaria [2] with explicit indication of burdens of proof in Carneades; it is possible to 'argublog' [12] in response to arguments analysed in Carneades, and compute the acceptability of the result using ASPIC+. The barriers between domains of argumentation (legal and medical, for example) are being broken down as effectively as the barrier between systems and theories of argumentation, and as these barriers come down, a foundation is laid for realising the vision of an open, integrated Argument Web.

Acknowledgements

This work was initiated by Tom Gordon's visit to Dundee, which was supported by the University of Dundee's School of Computing and Fraunhofer FOKUS in Berlin. We further acknowledge the support of the EPSRC under grant EP/G060347/1 for this work.

References

[1] F.J. Bex, S. Modgil, H. Prakken, and C.A. Reed. On logical reifications of the argument interchange format. *Journal of Logic and Computation*, 2012. To Appear.

[2] F.J. Bex, H. Prakken, C.A. Reed, and D.N. Walton. Towards a Formal Account of Reasoning about Evidence: Argumentation Schemes and Generalisations. *Artificial Intelligence and Law*, 11(2/3):125–165, 2003.

[3] F.J. Bex and C.A. Reed. Schemes of Inference, Conflict and Preference in a Computational Model of Argument. *Studies in Logic, Grammar and Rhetoric*, 36, 2011.

[4] 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*, 21:293–316, 2006.

[5] T.F. Gordon. The legal knowledge interchange format - estrella deliverable d4.1. Technical report, Fraunhofer FOKUS, Berlin, 2012.

[6] T.F. Gordon, H. Prakken, and D.N. Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171:875–896, 2007.

[7] T.F. Gordon and D. Walton. Proof Burdens and Standards. In I. Rahwan and G. Simari, editors, *Argumentation in Artificial Intelligence*, pages 239–260. Springer, 2009.

[8] J. Lawrence, F. Bex, M. Snaith, and C. Reed. Aifdb: Infrastructure for the argument web. Submitted to COMMA 2012 demonstration track, 2012.

[9] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1:93–124, 2010.

[10] I. Rahwan, F. Zablith, and C. Reed. Laying the foundations for a world wide argument web. *Artificial Intelligence*, 171:897–921, 2007.

[11] C.A. Reed and G. Rowe. Araucaria: Software for Argument Analysis, Diagramming and Representation. *International Journal of AI Tools*, 13(4):961–980, 2004.

[12] M. Snaith, F.J. Bex, J. Lawrence, and C. Reed. Implementing argublogging. Submitted to COMMA 2012 demonstration track, 2012.

[13] T. Van Gelder. The rationale for rationale. *Law, Probability and Risk*, 6:23–42, 2007.

[14] B. van Gijzel and H. Prakken. Relating carneades with abstract argumentation via the aspic+ framework for structured argumentation. *Argument and Computation*, 2012. To Appear.