Moving Between Argumentation Frameworks

Nir OREN [a,1], Chris REED [b] and Michael LUCK [a]
[a] *Dept. of Computer Science, King's College London, UK*
[b] *Dept. of Computer Science, University of Dundee, Scotland*

Abstract. Abstract argument frameworks have been used for various applications within multi-agent systems, including reasoning and negotiation. Different argument frameworks make use of different inter-argument relations and semantics to identify some subset of arguments as coherent, yet there is no easy way to map between these frameworks; most commonly, this is done manually according to human intuition. In response, in this paper, we show how a set of arguments described using Dung's or Nielsen's argument frameworks can be mapped from and to an argument framework that includes both attack and support relations. This mapping preserves the framework's semantics in the sense that an argument deemed coherent in one framework is coherent in the other under a related semantics. Interestingly, this translation is not unique, with one set of arguments in the support based framework mapping to multiple argument sets within the attack only framework. Additionally, we show how EAF can be mapped into a subset of the argument interchange format (AIF). By using this mapping, any other argument framework using this subset of AIF can be translated into a DAF while preserving its semantics.

Keywords. Argumentation, Abstract Argument Frameworks, Semantics

1. Introduction

Typical applications of argumentation theory represent background knowledge and facts about the world as arguments, and reach some decision (e.g. what price to name in a negotiation) based on the interactions between these arguments. In many cases, reaching a decision depends on identifying which subsets of the entire set of arguments are, in some sense, coherent. Abstract argument frameworks model sets of arguments as atomic entities, ignoring their inner structure, and concerning themselves only with the interactions between arguments. Such argument frameworks then identify a coherent set of arguments according to some semantics, based on the interactions between the arguments.

For example, Dung's argument framework [7] makes use of the notion of an attack between arguments, and identifies a set of arguments as coherent if a skeptical reasoner would believe they are coherent (in the case of the *grounded* extension). Within the argumentation literature, a plethora of argument frameworks have been proposed, capable of modelling not only attacks between arguments, but also support [2,12] and prefer-

[1]Corresponding Author: Dept. of Computer Science, King's College London, UK; E-mail: nir.oren@kcl.ac.uk

ence and value orderings [1,3]. Each argument framework, and its associated semantics, claims to capture some novel aspect of argument interaction that other frameworks do not. However, little attention has been paid to the possibility of translating between argument frameworks *at the semantic level*. Yet such translation is critical in an open system where agents make use of argumentation; by translating, entities making use of different underlying argumentation frameworks can agree on the status of an argument, justifying why some decision was taken.

In this paper, we begin addressing this translation problem by describing how arguments represented using Oren et al's Evidential Argument Framework [12] can be translated into Dung's abstract argument framework [7], and vice-versa. This translation is designed to be semantics preserving. That is, given some set of arguments deemed coherent in one framework under some specific semantics, the same set of arguments should be deemed consistent in the other framework under a similar semantics. As an added benefit, this translation procedure allows us to trivially translate from Nielsen's framework [11] to the other two frameworks (Translating from Dung's semantics to Nielsen's is trivial, and we may thus freely translate between the three frameworks).

As an example of the use of translation, we consider the AIF standard [6], which was created in order to allow for the interchange of arguments between different systems, each of which may use a different internal argument representation, and a different framework for reasoning. AIF is RDF based, and is purely representational, therefore not yielding to standard acceptability semantics, and has no defined operational semantics. However, it is capable of representing concepts found in many different frameworks, including attacks, supports and preference, as well as more complex notions such as argument schemes.Evidential argument frameworks (EAFs) provide an intuitively appealing representation of a subset of AIF, and our work thus provides the tantalising suggestion of translating from some argumentation framework into AIF, then from AIF into EAFs, and finally from EAFs into Dung's framework. This translation thus allows for linkages between many different frameworks.

The main contribution of this paper lies in identifying a mapping between Oren's, Dung's and Nielsen's frameworks, and providing an operationalisation of this mapping. As an application of this mapping, we show how EAFs may be represented as a subset of AIF, allowing any other argument framework described using this AIF subset to be mapped into Dung and Nielsen's frameworks.

In the next section, we provide an overview of the frameworks we examine in depth in this paper, namely Dung's argument framework, Nielsen's extensions to it, and Oren's evidential argument framework. Section 3 then describes a simple algorithm to translate between the frameworks, following which refinements to the basic algorithm are introduced. Finally, we show how a mapping from AIF to Oren's framework may be created, after which we discuss related work, and possible paths for future research.

2. Background

Abstract argument frameworks do not concern themselves with the internal structure of an argument, instead focusing on the interactions between sets of arguments. Thus, for example, an argument "Nixon should not invade Vietnam because it would start a war he could never win", could be represented by the argument a. Dung's seminal argument

framework consists of a set of arguments, and one possible type of interaction between them, namely by attacking each other.

Definition 1 ***(Dung Argument Framework)*** *A Dung argument framework (DAF) is a tuple* $DAF = (Args, Attacks)$ *where* $Args$ *is a set of arguments, and* $Attacks : Args \times Args$ *is a binary relation.*

An argument a is said to attack another argument b if $(a, b) \in Attacks$. From this simple representation, we may define a number of notions:

Definition 2 ***(Auxiliary Notions for DAFs)*** *Given a Dung argument framework* $(Args, Attacks)$*, a set of arguments* $S \subseteq Args$ *is* conflict free *iff* $\forall a, b \in S, (a, b) \notin Attacks$*.*
An argument $a \in Args$ *is* acceptable *with respect to a set of arguments* $S \subseteq Args$ *iff* $\forall b \in Args$ *such that* $(b, a) \in Attacks$*,* $\exists c \in S$ *such that* c *attacks* b*.*
A conflict free set of arguments S *is* admissible *iff all its elements are acceptable w.r.t* S*.*

Definition 3 ***(Semantics for DAFs)*** *Given a DAF, a set of arguments is said to be a* preferred extension *if it is a maximal (w.r.t. set inclusion) admissible set of arguments.*
A set of arguments S *is a* stable extension *iff* $S = \{a | a \in Args$ *and* a *is not attacked by* $S\}$*.*
A set of arguments S *is a* grounded extension *if it is the least fixed point of the function* $F_{AF}(S) = \{a | a$ *is acceptable with respect to* $S\}$*.*

Nielsen [11] popularised a simple, but important extension to DAFs. Instead of attacks operating between single arguments, an attack may require a set of arguments to take place. While it has been claimed (but not proven) that DAFs can model such cases, Nielsen's framework allows these situations to be represented more compactly, and without the need to introduce additional, *virtual* arguments into the system.

Definition 4 ***(Nielsen's Argument Framework)*** *An argument framework in Nielsen's extension to DAF, denoted a NAF, is a tuple* $NAF = (Args, NAttacks)$ *where* $Args$ *is a set of arguments, and* $NAttacks : 2^{Args} \times Args$ *is the attacks relation.*

Concepts such as acceptability, conflict freeness and admissibility, originally defined on DAFs are directly translatable to NAFs. By translating these concepts, extensions can be defined on NAFs which are analogous to DAF extensions.

Oren's[12] evidential argument framework (EAF) is another framework based on Dung's work, which also makes use of Nielsen's extensions to DAFs. Like a DAF, an EAF consists of a graph containing nodes representing arguments. An EAF also contains an additional, special argument η, which represents a default, or some form of unquestionable evidence. Unlike a DAF, two types of edges exist between nodes, the first associated with the attack relation, while the second represents support between arguments (one possible interpretation of a support edge from η to an argument is that the supported argument is true by default, or that some unassailable evidence for the argument exists. Support between other arguments can be seen as implying that an inferential link exists between them).

Definition 5 ***(Evidential Argumentation Systems)***

An evidential argumentation system is a tuple (A, R_a, R_e) *where* A *is a set of arguments,* R_a *and* R_e *are relations of the form* $(2^A \setminus \emptyset) \times A$*, and that within the argu-*

mentation system, $\nexists x \in 2^A, y \in A$ such that $xR_a y$ and $xR_e y$. We assume the existence of a "special" argument $\eta \in A$, such that $\nexists(x, y) \in R_a$ where $\eta \in x$; and $\nexists x$ where $(x, \eta) \in R_a$ or $(x, \eta) \in R_e$.

The introduction of the support relation means that concepts such as a successful attack between arguments, acceptability and admissibility, are computed in a different manner to a DAF.

A necessary condition for an argument to appear in an extension is that it is directly, or indirectly supported by some evidence; that is, there is a path from η to the argument, according to the edges of the R_e relation.

Definition 6 ***(Evidential Support)*** *An argument a has evidential support from a set S iff either $SR_e a$ where $S = \{\eta\}$ or*

$\exists T \subset S$ such that $TR_e a$ and $\forall x \in T$, x has evidential support from $S \setminus \{x\}$.

S is a minimum support for a if a has evidential support from S and there is no $T \subset S$ such that a is supported by T.

If a has evidential support from S, we may say that S e-supports a, or that a is e-supported by S. Where obvious, we abbreviate e-support to support. The notion of attack requires the attacking argument to be supported.

Definition 7 ***(Evidence Supported Attack)*** *A set S carries out an evidence supported attack on an argument a if $XR_a a$ where $X \subseteq S$, and, all elements $x \in X$ are supported by S.*

An evidence supported attack by a set S on a is minimal iff S carries out an evidence supported attack on a, and there is no $T \subset S$ such that T carries out an evidence supported attack on a.

From these basic concepts, we can define a number of auxiliary notions

Definition 8 ***(Auxiliary Notions for EAFs)*** *An argument a is acceptable with respect to a set S iff S e-supports a, and for any minimal evidence-supported attack by a set $X \subseteq 2^A$ against a, $\exists T \subseteq S$ such that $TR_a x$, where $x \in X$ so that $X \setminus \{x\}$ is no longer an evidence-supported attack on a.*
A set of arguments S is conflict free iff $\forall y \in S$, $\nexists X \subseteq S$such that $XR_a y$.
A set of arguments S is self supporting iff $\forall x \in S$, S e-supports x.

As in DAFs, a set of arguments is admissible if it acceptable and conflict free. We can then define semantics for EAFs in an identical manner as for DAFs. For example, a maximal admissible set of arguments is a member of the e-preferred extension.

Having provided a brief survey of DAFs, NAFs and EAFs, we proceed to show how a set of arguments can be converted from one framework to another, with arguments found in an extension in the original framework contained in an analogous extension in the new framework.

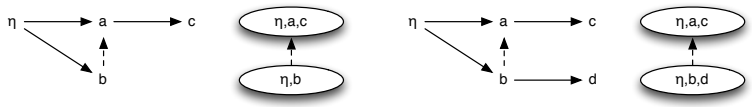

Figure 1. Some possible mappings from an EAF (left) to an equivalent DAF (right). Solid lines represent support between arguments, dashed lines indicate attacks.

3. Converting Between Argument Frameworks

In this section, we show how sets of arguments may be represented in different argument frameworks. We show how DAFs may be converted to NAFs, and from there to an EAF, and then show how an EAF may be converted to a DAF.

3.1. Converting a DAF/NAF to an EAF

Converting an argument system from a DAF to a NAF is trivial, with the source argument set in the attack relation in the NAF containing only the source attacking argument in the DAF. For example, an attack on b by a in a DAF would be mapped to an attack on b from $\{a\}$ in the analogous NAF.

To convert from a NAF to an EAF is also easy. The EAF contains all arguments and attacks from the NAF, together with argument η. η then supports all other arguments in the NAF. This is due to the fact that without any extra information, we must assume that all arguments in the original argument system are either true by default, or have some evidence to support them. Conversion from a DAF to an EAF can then take place by first converting the DAF to a NAF, and then converting the NAF to an EAF.

Clearly, any argument attacked in the original argument framework is evidence support attacked in the EAF. Also, if an argument is acceptable in the original framework with respect to a set S, it is acceptable in the EAF with respect to the set $S \cup \eta$. Similarly, the notion of conflict free remains the same, implying that any argument admissible in the DAF or NAF is also admissible in the EAF. Given this, any argument in a preferred/stable/grounded extension is in the e-preferred/e-stable/e-grounded extension, and any argument not in a DAF or NAF extension is not in the EAF extension.

3.2. Converting an EAF to a DAF

The notion of support, as well as the ability of multiple arguments to attack or support a single argument, makes mapping from EAFs and DAFs/NAFs more difficult than mapping from DAFs/NAFs to EAFs. In this section, we show a many to one mapping between EAFs and DAFs. This mapping is many to one in the sense that many different EAFs can be converted to a DAF with identical graph structure. The goal of this conversion is to preserve the EAF's semantics. That is, any argument that is within some extension within the EAF should be in an analogous extension with the DAF.

Our approach centres around the mapping of a set of related arguments — informally arguments taken together with their supporting arguments, recursively back to η — into a single DAF argument[2]. Consider, for example, the EAF and DAF shown on the left of

[2]If viewed as an argument/subargument relationship, then the ideas of [8] are highly applicable.

Algorithm 1 A simple algorithm to create a DAF with the same semantics as the EAF.

Require: An EAF $(Args, R_a, R_e)$
1: $DARGS = \{\}$ %DAF arguments
2: $DATT = \{\}$ %DAF attacks
3: **for all** $A \in 2^{Args}$ **do**
4: **if** A is self supporting **then**
5: Add A to $DARGS$
6: **end if**
7: **end for**
8: **for all** (X, a) such that $X R_a a$ **do**
9: **for all** $D \in DARGS$ such that $X \subseteq D$ and $A \in DARGS$ such that $a \in A$ **do**
10: Add (D, A) to $DATT$
11: **end for**
12: **end for**
13: **return** (DARGS,DATT)

Figure 1. Intuitively, This EAF can be converted into the DAF shown to the right of it[3]; arguments η, a, c from the EAF are grouped into one argument within the DAF, while η, b are grouped into another. This means that if η, b are found in some extension of the EAF, they also appear in an equivalent extension of the DAF, and vice-versa. The same holds for the remaining arguments, and thus, in the DAF, we have a single argument η, a, c composed of the EAF arguments η, a and c. The right hand EAF shown in Figure 1 maps to a DAF with the same structure as the left hand EAF in that figure. Thus, multiple EAFs can be represented as a DAF with identical graph structure.

We begin by describing a simple approach to performing the conversion between EAFs and DAFs. We then examine the properties of this conversion, to show that the semantics of the EAF are preserved when transformed into a DAF. In Section 4, we describe refinements to this simple approach.

Algorithm 1 details the basic approach. This algorithm operates by generating all possible self supporting argument sets (SSAS), and utilises these SSASs as the resultant DAF's atomic arguments. Attacks in the DAF are generated according to whether an attack exists between elements of the two SSASs (Lines 8–12).

We can show two simple, but important properties for the algorithm, namely that attacks carry over between the EAF and DAF, and that admissibility also carries over.

Lemma 1 *If an argument is e-support attacked in the original EAF, any SSAS containing the argument is still attacked in the resultant DAF.*
Proof: *An e-supported attack must be part of a SSAS; since only self supporting sets of arguments are copied into the DAF, all such e-supported attacks are copied (Line 5). Of course, if the attacked argument is not part of a SSAS, the attack against it are not be copied, but in this case, there is no possibility that the argument is acceptable as acceptability within the EAF requires support.* □

Lemma 2 *Given an EAF and the DAF derived from it according to Algorithm 1, a set of SSASs S' is admissible within the DAF if and only if all arguments found within $\bigcup S'$ (i.e. all arguments found in the set of SSASs) are admissible within the EAF.*

[3]When drawing EAFs and DAFs, arguments in the DAF are enclosed within ellipses. We omit set notation in the source of support and attack relations in the EAF if the source of the support or attack is a single argument.

Proof: If: *Clearly, since S is admissible, there is some set of SSASs S' within the DAF made up of subsets of S. Since S is conflict free, $S'$is also conflict free. Thus, to show S' is admissible, we must show that it is acceptable. In other words, given a SSAS that attacks some $s' \in S'$, we must show that there is some SSAS s'', made up of subsets of S, that attacks s'.*

Now, within the EAF, for any attack XR_as where $X \subseteq Y$ and Y is a minimal self supported set, for s to be acceptable w.r.t S, there must be some $T \subseteq S$ that (e-support) attacks an element of X, or (e-support) attacks an element of Y, thus causing the original attack to no longer be e-supported.

Since, within the EAF, all attacks against $s \in S$ are e-supported, there are sets of SSASs $Y_1 \ldots Y_n$ within the DAF attacking all SSASs containing s.

Since s is acceptable with respect to S, and S is self supporting, there must be some subsets of arguments $T_i \subseteq S$ that carry out an e-supported attack against Y_j for all $j = 1 \ldots n$.

Thus, if a set S is admissible within an EAF, there is a set of SSASs consisting of subsets of S that are admissible within the DAF.

Only if *clearly, S is self supporting. Furthermore, since S' is conflict free, there are no attacks between its elements, and S is thus also conflict free. Finally, since S' is admissible, any attack against its members, say by a SSAS T, is attacked by a member of S'. Since T is self supporting, the attack against T must either directly attack an argument that attacked an argument of S, or render T not self-supporting. Thus, each member of S is acceptable with respect to S, and S is thus admissible.* □

This result means that any semantics based on admissibility (such as the grounded and preferred semantics) are preserved by Algorithm 1 when moving from an EAF to a DAF. The carry over of attacks also means that the stable semantics is preserved by the translation process.

4. Eliminating Redundant Sets of Arguments

Given an EAF containing a minimal self supporting set of n arguments with no attacks against it, it is clear that on the order of 2^n SSASs can be formed when using the simple algorithm described above. All of these 2^n SSASs would not be attacked in the resultant DAF, and if so all appear in the admissible set, with all of their component arguments being admissible in the original EAF. The removal of all but the maximal SSAS (with respect to set inclusion) from the DAF has no impact on the arguments in the final group of admissible sets within the DAF, and thus, no information is lost if all non-maximal SSASs are removed from the DAF. In this section, we examine which SSASs must be kept so as to convey the information found in the original EAF.

Consider a self supporting set of arguments $\{\eta, a, b\}$ found as part of an EAF. If no attacks exist against this argument set, then, when converted into a DAF, it forms part of the (for example) preferred extension, as do the SSASs $\{\eta, a\}$ and $\{\eta\}$. The latter two SSASs are thus, in a sense, redundant. In fact, the only time a subset of the maximal SSAS can be found in an extension while the maximal SSAS may not be is when an argument within the maximal SSAS is attacked. Now instead, consider the argument set $\{\eta, a, b, c\}$, and assume that a, which is supported by η, supports b which in turn supports c. Now assume that an attack exists against argument b. In such a situation only two

Algorithm 2 An algorithm for converting from an EAF to a DAF.

Require: An EAF $(Args, R_a, R_e)$
AC={all maximal self supporting argument sets of the EAF},Att={}
for all $(S, t) \in R_a$ **do**
  Let $TEAF = (Args \backslash \{t\}, R_a, R_e)$
  Add all maximal self supporting chains of $TEAF$ to AC
end for
for all $C \in AC$ **do**
  for all $D \in AC$ **do**
    if $(S, t) \in R_a$ such that $S \subseteq C, t \in D$ **then**
      Add (c, d) to Att
    end if
  end for
end for
return (AC, Att)

argument SSASs need be considered, namely the full SSAS (containing η, a, b and c) and the SSAS $\{a, \eta\}$. If the full SSAS is not in the extension, but the latter SSAS is, then $\{\eta\}$ is also in the extension, while $\{\eta, a, b\}$ will not be present. More formally (where by *status*, we mean whether the SSAS is admissible or not):

Lemma 3 *Given a SSAS $AS = \{a_1, \ldots, a_n\}$ and an attack against a_i,*

1. *All self supporting subsets that can be formed from $AS \backslash \{a_i\}$ have the same status (we label these subsets S_{i-1}).*
2. *All subsets of $AS \backslash S_{i-1}$ have the same status.*

Proof: *The admissibility of a SSAS depends on whether it is attacked or not. Since S_{i-1} are not attacked, their status is identical. Similarly, since all subsets of $AC \backslash S_{i-1}$ are attacked, their status is identical.* □

Thus, a DAF containing only maximal SSASs, as well as subsets of the maximal SSASs, up to the point in which the set is attacked, is sufficient to represent the original EAF system. This result allows us to propose Algorithm 2 for converting from an EAF to a DAF. This algorithm, as with the algorithms proposed later, runs in polynomial time. While correct in the sense that identical arguments appear in admissible sets within both the EAF and DAF, This algorithm does yield some unintuitive results. Consider, for example, the following EAF system:

$$(\{\eta, a, b, c\}, \{(\eta, a), (\eta, b), (a, c)\}, \{(b, a)\}) \tag{1}$$

As shown on the left of Figure 2, applying this algorithm results in the DAF $(\{\alpha, \beta\}, \{(\alpha, \beta), (\beta, \beta)\})$. Here, $\alpha = \{\eta, b\}$ and $\beta = \{\eta, a, b, c\}$. As expected, η and b are admissible in both the DAF and EAF. However, as shown on the right of Figure 2, if $\beta = \{\eta, a, c\}$, we obtain the same set of admissible arguments, and this representation of the interactions between arguments more closely agrees with our intuitions.

The fact that there is initially one maximal argument set, starting at η, is the source of this problem. In order to overcome this issue, we must adapt the manner in which we make use of η. For example, we could allow (conceptually) different versions of η to support different arguments, which in turn would allow for distinct sets of maximal SSASs to appear. In such a situation, each support relation from η to another argument could

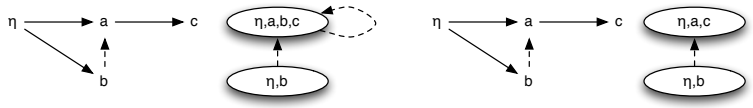

Figure 2. The EAF and the DAF that results from applying Algorithm 2 (left) and Algorithm 3 (right).

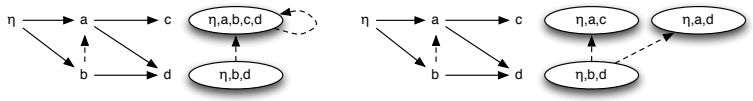

Figure 3. The EAF and the DAF that results from applying Algorithm 3 (left) and Algorithm 5 (right).

Algorithm 3 An algorithm for computing maximal argument sets.

Require: $AF = (Args, R_a, R_e)$
1: $MaximalChains = \{\}$
2: **for all** $a \in Args$ **do**
3: $maxChain = computeMaxSupportedSet(a, \{\eta\}, AF)$
4: **if** $maxChain \cup \{\eta\}$ is a self supporting chain **then**
5: $MaximalChains = MaximalChains \cup maxChain \cup \{\eta\}$
6: **end if**
7: **end for**
8: **return** $MaximalChains$

be replaced by a support from some new, unique, non-attacked, argument to the other argument. To achieve this, we change Lines 1 and 4 of Algorithm 2 to make use of Algorithm 3 when computing the SSASs. Constructing a maximal SSAS according to this algorithm "ignores" η when forming argument sets. However, this is in fact equivalent to making use of a unique η for each SSAS.

When evaluated using the modified form of Algorithm 2, The EAF from Equation 1 results in the DAF $(\{\alpha, \beta\}, \{(\alpha, \beta)\})$. Now, $\alpha = \{\eta, b\}$ and $\beta = \{\eta, a, c\}$ (as shown on the right of Figure 2), agreeing with our intuition. However, this algorithm still yields counter-intuitive results in some situations. For example, given the EAF

$$(\{a, b, c, d\}, \{(\eta, a), (\eta, b), (a, d), (a, c), (b, d)\}, \{(b, a)\}) \tag{2}$$

this algorithm would result in a DAF containing SSASs $\{\eta, a, b, c, d\}$ and $\{\eta, b, d\}$, with attacks between the latter and former, and the first node attacking itself (shown on the left of Figure 3). The independence of support between a, c and a, d means that a, c and a, b, d should appear as separate nodes within the SSAS.

By assuming that a SSAS has a single argument as its conclusion, i.e. that a SSAS ultimately provides support for only a single argument, we can use Algorithm 5 to create a DAF from an EAF in a more intuitive manner. This algorithm computes (using the $computeBackSet()$ described in Algorithm 6) all possible SSASs that have some argument a as their single conclusion, for all arguments in the system. Any SSAS not containing η is removed. Finally, any SSAS that is a subset of another SSAS is removed, leaving maximal SSASs, which can then be used in Lines 1 and 4 of Algorithm 2.

Algorithm 4 $computeMaxSupportedSet(a, Visited, AF)$

Require: An argument a, a set of nodes $Visited \subseteq Args$, an EAF $AF = (Args, R_a, R_e)$

$Answer = \{a\}, ToVisit = \{\}, Visited = Visited \cup \{a\}$
for all $B \subseteq Args$ such that $(B, a) \in R_e$ **do**
  $ToVisit = ToVisit \cup B$
end for
for all $c \in Args$ such that $(\{\ldots, a, \ldots\}, c) \in R_e$ **do**
  $ToVisit = ToVisit \cup c$
end for
for all $d \in ToVisit \backslash Visited$ **do**
  $Answer = Answer \cup computeMaxSupportedSet(d, Visited, AF)$
end for
return $Answer$

Algorithm 5 Algorithm to generate a DAF from an EAF.

Require: An EAF $AF = (Args, R_a, R_e)$

for all $a \in Args$ **do**
  $Answer = Answer \cup computeBackSet(a, \{\}, AF)$
end for
for all $AS \in Answer$ **do**
  if $\eta \notin AS$ **then**
    $Answer = Answer \backslash \{AS\}$
  end if
  if $\exists AS' \in Answer$ such that $AS \subseteq AS'$ **then**
    $Answer = Answer \backslash \{AS\}$
  end if
end for
return $Answer$

Algorithm 6 $computeBackSet(a, Visited, AF)$

Require: An argument a, a set of visited edges $Visited \subseteq R_e$, an EAF $AF = (Args, R_a, R_e)$

for all $(X, a) \in R_e$ such that $(X, a) \notin Visited$ **do**
  $B = \times_{x_i} computeBackSet(x_i, Visited \cup \{(X, a)\}, AF)$ for $x_i \in X$
  for all $b \in B$ **do**
    $b = b \cup \{a\}$
  end for
  $Ans = Ans \cup B$
end for
return Ans

Algorithm 5 returns a DAF containing the SSASs $\{\eta, a, c\}$, $\{\eta, a, d\}$ and $\{\eta, b, d\}$ as arguments when evaluated on the EAF from Equation 2. This, together with the attacks found in the DAF, is illustrated on the right of Figure 3.

5. From AIF to EAF

Having described how translation between DAFs, NAFs and EAFs is possible, we now examine the argument interchange format (AIF), a RDF based ontology for the repre-

sentation of argument related concepts. As discussed below, a simple mapping exists between a subset of AIF and EAF. By utilising this mapping, we can translate between any argument system using this subset of AIF, and the three argumentation frameworks described in this paper. We begin this section by providing a brief introduction to AIF, following which we show how translations between EAFs and AIFs can take place.

AIF assumes that arguments can be represented as nodes in a directed graph, and makes use of two types of nodes: I-nodes, which hold data, and S-nodes, which represent "the inferential passage associated with an argumentative statement" [13]. S-nodes fall into one of three categories: RA-nodes, which represent the application of a rule of inference; PA-nodes, which represent some preference ordering; and CA-nodes, which represent conflict between information. I-Nodes may not have edges linking them to other I-Nodes, while S-Nodes may link to any node. In this paper, we examine a subset of AIF, considering only RA and CA nodes, and assume that S-Nodes are linked only to I-Nodes. While this restriction appears severe, many natural language oriented argument systems, such as Araucaria [14] make use of such simplified AIF graphs.

From the foregoing discussion, it should be clear that there are strong similarities between RA-nodes and support in an EAF, and CA-nodes and the notion of an attack. Moving between an AIF representation of an argument system and an EAF is thus simple: let the set of arguments be the set of I-Nodes (together with the additional argument η); add an element $(\{a_1, \ldots, a_n\}, b)$ into R_a if there is a set of edges from $a_1, \ldots, a_n$ to a CA-Node c, and another edge from c to b; and add an element $(\{a_1, \ldots, a_n\}, b)$ into R_s if there is a set of edges from $a_1, \ldots, a_n$ to a RA-Node c, and another edge from c to b. Finally, add an edge from η to any I-Node which does not have an edge leading to it that originates at an RA-Node. This last step makes the assumption that the I-Node is either true by default, or has some support from unassailable evidence. If the AIF graph encodes such evidential notions, then this last step is not necessary.

Similarly, it is trivial to represent an EAF in AIF; the set of I-Nodes is derived from the set $Args$; for every element of $(A, b) \in R_a$, create an CA-Node with edges going to the CA-Node from all elements of A, and an edge from the CA-Node to b. Finally, a similar operation can be performed when creating RA-Nodes. Any argument framework that may be mapped to the subset of AIF used here can be mapped to an EAF, and thus to a DAF/NAF, and vice-versa.

6. Discussion and Conclusions

In this paper, we have described semantics preserving translations between DAFs, NAFs and EAFs. We also presented a number of algorithms for performing these translations. Different algorithms made different assumptions about the nature of groups of arguments, and yielding different, but equivalent DAFs. The fact that multiple EAFs may be represented as a single DAF is interesting; this result indicates that the notion of support adds information to an argument system which cannot be captured in a DAF alone.

In this work, we focused on the preferred, stable and grounded semantics, due to their widespread use. These semantics are based on the notion of admissibility, and since our results show the equivalence of admissibility between different frameworks, our work is applicable to any other admissibility based semantics. We intend to investigate the effects of translation on additional semantics as part of our future work.

We also intend to extend the translation process to preference and value based argument frameworks [1,3]. Apart from EAFs, bipolar argument frameworks (BAFs) are another approach to including support into abstract argument frameworks [2]. While [5] discusses how EAFs can be translated into BAFs, we hope to investigate how a BAF may be translated into an EAF. Other work that deals with translating between disparate argument frameworks includes [10], which shows how preference type argument frameworks can be represented within his extended argument framework [9]. Finally, in [4], the authors show how a DAF can be directly translated into a bipolar argument framework.

Finally, we showed how EAF can be mapped into a subset of AIF. The ability to map an AIF argument structure into an EAF, and then move from there into other frameworks allows translation between many disparate frameworks while preserving the semantics of the argument set. Two related pieces of future work that we intend to investigate involve examining translations between additional frameworks and EAF, and extending the subset of AIF that we can map into an EAF.

Finally, we intend to examine additional algorithms for performing translation between different systems. As seen here, different approaches make sense when different assumptions are made, and additional algorithms with pleasing intuitive properties may exist for the EAF to DAF case. We would like to see if such algorithms can be identified.

References

[1] L. Amgoud and C. Cayrol. Integrating preference orderings into argument-based reasoning. In *ECSQARU/FAPR '97*, pp 159–170, 1997.

[2] L. Amgoud, C. Cayrol, and M.-C. Lagasquie-Schiex. On the bipolarity in argumentation frameworks. In *Proc. of the 10th Int. Workshop on Non-monotonic Reasoning*, pp 1–9, 2004.

[3] T. Bench-Capon. Value based argumentation frameworks. In *Proc. of the 9th Int. Workshop on Non-monotonic Reasoning*, pp 444–453, 2002.

[4] C. Cayrol and M.-C. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *ECSQUARU '05* volume 3571 of *LNAI*, pp 378–389, 2005.

[5] C. Cayrol and M.-C. Lagasquie-Schiex. Bipolar Abstract Argumentation Systems. In I. Rahwan and G. Simari, editors, *Argumentation in Artif. Intell.*, chapter 4, pp 65–84. Springer, 2009.

[6] C. Chesñevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott. Towards an argument interchange format. *Knowl. Eng. Rev.*, 21(4):293–316, 2006.

[7] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, 1995.

[8] D. C. Martínez, A. J. García, and G. R. Simari. Progressive defeat paths in abstract argumentation frameworks. In *Proc. Canadian Conf. on AI*, volume 4013 of *LNCS*, pp 242–253. 2006.

[9] S. Modgil. An abstract theory of argumentation that accommodates defeasible reasoning about preferences. In *ECSQARU '07*, pp 648–659, 2007.

[10] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artif. Intell.*, 173(9–10):901–934, 2009.

[11] S. H. Nielsen and S. Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *ArgMAS-06*, pp 7–19, 2006.

[12] N. Oren and T. J. Norman. Semantics for evidence-based argumentation. In P. Besnard, S. Doutre, and A. Hunter, editors, *COMMA*, vol. 172 of *Frontiers in Artif. Intell. and Applications*, pp 276–284, 2008.

[13] I. Rahwan, F. Zablith, and C. Reed. Laying the foundations for a world wide argument web. *Artif. Intell.*, 171(10-15):897–921, 2007.

[14] C. A. Reed and G. W. A. Rowe. Araucaria: Software for argument analysis, diagramming and representation. *Int. Journal of AI Tools*, 14(3-4), 2004.