Understanding the Abstract Dialectical Framework

Sylwia Polberg*

University College London,
Gower Street 66–72, London WC1E 6EA, United Kingdom

Abstract. Among the most general structures extending the framework by Dung are the abstract dialectical frameworks (ADFs). They come equipped with various types of semantics, with the most prominent – the labeling–based one – being analyzed in the context of computational complexity, instantiations and software support. This makes the abstract dialectical frameworks valuable tools for argumentation. However, there are fewer results available concerning the relation between the ADFs and other argumentation frameworks. In this paper we would like to address this issue by introducing a number of translations from various formalisms into ADFs. The results of our study show the similarities and differences between them, thus promoting the use and understanding of ADFs. Moreover, our analysis also proves their capability to model many of the existing frameworks, including those that go beyond the attack relation. Finally, translations allow other structures to benefit from the research on ADFs in general and from the existing software in particular.

1 Introduction

Argumentation has become an influential subfield of AI [1]. Within this domain, we distinguish the abstract argumentation, at the heart of which lies Dung's framework (AF) [2]. A number of its generalizations has been proposed [3], including the abstract dialectical framework (ADF)[4]. ADFs come equipped with various types of semantics [5–8], the most prominent of which – the labeling–based one – is analyzed in the context of computational complexity [9], instantiations [10] and software support [11]. This makes ADFs valuable tools for argumentation. Unfortunately, their unusual structure can be a deterrent against their more widespread use. Moreover, at the first glance it is also difficult to say what is the relation between the ADFs and the other argumentation frameworks, in particular those that can express support [12–14].

In this paper we would like to tackle these issues by introducing a number of translations from various formalisms into the ADFs. This includes the Dung's framework [2], the Nielsen's and Parson's framework with joint attacks [15], the extended argumentation framework [16] and the argumentation framework with necessities [13]. The results of our study show the similarities and differences between ADFs and other argumentation formalisms, thus promoting the use and understanding of ADFs. Moreover,

* The author is a member of the Vienna PhD School of Informatics. This research was funded by project I1102 supported by the Austrian Science Fund FWF. The author is currently supported by EPSRC Project EP/N008294/1 "Framework for Computational Persuasion"

our analysis also proves their capability to model many of the existing frameworks, including those that go beyond the attack relation. Furthermore, a wider range of extended argumentation frameworks can be translated into ADFs than into AFs [17].

This paper is structured as follows. In Sec. 2 and 3 we recall the aforementioned argumentation frameworks. We also provide a discussion on certain design differences between the ADFs and the other structures. In Sec. 5 we present our translations. We close the paper with final remarks and comments on shifting other frameworks to ADFs.

2 Argumentation Frameworks

In this section we will recall the relevant argumentation frameworks and their extension–based semantics. Despite the various structural differences between the frameworks, their semantics tend to follow the design patterns established by Dung [2]. We can obtain most of them by combining conflict–freeness, acceptability and various ways to maximize or minimize the extensions. Thus, many frameworks tend to redefine these "building blocks", and then reuse the original (or similar) definitions from [2]. Therefore, when recalling the relevant structures in this section, we will mostly provide only the necessary notions. Throughout this work, we will be focusing on finite structures.

2.1 Dung's Argumentation Framework

Let us start with the famous Dung's framework [2], which is based on binary attack.

Definition 1. *A **Dung's abstract argumentation framework** (AF) is a pair $F = (A, R)$, where A is a set of **arguments** and $R \subseteq A \times A$ is the **attack** relation.*

Definition 2. *Let $F = (A, R)$ be a Dung's framework and $X \subseteq A$ a set of arguments.*

- *the **attacker set** of X is $X^- = \{a \mid \exists b \in X, aRb\}$*
- *the **discarded set** of X is $X^+ = \{a \mid \exists b \in X, bRa\}$.*
- *X **defends**[1] an argument $a \in A$ iff every argument $b \in A$ that attacks a is in X^+.*
- *X is **conflict–free** in F iff there are no $a, b \in X$ s.t. a attacks b.*

Definition 3. *Let $F = (A, R)$ be an AF. A set $X \subseteq A$ is:*

- ***admissible** in F iff it is conflict–free in F and defends in F all of its members.*
- ***preferred** in F iff it is maximal w.r.t. $\subseteq$ admissible in F.*
- ***complete** in F iff it is admissible and every $a \in A$ that is defended by X, is in X.*
- ***grounded** in F iff it is the least fixed point of the characteristic operator $\mathcal{F}_F : 2^A \to 2^A$ defined as $\mathcal{F}_F(X) = \{a \mid a \text{ is defended by } X \text{ in } F\}$*
- ***stable** in F iff it is conflict–free in F and $A \setminus X = X^+$.*

Different types of semantics can be related to each other in a number of ways [2], however, it is usually the following properties that will hold:

Theorem 1. *Let $F = (A, R)$ be an AF. The following holds:*

1. *Every stable extension of F is also preferred, but not vice versa.*
2. *Every preferred extension of F is also complete, but not vice versa.*
3. *The grounded extension of F is the least w.r.t. $\subseteq$ complete extension of F.*

[1] Defense is often substituted with acceptability, i.e. X defends a iff a is acceptable w.r.t. X.

2.2 Framework with Sets of Attacking Arguments

In some cases, a single argument might not be enough to carry out an attack on another argument. For example, all of the means, motive, opportunity and evidence might be required to prove guilt. In order to grasp such problems, a framework with group conflict was developed [15]. The semantics of SETAFs are almost identical to the AF ones. Given a set $X \subseteq A$, the attacks will now be carried out not by single arguments in X, but its subsets. Thus, in the interest of space, we will not formally give their definitions.

Definition 4. *A **framework with sets of attacking arguments** (SETAF) is a pair $SF = (A, R)$, where A is the set of **arguments** and $R \subseteq (2^A \setminus \emptyset) \times A$ is the **attack** relation.*

Example 1. Let us consider the SETAF $SF = (A, R)$, where $A = \{a, b, c, d, e\}$ and $R = \{(\{a\}, c), (\{b\}, a), (\{b\}, b), (\{c\}, d), (\{e\}, a), (\{b, d\}, e)\}$. The only admissible extensions are $\emptyset$ and $\{c, e\}$; both of them are complete. $\{c, e\}$ is the preferred extension, while $\emptyset$ is grounded. Because of b, this particular framework has no stable extensions.

2.3 Extended Argumentation Framework with Collective Attacks

The extended argumentation framework with collective defense attacks [16] is an improvement of the framework studied in [17, 18]. It introduces the notion of defense attacks, which occur between sets of arguments and binary conflicts. They can "override" a given attack due to e.g. the target's importance, which is a common approach in the preference–based argumentation [**?**,19, 20]. The added value of defense attacks is the fact that the arguments carrying them out can also be attacked and questioned.

Definition 5. *An **extended argumentation framework with collective defense attacks (EAFC)** is a tuple $EFC = (A, R, D)$, where A is a set of **arguments**, $R \subseteq A \times A$ is a set of **attacks** and $D \subseteq (2^A \setminus \emptyset) \times R)$ is the set of **collective defense attacks**.*

We can observe that a given attack can be successful (referred to as a defeat) or not, depending on the presence of suitable defense attacks. The defense has to include not just defending the arguments, but also a form of "protection" of the important defeats:

Definition 6. *Let $EFC = (A, R, D)$ be an EAFC and $X \subseteq A$ a set of arguments.*

- *an argument a **defeats**$_X$ an argument b in EFC w.r.t. X iff $(a, b) \in R$ and there is no $C \subseteq X$ s.t. $(C, (a, b)) \in D$.*
- *a set of pairs $R_X = \{(x_1, y_1), ..., (x_n, y_n)\}$ s.t. x_i defeats$_X$ y_i in EFC and for $i = 1...n$, $x_i \in X$, is a **reinstatement set** on X for a defeat$_X$ by argument a on argument b iff $(a, b) \in R_X$ and for every pair $(x, y) \in R_X$ and set of arguments $C \subseteq A$ s.t. $(C, (x, y)) \in D$, there is a pair $(x', y') \in R_X$ for some $y' \in C$.*
- *the **discarded set** of X is $X^+ = \{a \mid \exists b \in X$ s.t. b defeats$_X$ a and there is a reinstatement set on X for this defeat$_X\}$.*
- *X **defends** and argument $a \in A$ in EFC iff every argument $b \in A$ s.t. b defeats$_X$ a in EFC is in X^+.*
- *$X \subseteq A$ is **conflict–free** in EFC iff there are no $a, b \in X$ s.t. a defeats$_X$ b in EFC.*

With the exception of the grounded semantics, all extensions are defined in the same way as in Def. 3. Unfortunately, despite these similarities, Thm. 1 cannot be entirely extended to EAFCs. Finally, within EAFCs we can distinguish the bounded hierarchical subclass, enforcing certain restrictions on the attacks and defense attacks.

Definition 7. *Let $EFC = (A, R, D)$ be a finitiary[2] EAFC, $X \subseteq A$ a set of arguments and 2^{CF} the set of all conflict–free sets of EFC. The **characteristic function** $\mathcal{F}_{EFC} : 2^{CF} \rightarrow 2^A$ of EFC is defined as $\mathcal{F}_{EFC}(X) = \{a \mid a \text{ is defended by } X \text{ in } EFC\}$. We define a sequence of subsets of A s.t. $\mathcal{F}^0_{EFC} = \emptyset$ and $\mathcal{F}^{i+1}_{EFC} = \mathcal{F}_{EFC}(\mathcal{F}^i_{EFC})$. The **grounded** extension of EFC is $\bigcup_{i=0}^{\infty}(\mathcal{F}^i_{EFC})$.*

Theorem 2. *Let $EFC = (A, R, D)$ be a finitary EAFC. The following holds:*

1. *Every preferred extension is complete, but not vice versa*
2. *Every stable extension is complete, but not vice versa*
3. *The grounded extension is a minimal w.r.t. $\subseteq$ complete extension*

Definition 8. *An EAFC $EFC = (A, R, D)$ is **bounded hierarchical** iff there exists a partition $\delta_H = (((A_1, R_1), D_1), ..., ((A_n, R_n), D_n))$ s.t. $D_n = \emptyset$, $A = \bigcup_{i=1}^{n} A_i$, $R = \bigcup_{i=1}^{n} R_i$, $D = \bigcup_{i=1}^{n} D_i$, for every $i = 1...n$ (A_i, R_i) is a Dung's framework, and $(c, (a, b)) \in D_i$ implies $(a, b) \in R_i$, $c \subseteq A_{i+1}$.*

Example 2. [21] Let $EFC = (\{a, b, c, d, e, f, g\}, \{(a, b), (d, c), (b, e), (e, f), (f, g)\}, \{(\{b\}, (d, c)), (\{c\}, (a, b))\})$ be an EAFC. Let us look at some of its conflict–free extensions. We can see that $\{a, b\}$ and $\{c, d\}$ are not conflict–free. However, both $\{a, b, c\}$ and $\{b, c, d\}$ are, due to the presence of defense attackers. Additionally, also $\{a, b, c, d\}$, $\{a, d, e, g\}$ and $\{b, c, a, d, f\}$ are conflict–free. The admissible extensions of EFC include $\emptyset$, $\{a\}$, $\{d\}$, $\{a, d\}$, $\{b, c\}$, $\{a, b, c\}$, $\{b, c, d\}$, $\{a, d, e\}$, $\{b, c, f\}$, $\{a, b, c, f\}$, $\{b, c, d, f\}$, $\{a, d, e, g\}$, $\{a, b, c, d\}$ and $\{a, b, c, d, f\}$. We can observe that the set $X = \{b, c\}$ is admissible. Neither a nor d defeat$_X$ any of its elements, and thus there is nothing to defend from. The set $\{a, d, e\}$ is admissible since the defeat of b by a has a reinstatement set $\{(d, c), (a, b)\}$. Although its behavior appears cyclic, it suffices for defense. The sets $\{a, d, e, g\}$ and $\{a, b, c, d, f\}$ are complete. We can observe they are incomparable and do not follow the typical semi–lattice structure of complete extensions. The grounded extension is $\{a, d, e, g\}$; it is minimal, but not the least complete extension. Both $\{a, d, e, g\}$ and $\{a, d, b, c, d, f\}$ are stable and preferred.

2.4 Argumentation Framework with Necessities

Various types of support have been studied in abstract argumentation [12–14]. Due to limited space, we will focus on the necessary support, though based on the research in [12, 14] our results can be extended to other relations as well. We say that a set of arguments X *necessarily supports* b if we need to assume at least one element of

[2] An EAFC is finitiary if for every argument and attack, the collection of its (defense) attackers is finite.

X in order to accept b. Using this relation has certain important implications. First of all, argument's supporters need to be present in an extension. Secondly, an argument can be now indirectly attacked by the means of its supporters, i.e. we can "discard" an argument not just by providing a direct conflict, but also by cutting off its support. Finally, a certain notion of a validity of an argument is introduced, stemming from its participation in support cycles. It affects the acceptance and attack capabilities of an argument. Let us now recall the framework with necessities [13]:

Definition 9. *An **abstract argumentation framework with necessities** (AFN) is a tuple $FN = (A, R, N)$ where A is a set of **arguments**, $R \subseteq A \times A$ represents the **attack** relation and $N \subseteq (2^A \setminus \emptyset) \times A$ represents the **necessity** relation.*

The acyclicity restrictions are defined through the powerful sequences and the related coherent sets. By joining conflict–freeness and coherence, we obtain a new semantics which replaces conflict–freeness as the basis of stable and admissible extensions. The remaining notions are defined similarly as in Def. 3 and satisfy Thm. 1.

Definition 10. *Let $FN = (A, R, N)$ be an AFN and $X \subseteq A$ a set of arguments. An argument $a \in A$ is **powerful** in X iff $a \in X$ and there is a sequence $a_0, ..., a_k$ of elements of X s.t. : i) $a_k = a$ ii) there is no $B \subseteq A$ s.t. BNa_0 iii) for $1 \leq i \leq k$: for each $B \subseteq A$ s.t. BNa_i, it holds that $B \cap \{a_0, ..., a_{i-1}\} \neq \emptyset$. A set of arguments $X \subseteq A$ is **coherent** in FN iff each $a \in X$ is powerful in X.*

Definition 11. *Let $FN = (A, R, N)$ be an AFN and $X \subseteq A$ a set of arguments.*

- *the **discarded** set of X in FN is defined as $X^{att} = \{a \mid$ for every coherent $C \subseteq A$ s.t. $a \in C$, $\exists c \in C, e \in X$ s.t. $eRc\}$[3].*
- *X **defends** an argument $a \in A$ in FN iff $X \cup \{a\}$ is coherent and for each $b \in A$, if bRa then $b \in X^{att}$.*
- *X is **conflict–free** in FN iff there are no $a, b \in X$ s.t. a attacks b.*

Definition 12. *Let $FN = (A, R, N)$ be an AFN. A set of arguments $X \subseteq A$ is:*

- ***strongly coherent** in FN iff it is conflict–free and coherent in FN*
- ***admissible** in FN iff it is strongly coherent and defends all of its arguments in FN.*
- ***stable** in FN iff it is strongly coherent in FN and $X^{att} = A \setminus X$.*

Example 3. Let $(\{a, b, c, d, e, f\}, \{(a, e), (d, b), (e, c), (f, d)\}, \{(\{b, c\}, a), (\{f\}, f)\})$ be an AFN. Its coherent sets include $\emptyset$, $\{a, b\}$, $\{a, c\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$ and any of their combinations. In total, we have six admissible extensions. $\emptyset$ is trivially admissible. So is $\{d\}$ due to the fact that f does not possess a powerful sequence in FN. However, $\{e\}$ is not admissible; it does not attack one of the coherent sets of a, namely $\{a, b\}$. Fortunately, $\{d, e\}$ is already admissible. We can observe that b can never be defended and will not appear in an admissible set. The last two admissible sets are $\{a, c\}$ and $\{a, c, d\}$. The extensions $\{d\}$, $\{d, e\}$ and $\{a, c, d\}$ are complete, with the first one being grounded and the latter two preferred. In this case, both $\{d, e\}$ and $\{a, c, d\}$ are stable.

[3] Please note that we do not denote the AFN discarded set with X^+ as in the previous cases in order not to confuse it with the notion of the deactivated set from [13], which is less restrictive

3 Abstract Dialectical Frameworks

Abstract dialectical frameworks have been defined in [4] and further studied in [5–10]. Their main goal was to be able to express a wide range relations and try to avoid the need of introducing a new relation set each time it is needed. This is achieved by the means of acceptance conditions, which define when an argument can be accepted or rejected. They can be defined either as total functions over the parents of an argument [4] or as propositional formulas over them.

Definition 13. *An **abstract dialectical framework** (ADF) is a tuple $DF = (A, L, C)$, where A is a set of **arguments**, $L \subseteq A \times A$ is a set of **links** and $C = \{C_a\}_{a \in A}$ is a set of **acceptance conditions**, one condition per each argument. An acceptance condition is a total function $C_a : 2^{par(a)} \to \{in, out\}$, where $par(a) = \{p \in A \mid (p, a) \in L\}$ is the set of **parents** of an argument a.*

Due to the fact that the set of links can be inferred from the conditions, we will write simply (A, C) to denote an ADF. The basic "building blocks" of the extension–based ADF semantics from [7, 8] are the decisively in interpretations and various types of evaluations we derive from them. A two–valued interpretation is simply a mapping that assigns truth values $\{\mathbf{t}, \mathbf{f}\}$ to (a subset of) arguments. For an interpretation v, v^x is the set of elements mapped to $x \in \{\mathbf{t}, \mathbf{f}\}$ by v. A decisive interpretation v for an argument $a \in A$ represents an assignment for a set of arguments $X \subseteq A$ s.t. independently of the status of the arguments in $A \setminus X$, the outcome of the condition of a stays the same.

Definition 14. *Let A be a collection of elements, $X \subseteq A$ its subset and v a two–valued interpretation defined on X. A **completion** of v to a set Z where $X \subseteq Z \subseteq A$, is an interpretation v' defined on Z in a way that $\forall a \in X \; v(a) = v'(a)$. v' is a $\mathbf{t}/\mathbf{f}$ completion of v iff all arguments in $Z \setminus X$ are mapped respectively to $\mathbf{t}/\mathbf{f}$.*

Definition 15. *Let $DF = (A, L, C)$ be an ADF, $X \subseteq A$ a set of arguments and v a two–valued interpretation defined on X. v is **decisive** for an argument $s \in A$ iff for any two completions $v_{par(s)}$ and $v'_{par(s)}$ of v to $X \cup par(s)$, it holds that $v_{par(s)}(C_s) = v'_{par(s)}(C_s)$. s is **decisively out/in** w.r.t. v if v is decisive and all of its completions evaluate C_s to respectively out, in.*

From now on we will focus on the minimal interpretations, i.e. those in which both $v^{\mathbf{t}}$ and $v^{\mathbf{f}}$ are minimal w.r.t. $\subseteq$. By $min\_dec(x, s)$ we denote the set of minimal two–valued interpretations that are decisively x for s, where s is an argument and $x \in \{in, out\}$. From the positive parts of a decisively in interpretation for a we can extract arguments required for the acceptance of a. With this information, we can define various types of evaluations, not unlike the powerful sequences in AFNs. However, due to the fact that ADFs are more expressive than AFNs, it is also the $\mathbf{f}$ parts of the used interpretations that need to be stored [7, 8]:

Definition 16. *Let $DF = (A, L, C)$ be an ADF and $X \subseteq A$ a set of arguments. A **positive dependency function** (pd–function) on X is a function pd^{DF}_X assigning every argument $a \in X$ an interpretation $v \in min\_dec(in, a)$ s.t. $v^t \subseteq X$, or $\mathcal{N}$ for null iff no*

*such v can be found. pd_X^{DF} is **sound** on X iff for no $a \in X$, $pd_X^{DF}(a) = \mathcal{N}$. pd_X^{DF} is **maximally sound** on X iff it is sound on $X' \subseteq X$ and there is no other sound function ${pd'}_X^{DF}$ on X'' s.t. $\forall a \in X'$, $pd_X^{DF}(a) = {pd'}_X^{DF}(a)$, where $X' \subset X'' \subseteq X$.*

Definition 17. *Let $DF = (A, L, C)$ be an ADF, $S \subseteq A$ and pd_X^{DF} a maximally sound pd–function of S defined over $X \subseteq S$. A **partially acyclic positive dependency evaluation** based on pd_X^{DF} for an argument $x \in X$ is a triple $(F, (a_0, ..., a_n), B)$, where $F \cap \{a_0, ..., a_n\} = \emptyset$, $(a_0, ..., a_n)$ is a sequence of distinct elements of X satisfying the following requirements:*

- *if the sequence is non–empty, then $a_n = x$; otherwise, $x \in F$*
- *$\forall_{i=1}^{n}$, $pd_X^{DF}(a_i)^{\mathbf{t}} \subseteq F \cup \{a_0, ..., a_{i-1}\}$, $pd_X^{DF}(a_0)^{\mathbf{t}} \subseteq F$*
- *$\forall a \in F$, $pd_X^{DF}(a)^{\mathbf{t}} \subseteq F$*
- *$\forall a \in F, \exists b \in F$ s.t. $a \in pd_X^{DF}(b)$.*

*Finally, $B = \bigcup_{a \in F} pd_X^{DF}(a)^{\mathbf{f}} \cup \bigcup_{i=0}^{n} pd_X^{DF}(a_i)^{\mathbf{f}}$. We refer to F as the **pd–set**, to $(a_0, ..., a_n)$ as the **pd–sequence** and to B as the **blocking set** of the evaluation. A partially acyclic evaluation $(F, (a_0, ..., a_n), B)$ for an argument $x \in X$ is an **acyclic positive dependency evaluation** for x iff $F = \emptyset$.*

We will use the shortened notation $((a_0, ..., a_n), B)$ for the acyclic evaluations. There are two ways we can "attack" an evaluation. Either we accept an argument that needs to be rejected (i.e. it is in the blocking set), or we are able to discard one that needs to be accepted (i.e. is in the the pd–sequence or the pd–set). We will be mostly concerned with the first type. We can now define various discarded sets in ADFs[4]:

Definition 18. *Let $DF = (A, L, C)$ be an ADF and $X \subseteq A$ a set of arguments. The **standard discarded** set of X is $X^{+} = \{a \in A \mid$ for every partially acyclic evaluation (F, G, B) for a, $B \cap X \neq \emptyset\}$. The **partially acyclic discarded** set of X is $X^{p+} = \{a \in A \mid$ there is no partially acyclic evaluation (F', G', B') for a s.t. $F' \subseteq X$ and $B' \cap X = \emptyset\}$. The **acyclic discarded** set of X is $X^{a+} = \{a \in A \mid$ for every pd–acyclic evaluation (F, B) for a, $B \cap X \neq \emptyset\}$.*

Given a set of arguments X and its discarded set S, we can build a special interpretation – called **range** – with which we can check for decisiveness. The range can be constructed by assigning $\mathbf{t}$ to arguments in X and $\mathbf{f}$ to those in $S \setminus X$. Under certain conditions X and S are disjoint, which brings us to the conflict–free semantics:

Definition 19. *Let $DF = (A, L, C)$ be an ADF. A set $X \subseteq A$ is a **conflict–free extension** of DF if for all $s \in X$ we have $C_s(X \cap par(s)) = in$. X is a **pd–acyclic conflict–free extension** of DF iff every $a \in X$ has an acyclic evaluation (F, B) on X s.t. $B \cap X = \emptyset$.*

Lemma 1. *Let $DF = (A, L, C)$ be an ADF and $X \subseteq A$ a set of arguments. If X is conflict–free in DF, then $X \cap X^{+} = \emptyset$ and $X \cap X^{p+} = \emptyset$. Moreover, it holds that $X^{+} \subseteq X^{p+} \subseteq X^{a+}$. If X is pd–acyclic conflict–free, then $X \cap X^{a+} = \emptyset$ and $X^{p+} = X^{a+}$.*

[4] The presented definitions are generalizations of the ones from [7, 8].

By combining a given type of a discarded set and a given type of conflict–freeness, we have developed various families of extension–based semantics [7, 8]. We have classified them into the four main types and used an $xy-$ prefixing system to denote them. In the context of this work, three of the families will be relevant. We will now recall their definitions and refer the readers to [8] for proofs and further explanations.

Definition 20. *Let $DF = (A, L, C)$ be an ADF. Let $X \subseteq A$ be a set of arguments and v_X, v_X^a and v_X^p its standard, acyclic and partially acyclic ranges.*

*If X is conflict–free and every $e \in X$ is decisively in w.r.t. v_X (v_X^p), then X is **cc–admissible (ca$_2$–admissible)** in DF. If X is pd–acyclic conflict–free and every $e \in X$ is decisively in w.r.t. v_X^a, then X is **aa–admissible** in DF.*

*If X is cc–admissible (ca$_2$–admissible, aa–admissible) and every argument $e \in A$ decisively in w.r.t. v_X (v_X^p, v_X^a) is in X, then X is **cc–complete (ca$_2$–complete, aa–complete)** in DF. If $X$is maximal w.r.t. set inclusion xy–admissible extension, where $x, y \in \{a, c\}$, then it is an **xy–preferred** extensions of DF.*

*If X is conflict–free and for every $a \in A \setminus X$, $C_a(X \cap par(a)) = out$, then X is a **model** of DF. If X is pd–acyclic conflict–free and $X^{a+} = A \setminus X$, then X is a **stable** extension of DF .*

*If X is the least w.r.t. $\subseteq$ cc–complete extension, then it is the **grounded** extension of DF. If X is the least w.r.t. $\subseteq$ aa–complete extension, then it is the **acyclic grounded** extension of DF.*

Finally, we can define the two important ADF subclasses. The bipolar ADFs consist only of links that are supporting or attacking. This class is particularly valuable due to its computational complexity properties [9]. The other subclass, referred to as AADF$^+$, consists of ADFs in which our semantics classification collapses. By this we understand that e.g. every cc–complete extension is aa–complete and vice versa. Moreover, this class provides a more precise correspondence between the extension and labeling–based semantics for ADFs [8]. This means that for these frameworks, we can use the DIAMOND software [11] and other results for the labeling–based semantics [9, 10].

Definition 21. *Let $DF = (A, L, C)$ be an ADF. A link $(r, s) \in L$ is: i) supporting iff for no $R \subseteq par(s)$ we have that $C_s(R) = in$ and $C_s(R \cup \{r\}) = out$ ii) attacking iff for no $R \subseteq par(s)$ we have that $C_s(R) = out$ and $C_s(R \cup \{r\}) = in$. DF is a **bipolar** ADF (BADF) iff it contains only links that are supporting or attacking. DF is a **positive dependency acyclic** ADF (AADF$^+$) iff every partially acyclic evaluation (F, G, B) of DF is acyclic.*

Theorem 3. *Let $DF = (A, L, C)$ be an AADF$^+$. The following holds:*

- *Every conflict–free extension of DF is pd–acyclic conflict–free in DF*
- *Every model of DF is stable in DF*
- *The aa/cc/ca$_2$–admissible extensions of DF coincide*
- *The aa/cc/ca$_2$–complete extensions of DF coincide*
- *The aa/cc/ca$_2$–preferred extensions of DF coincide*
- *The grounded and acyclic grounded extensions of DF coincide*

Example 4. Let us consider the framework is $DF = (\{a, b, c, d, e, f, g\}, \{C_a : \top, C_b : \neg a \vee c, C_c : \neg d \vee b, C_d : \top, C_e : \neg b, C_f : \neg e, C_g : \neg f\})$. We can observe that both a and d have trivial acyclic evaluations $((a), \emptyset)$ and $((d), \emptyset)$. For e, f and g we can construct $((e), \{b\})$, $((f), \{e\})$ and $((g), \{f\})$. The situation only gets complicated with b and c; we have the acyclic evaluations $((b), \{a\})$, $((c, b), \{d\})$, $((c), \{d\})$, $((b, c), \{a\})$ and the partially acyclic one $(\{b, c\}, \emptyset)$. We can observe that $\emptyset$ is an admissible extension of any type; all of its discarded sets are empty. Decisively in w.r.t. its ranges are thus a and d. The set $\{a, d\}$ is again admissible. Its standard discarded set is $\emptyset$, however, the acyclic and partially acyclic ones are $\{b, c\}$. Therefore, $\{a, d\}$ is only cc–complete. Discarding b leads to the acceptance of e and g. Hence, $\{a, d, e, g\}$ is an aa– and ca_2–complete extension, though it does not even qualify as a cc–admissible set. We can now consider the set $\{a, b, c, d\}$. It is conflict–free, but not pd–acyclic conflict–free. Its standard and partially acyclic discarded set is $\{e\}$, which means that f can be accepted. Hence, $\{a, b, c, d, f\}$ is cc and ca_2–complete. Thus, in total we obtain two cc–complete, one aa–complete and two ca_2–complete sets. Our grounded and acyclic grounded extensions are $\{a, d\}$ and $\{a, d, e, g\}$ respectively. The latter set is also the only stable extension of our framework. However, both $\{a, d, e, g\}$ and $\{a, b, c, d, f\}$ are models.

4 Conceptual Differences Between ADFs and Other Frameworks

The more direct descendants of the Dung's framework explicitly state "this is a supporter", "this is an attacker" and so on. Thus, in order to know if a given argument can be accepted along with the other arguments, i.e. whether it is attacked, defeated or receives sufficient support, we need to go through all the relations it is a target of. In contrast, the acceptance conditions "zoom out" from singular relations. They tell us whether the argument can be accepted or not w.r.t. a given set of arguments in a straightforward manner. The focus is put on what would usually be seen as a target of a relation, while in other frameworks the attention is on the relation source. As a consequence, in order to say if a parent of an argument is its supporter, attacker or none of these, we need analyze the condition further, as seen in e.g. Def. 21. This is also one of the reasons why finding support cycles in ADFs is more difficult than in other support frameworks. Finally, since the role of parent is derived from how it affects the behavior of an argument, not whether it is in e.g. the support relation N, an attacker or a supporter in a given framework may not have the same role in the corresponding ADF:

Example 5. Let $(\{a, b, c\}, \{(b, a), (a, c)\}, \{(\{b\}, a)\})$ be an AFN, where the argument a is at the same time supported and attacked by b. In a certain sense, the (a, b) relation is difficult to classify as positive or negative. Although a cannot be accepted, it is still a valid attacker that one needs to defend from. In the ADF setting, the acceptance condition of a is unsatisfiable – whether we include or exclude b, we always reject a. It can also be seen as a $b \wedge \neg b$ formula. a does not possess any type of an evaluation and will always end up in any type of a discarded set. This also means that we do not have to "defend" from it. In this particular example, the set $\{c\}$ would not be considered admissible in our AFN, but it would be considered an admissible extension of any type in the ADF $(\{a, b, c\}, \{C_a = b \wedge \neg b, C_b = \top, C_c = \neg a\})$.

Thus, there is an important difference between the design of ADFs and other argumentation frameworks. If we were to represent the situation as a propositional formula, it is like comparing an atom based and a literal based evaluation. The same issue arises when we consider standard and ultimate versions of logic programming semantics, as already noted in [5]. This means that if we want to translate e.g. an AFN into an ADF while still preserving the behavior of the semantics, we need to make sure that no argument is at the same time an attacker and a supporter of the same argument. A similar issue also appears in the extended argumentation frameworks. The defense attack is a type of a positive, indirect relation towards the "defended" argument. The difference is that while in the first case it is also a negative relation towards the argument carrying out the attack, in the latter the attacker and the defense attacker might be unrelated. It is not unlike what is informally referred to as the "overpowering support" in ADFs. A typical example is a condition of the form $C_a = \neg b \vee c$, where b has the power to out the condition unless c is present. Therefore, defense attackers from EAFC become directly related to the arguments they "protect" in ADFs, which can lead to inconsistencies.

Definition 22. *Let $FN = (A, R, N)$ be an AFN and a an argument in A. By $N(a) = \{b \mid \exists B \subseteq A \text{ s.t. } b \in B, BNa\}$ and $R(a) = \{b \mid bRa\}$ we denote the sets of arguments supporting and attacking a. Then a is **strongly consistent** iff $N(a) \cap R(a) = \emptyset$. FN is strongly consistent iff all of its arguments are strongly consistent.*

*Let $EFC = (A, R, D)$ be an EAFC. EFC is **strongly consistent** iff there are no $x, y, z \in A$ and $X \subseteq A$ s.t. $(x, y) \in R$, $x \in X$ and $(X, (z, y)) \in D$.*

Any AFN can be made strongly consistent with the help of no more than $|A|$ arguments. We basically introduce extra arguments, that we call "bypasses", that take over the support links leading to inconsistency and connect them to the original sources of these relations. For example, the AFN $(\{a, b\}, \{(a, b)\}, \{(\{a\}, b)\})$ is extended to $(\{a, a', b\}, \{(a, b)\}, \{(\{a\}, a'), (\{a'\}, b)\})$. The auxiliary arguments then need to be removed from the extensions. We can also turn them into self–attackers, which addresses the removal issue, but it also affects the stable semantics. Similar techniques can be used in the translations for EAFCs. Unfortunately, due to the space restrictions, we cannot focus on this approach here.

Please note that this analysis does not in any way imply that a given (a, b) link is assigned a single permanent "role" in ADFs, such as "attack" or "support". The framework is flexible and a link can be positive on one occasion an negative on another. A more accurate description is that a link (or its source) should have a defined role "at a point", i.e. w.r.t. a given set of arguments. ADFs ensure consistency, not constancy.

5 Translations

In this section we will show how to translate the recalled frameworks to ADFs. We will provide both functional and propositional acceptance conditions. For the latter, we would like to introduce the following notations. For a set of arguments $X = \{x_1, ..., x_n\}$, we will abbreviate the formula $x_1 \wedge ... \wedge x_n$ with $\bigwedge X$ and $\neg x_1 \wedge ... \wedge \neg x_n$ with $\bigwedge \neg X$. Similarly, $x_1 \vee ... \vee x_n$ and $\neg x_1 \vee ... \vee \neg x_n$ will be shortened to $\bigvee X$ and $\bigvee \neg X$.

5.1 Translating SETAFs and AFs into ADFs

A straightforward translation from AFs to ADFs has already been introduced in [6]. Let $a \in A$ be an argument and $\{a\}^- = \{x_1, .., x_n\}$ its attacker set in an AF. Whenever any of $x_i's$ is present, a cannot be accepted. Only when all of them are absent, we can assume a. The SETAF translation is quite similar. Let $\{a\}^- = \{X_1, ..., X_n\}$ be the collection of all sets that attack an argument a, i.e. sets s.t. $X_i Ra$. Only the presence of all members of any X_i, not just some of them, renders a unacceptable. Therefore given any set of arguments that does not fully include at least one attacking set, the acceptance condition of a is in. This brings us to the following two translations:

Translation 1. *Let $F = (A, R)$ be a Dung's framework. The ADF corresponding to F is $DF^F = (A, R, C)$, where $C = \{C_a\}_{a \in A}$ and every C_a is as follows:*

- *Functional form: $C_a(\emptyset) = in$ and for all nonempty $B \subseteq \{a\}^-$, $C_a(B) = out$.*
- *Propositional form: $C_a = \bigwedge \neg\{a\}^-$. In case $\{a\}^-$ is empty, $C_a = \top$.*

Translation 2. *Let $SF = (A, R)$ be a SETAF. The ADF corresponding to SF is $DF^{SF} = (A, L, C)$, where $L = \{(x, y) \mid \exists B \subseteq A, x \in B$ s.t. $BRy\}$, $C = \{C_a\}_{a \in A}$ and every C_a is created in the following way:*

- *Functional form: for every $B \subseteq \bigcup\{a\}^-$, if $\exists X_i \in \{a\}^-$ s.t. $X_i \subseteq B$, then $C_a(B) = out$; otherwise, $C_a(B) = in$.*
- *Propositional form: $C_a = \bigvee \neg X_1 \wedge ... \wedge \bigvee \neg X_n$. If $\{a\}^-$ is empty, $C_a = \top$.*

Neither AFs nor SETAFs rely on any form of support. Therefore, their associated ADFs are both AADF$^+$s and BADFs. Consequently, our semantics classification collapses and it does not matter which type of ADF semantics we work with.

Theorem 4. *Let $SF = (A, R)$ be a SETAF or AF and $DF^{SF} = (A, L, C)$ its corresponding ADF. Then DF^{SF} is an AADF$^+$ and a BADF.*

Theorem 5. *Let $SF = (A, R)$ be a SETAF or AF and $DF^{SF} = (A, L, C)$ its corresponding ADF. A set of arguments $X \subseteq A$ is a conflict–free extensions of SF iff it is (pd–acyclic) conflict–free in DF^F. $X \subseteq A$ is a stable extensions of SF iff it is (stable) model of DF^F. $X \subseteq A$ is a grounded extensions of SF iff it is (acyclic) grounded in DF^F. $X \subseteq A$ is a σ–extensions of SF, where where $\sigma \in \{$admissible, preferred, complete$\}$ iff it is an xy–σ–extension of DF^F for $x, y \in \{a, c\}$.*

Example 6. Let us continue Example 1. The ADF associated with SF is $DF^{SF} = (\{a, b, c, d, e\}, \{C_a : \neg a \wedge \neg e, C_b : \neg b, C_c : \neg a, C_d : \neg c, C_e : \neg b \vee \neg d\})$. $\emptyset$ is an admissible extension of any type; its discarded set is also empty. We can observe that $\{c, e\}$ is conflict–free in DF^{SF}. Its discarded set is $\{a, d\}$, thus making the set admissible in DF^{SF}. No other argument is decisively in w.r.t. the produced ranges and thus both sets are also complete. This makes $\emptyset$ the grounded and $\{c, e\}$ the preferred extension. Since b is not contained in any discarded set, DF^{SF} has no stable or model extensions.

5.2 Translating EAFCs into ADFs

We can now focus on translating EAFCs into ADFs. Let us assume we have an attack (b, a) that is defense attacked by sets $\{c, d\}$ and $\{e\}$. We can observe that a is rejected only if b is present and none of the defense attacking sets is fully present. On the other hand, if b is not there or either $\{c, d\}$ or $\{e\}$ are accepted, then the requirements for a are satisfied. Therefore, for a given EAFC, we can create an ADF in the following way:

Translation 3. *Let $EFC = (A, R, D)$ be a strongly consistent EAFC. Its corresponding ADF is $DF^{EFC} = (A, L, C)$, where $L = \{(a, b) \mid aRb$ or $\exists c \in A, X \subseteq A$ s.t. $a \in X, (X, (c, b)) \in D\}$, $C = \{C_a \mid a \in A\}$ and every C_a is as follows:*

- *Functional form: for every set $B \subseteq par(a)$, if $\exists x \in B$ s.t. $(x, a) \in R$ and $\nexists B' \subseteq B$ s.t. $(B', (x, a)) \in D$, then $C_a(B) = out$; otherwise, $C_a(B) = in$*
- *Propositional form: if $\{a\}^- = \emptyset$, then $C_a = \top$; otherwise, $C_a = \bigwedge_{b \in A, (b,a) \in R} att_a^b$, where $att_a^b = \neg b \vee (\bigwedge B_1 \vee ... \bigwedge B_m)$ and $D_{b,a} = \{B_1, ..., B_m\}$ is the collection of all sets $B_i \subseteq A$ s.t. $(B, (b, a)) \in D$. If $D_{b,a}$ is empty, then $att_a^b = \neg b$.*

Although EAFCs are more advanced than e.g. AFs, their associated ADFs are still bipolar. However, only in the case of bounded hierarchical EAFCs they are also AADF$^+$s. The EAFC semantics are now connected to the ca$_2$–semantics family. Since the ADF associated with the framework from Example 2 is precisely the one we have considered in Example 4; we refer the reader there for further details.

Theorem 6. *Let $EFC = (A, R, D)$ be a strongly consistent EAFC and $DF^{EFC} = (A, L, C)$ its corresponding ADF. DF^{EFC} is a BADF. If EFC is bounded hierarchical, then DF^{EFC} is an AADF$^+$.*

Theorem 7. *Let EFC be a strongly consistent EAFC and $DF^{EFC} = (A, L, C)$ its corresponding ADF. A set of arguments $X \subseteq A$ is a conflict–free extension of EFC iff it is conflict–free in DF^{EFC}. X is a stable extension of EFC iff it is a model of DF^{EFC}. X is a grounded extension of EFC iff it is the acyclic grounded extension of DF^{EFC}. Finally, X is a σ–extension of EFC, where $\sigma \in \{$admissible, complete, preferred$\}$, iff it is a ca$_2$–σ–extension of DF^{EFC}.*

5.3 Translating AFNs into ADFs

In order to accept an AFN argument, two conditions need to be met. First of all, just like in AFs, the attackers of a given argument need to be absent. However, in addition, at least one member of every supporting set needs to be present. This gives us a description of an acceptance condition; the acyclicity will be handled by the appropriate semantics.

Translation 4. *Let $FN = (A, R, N)$ be a strongly consistent AFN. The corresponding ADF is $DF^{FN} = (A, L, C)$, where $L = \{(x, y) \mid (x, y) \in R$ or $\exists B \subseteq A, x \in B$ s.t. $BNy\}$, $C = \{C_a \mid a \in A\}$ and every C_a is as follows:*

- *Functional form: for every $P' \subseteq par(a)$, if $\exists p \in P'$ s.t. pRa or $\exists Z \subseteq A$ s.t. ZNa and $Z \cap P' = \emptyset$, then $C_a(P') = out$; otherwise, $C_a(P') = in$.*

– *Propositional form:* $C_a = att_a \cap sup_a$*, where:*
  - $att_a = \bigwedge \neg\{a\}^-$ *or* $att_a = \top$ *if* $\{a\}^- = \emptyset$
  - $sup_a = (\bigvee Z_1 \wedge ... \wedge \bigvee Z_m)$*, where* $Z_1, ..., Z_m$ *are all subsets of* A *s.t.* $Z_i N a$*, or* $sup_a = \top$ *if no such set exists*

The produced ADFs are still bipolar. However, whether a given ADF is an AADF$^+$ or not, depends on the support relation in the source AFN.

Theorem 8. *Let* $FN = (A, R, N)$ *be a strongly consistent AFN and* $DF^{FN} = (A, L, C)$ *its corresponding ADF. Then* DF^{FN} *is a BADF.*

The AFN semantics are built around the notion of coherence, which requires all relevant arguments to be (support–wise) derived in an acyclic manner. Thus, not surprisingly, it is the aa–family of ADF semantics that will be associated with the AFN semantics. In particular, we can relate powerful sequences to the acyclic evaluations. This also allows us to draw the connection between the acyclic discarded set in ADFs and the discarded set X^{att} in AFNs. Hence, there is a correspondence between the defense in AFNs and being decisively in w.r.t. a given interpretation in ADFs. This in turns tells us the relation between the extensions of AFNs and ADFs:

Lemma 2. *Let* $FN = (A, R, N)$ *be a strongly consistent AFN and* $DF^{FN} = (A, L, C)$ *its corresponding ADF. For a given powerful sequence for an argument* $a \in A$ *we can construct an associated pd–acyclic evaluation and vice versa.*

Theorem 9. *Let* $FN = (A, R, N)$ *be a strongly consistent AFN,* $DF^{FN} = (A, L, C)$ *its corresponding ADF.* X *is strongly coherent in* FN *iff it is pd–acyclic conflict–free in* DF^{FN}*.* X *is a* σ*–extension of* FN*, where* $\sigma \in \{$*admissible, complete, preferred*$\}$ *iff it is an aa–*σ*–extension of* DF^{FN}*.* X *is stable in* FN *iff it is stable in* DF^{FN}*.* X *is grounded in* FN *iff it is acyclic grounded in* DF^{FN}*.*

Example 7. Let us continue Example 3. The ADF associated with our AFN is ($\{a$, b, c, d, e, $f\}$, $\{C_a : b \vee c$, $C_b : \neg d$, $C_c : \neg e$, $C_d : \neg f$, $C_e : \neg a, C_f : f\}$). $\emptyset$ is trivially aa–admissible. Its acyclic discarded set is $\{f\}$, thus making d decisively in. Hence, $\emptyset$ is not aa–complete. The set $\{d\}$ discards f and b. This is not enough to accept any other argument. Hence, it is both aa–admissible and aa–complete. The set $\{e\}$ is pd–acyclic conflict–free, but not aa–admissible (it discards f and c). However, $\{d, e\}$ is aa–admissible (discarded set is $\{a, b, f, c\}$) and aa–complete. We can also show that $\{a, c, d\}$ is aa–admissible and aa–complete (discarded set is $\{b, f, e\}$). Therefore, $\{d\}$ is the acyclic grounded extension, while $\{d, e\}$ and $\{a, c, d\}$ are aa–preferred and stable.

6 Conclusions and Final Remarks

In this paper we have presented a number of translations from different argumentation frameworks to ADFs. We could have observed that for every structure, we have found a family of the extension–based ADF semantics which followed similar principles and thus were able to retrieve exactly the extensions of the framework we were translating.

We have also identified to which ADF subclass a given translation–produced framework belongs, so that the results from [9–11] can be exploited. Our results also show the differences between ADFs and other formalisms; in particular, we had to introduce consistency constraints in order to perform a translation. Nevertheless, this shortcoming can be addressed by the introduction of linearly many new arguments that take over the support relation. Unfortunately, due to the space constraints we did not describe the bypass method in detail. For the same reasons, we could not have presented certain translations. In particular, we have omitted the approach for evidential argumentation systems [22, 14]. However, based on the SETAF and AFN methods and the results from [14], this approach can be easily extrapolated. We hope we will manage to present these results in the extended version of this work.

In establishing the connections between the semantics of argumentation frameworks and ADFs, we have focused on the extension–based family. Nevertheless, the labeling–based approach is also a prominent one, and at least in the case of ADFs, better studied. However, as analyzed in [8], the usual relation between extensions and labelings that is found e.g. in the Dung's framework, does not hold for the dialectical framework. Due to the specialized nature the semantics we have presented here, all of of the available approaches can sometimes produce different results when faced with support cycles. This means that the labeling–based method can give us complete, preferred or grounded interpretations that do not necessarily correspond to the complete, preferred or grounded extensions of arbitrary AFNs and EAFCs. This can be addressed by limiting ourselves to those frameworks that are associated with AADF^{+}s. Therefore, although the approaches for AFs and SETAFs can be used without any modifications, we would have to distinguish a support acyclic subclass of AFNs and work only with bounded hierarchical EAFCs. Consequently, we have decided to focus on the extension–based semantics for ADFs which can be used without such restrictions.

Our research falls into the area of framework intertranslatability [12, 14, 17, 22, 23]. However, in this case we are moving from less to more complex structures, not the other way around. Moreover, the fact that we are working with ADFs means that the currently established methods are not particularly applicable. To the best of our knowledge, our work is the first one to focus on analyzing the relations between ADFs and other argumentation frameworks.

References

1. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence. 1st edn. Springer (2009)
2. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77** (1995) 321–357
3. Brewka, G., Polberg, S., Woltran, S.: Generalizations of Dung frameworks and their role in formal argumentation. Intelligent Systems, IEEE **29**(1) (January 2014) 30–38
4. Brewka, G., Woltran, S.: Abstract dialectical frameworks. In: Proc. of KR 2010, AAAI Press (2010) 102–111
5. Strass, H.: Approximating operators and semantics for abstract dialectical frameworks. Artif. Intell. **205** (2013) 39 – 70
6. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract dialectical frameworks revisited. In: Proc. of IJCAI 2013, AAAI Press (2013) 803–809
7. Polberg, S.: Extension-based semantics of abstract dialectical frameworks. In: Proc. of STAIRS 2014. Volume 264 of FAIA., IOS Press (2014) 240–249
8. Polberg, S.: Revisiting extension–based semantics of abstract dialectical frameworks. Technical Report DBAI-TR-2015-88, Institute for Information Systems, Technical University of Vienna (2015)
9. Strass, H., Wallner, J.P.: Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In: Proc. of KR 2014, Vienna, Austria, AAAI Press (July 2014) 101–110
10. Strass, H.: Instantiating knowledge bases in abstract dialectical frameworks. In: Proc. of CLIMA 2013. Volume 8143 of LNCS., Springer (2013) 86–101
11. Ellmauthaler, S., Strass, H.: The DIAMOND system for computing with abstract dialectical frameworks. In: Proc. of COMMA 2014. Volume 266 of FAIA., IOS Press (2014) 233–240
12. Cayrol, C., Lagasquie-Schiex, M.C.: Bipolarity in argumentation graphs: Towards a better understanding. Int. J. Approx. Reasoning **54**(7) (2013) 876–899
13. Nouioua, F.: AFs with necessities: Further semantics and labelling characterization. In: Proc. of SUM 2013. Volume 8078 of LNCS. Springer Berlin Heidelberg (2013) 120–133
14. Polberg, S., Oren, N.: Revisiting support in abstract argumentation systems. In: Proc. of COMMA 2014. Volume 266 of FAIA., IOS Press (2014) 369–376
15. Nielsen, S., Parsons, S.: A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In: Proc. of ArgMAS 2006. Volume 4766 of LNCS. Springer (2007) 54–73
16. Modgil, S., Prakken, H.: Reasoning about preferences in structured extended argumentation frameworks. In: Proc. of COMMA 2010. (2010) 347–358
17. Modgil, S., Bench-Capon, T.J.M.: Metalevel argumentation. J. Log. Comput. **21**(6) (2011) 959–1003
18. Modgil, S.: Revisiting abstract argumentation frameworks. In Black, E., Modgil, S., Oren, N., eds.: Theory and Applications of Formal Argumentation. Volume 8306 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2014) 1–15
19. Bench-Capon, T.J.M.: Persuasion in practical argument using value-based argumentation frameworks. J. Log. Comput. **13**(3) (2003) 429–448
20. Amgoud, L., Vesic, S.: A new approach for preference-based argumentation frameworks. Ann. Math. Artif. Intell. **63** (2011) 149–183
21. Modgil, S.: Reasoning about preferences in argumentation frameworks. Artif. Intell. **173**(9-10) (2009) 901–934
22. Oren, N., Reed, C., Luck, M.: Moving between argumentation frameworks. In: Proc. of COMMA 2010, Amsterdam, The Netherlands, The Netherlands, IOS Press (2010) 379–390
23. Boella, G., Gabbay, D.M., van der Torre, L., Villata, S.: Meta-argumentation modelling I: Methodology and techniques. Studia Logica **93**(2-3) (2009) 297–355