5

Abstract Dialectical Frameworks

Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes P. Wallner, Stefan Woltran

ABSTRACT. This handbook chapter describes abstract dialectical frameworks, or ADFs for short. ADFs are generalizations of the widely used Dung argumentation frameworks. Whereas the latter focus on a single relation among abstract arguments, namely attack, ADFs allow arbitrary relationships among arguments to be expressed. For instance, arguments may support each other, or a group of arguments may jointly attack another one while each single member of the group is not strong enough to do so. This additional expressiveness is achieved by handling acceptance conditions for each argument explicitly.

The semantics of ADFs are inspired by approximation fixpoint theory (AFT), a general algebraic theory for approximation based semantics developed by Denecker, Marek and Truszczyński. We briefly introduce AFT and discuss its role in argumentation. This puts us in a position to formally introduce ADFs and their semantics. In particular, we show how the most important Dung semantics can be generalized to ADFs. Furthermore, we illustrate the use of ADFs as semantical tool in various modelling scenarios, demonstrating how typical representations in argumentation can be equipped with precise semantics via translations to ADFs. We also present GRAPPA, a related approach where the semantics of arbitrary labelled argument graphs can be directly defined in an ADF-like manner, circumventing the need for explicit translations. Finally, we address various computational aspects of ADFs, like complexity, expressiveness and realizability, and present several implemented systems.

1 Introduction

This chapter is about abstract dialectical frameworks, or ADFs for short. ADFs are generalizations of Dung argumentation frameworks (AFs, see Chapter 4 of this Handbook). AFs are very popular tools in argumentation. They abstract away from the content of particular arguments and focus on conflicts among arguments, where each argument is viewed as an atomic item. The only information AFs take into account is whether an argument attacks another one or not. Based on a set of arguments and an attack relation, different AF semantics single out coherent subsets of arguments which "fit" together, according to specific criteria. More formally, an AF semantics takes an argumentation framework as input and produces as output a collection of sets of arguments, called extensions.

AFs are typically not used directly for knowledge representation purposes, but as semantical tools: given a knowledge base KB in some knowledge representation formalism, the set of arguments induced by KB is formally defined and the attack relation on these arguments is identified. This defines an AF that can be evaluated according to a chosen semantics. The KB formulas supported by accepted arguments are then the ones which are accepted. This stepwise evaluation is often referred to as the argumentation process [Caminada and Amgoud, 2007].

Given that AFs are in wide use, a natural question to ask is why a generalization of AFs is useful in the first place. There are at least two possible answers to this question:

- the generalization is more expressive than AFs,
- the generalization allows for easier modelling.

In fact, it turns out that both answers apply to ADFs. We will discuss the issue of expressiveness in detail in Section 6.2. For the time being let us focus on the modelling issue. AFs restrict their attention to the attack relation, and the basic intuition is the following: assume an argument b is attacked by argument c, then whenever c is accepted b is defeated. But how about more fine-grained – or entirely different – relations which could be of potential interest? What if c alone is not strong enough and a second argument, say d, is needed to jointly defeat b? And, maybe even more importantly, aren't there situations where accepting an argument can be a reason for accepting another one, in other words, where arguments are in support rather than in attack relation? We do not claim here that examples like the ones just discussed cannot be modelled at all with AFs. However, additional nodes in the AF argument graph will be needed which have the sole purpose of modelling other relations indirectly, via attack. These nodes will often be entirely unrelated to the original knowledge base and thus meaningless from the perspective of the application.

Indeed, for these reasons many authors have felt the need to extend the functionality available in AFs in one way or another. Examples of extensions described in the literature are preference or value-based AFs [Simari and Loui, 1992; Amgoud and Cayrol, 2002; Amgoud and Vesic, 2011; Bench-Capon, 2003], AFs with support relations [Cayrol and Lagasquie-Schiex, 2013; Oren and Norman, 2008; Polberg and Oren, 2014], necessities [Nouioua, 2013], set attacks [Nielsen and Parsons, 2007], attacks on attacks [Modgil, 2009], recursive attacks [Baroni *et al.*, 2011] and AFs with weights [Martínez *et al.*, 2008; Dunne *et al.*, 2011; Coste-Marquis *et al.*, 2012] or probabilities [Hunter, 2013; Thimm, 2012]. We refer the reader to [Brewka *et al.*, 2014] for an overview of such extensions.

In a nutshell, ADFs are an attempt to unify several of these different approaches and to generalize AFs in a principled, systematic way. The basic idea is very simple. Consider again the conditions under which an argument, say b, with attackers c and d is accepted in an AF: b is accepted iff c is not accepted

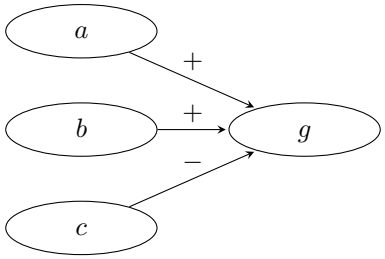


Figure 1: An argument with two supporters and one attacker.

and d is not accepted. This condition can easily be expressed as the propositional formula $\neg c \wedge \neg d$. The acceptance condition for each argument in an AF is obtained in exactly the same way, by constructing the conjunction of the negations of its attackers. Once the implicit acceptance conditions which are at work in AFs are made explicit this way, the generalization ADFs build upon are pretty straightforward: rather than using implicit acceptance conditions of the form we just saw, ADFs use explicit acceptance conditions which can conveniently be expressed as arbitrary propositional formulas.

Let us see how explicit acceptance conditions allow us to handle some of the examples discussed above. We start with joint attack. If b can only be defeated jointly by c and d, then all we have to do is change the acceptance condition accordingly: rather than a conjunction, we have to use the disjunction $\neg c \vee \neg d$ as acceptance condition for b. The effect is that b is only defeated when both c and d are accepted, as intended. As soon as one of them is not accepted, b is no longer defeated.

Support can be handled in a similar manner. Assume g has two supporting arguments a and b, and one attacking argument c, as illustrated in Figure 1. We use $+$ and $-$ to indicate support and attack, respectively.

Note that the information about supporting and attacking links in the graph does not sufficiently specify under what conditions g should be accepted. Let us call a link active if its source node is accepted. There are various options we may want to choose, all of them expressible as a particular acceptance condition for g:

- no negative and all positive links must be active: $\neg c \wedge (a \wedge b)$
- no negative and at least one positive link must be active: $\neg c \wedge (a \vee b)$
- no negative or both positive links must be active: $\neg c \vee (a \wedge b)$
- no negative or at least one positive link must be active: $\neg c \vee (a \vee b)$
- more positive than negative links must be active: $(\neg c \wedge (a \vee b)) \vee (a \wedge b)$

Note how it depends on the acceptance condition whether supporting links are "stronger than" attacking links (meaning that if all incoming links are active, the node is accepted), as in the last three items, or attacking links are "stronger than" supporting links (meaning that if all incoming links are active, the node is rejected), as for the first two items.

We hope these examples are sufficient to illustrate the additional modelling capabilities ADFs provide, and also the simplicity of the basic idea they rest upon. We will see, however, that generalizing the AF semantics to ADFs is far from being simple. This issue will be addressed in Section 3.

In spite of their additional expressiveness, we do not view ADFs primarily as a knowledge representation formalism. We rather consider them as "argumentation middleware", that is, as a framework which is particularly useful for providing semantics to other, maybe more user-friendly formalisms via translations [Brewka *et al.*, 2014]. We will further illustrate this in Section 4.

The rest of this chapter is organized as follows. In Section 2 we recall some relevant background and in particular discuss some relationships between approximation fixpoint theory and AFs which will be useful later. Section 3 introduces ADFs and their semantics formally. The presentation of this section is based on [Brewka *et al.*, 2013]. Section 4 illustrates the role of ADFs in argumentation, showing how they can be used for modelling. Section 5 describes GRAPPA (GRaph-based Argument Processing based on Patterns of Acceptance) along the lines of [Brewka and Woltran, 2014]; GRAPPA is an approach to graph-based argumentation which is closely related to ADFs and their underlying formal techniques. Section 6 discusses subclasses, computational aspects, and expressivity of ADFs. Section 6.1 focuses on an interesting special case of ADFs, so-called bipolar ADFs where each link in the ADF graph is attacking or supporting (or both). This rather expressive class is not only of practical interest, but also has nice computational properties. Expressiveness of ADFs and bipolar ADFs is investigated in Section 6.2, computational complexity in Section 6.3, and recent systems in Section 6.4. Section 7 concludes the chapter.

2 Approximation Fixpoint Theory in Abstract Argumentation

Denecker, Marek and Truszczyński [Denecker *et al.*, 2000] (henceforth shortened to DMT) introduced an algebraic framework for studying semantics of knowledge representation formalisms. In this framework – approximation fixpoint theory (AFT) – knowledge bases are associated with operators (functions) on algebraic structures (for example lattices). The fixpoints of those operators are then studied in order to analyse the semantics of knowledge bases. While this technique is standard to define semantics of programming languages and has indeed been used in early works on logic programming [van Emden and Kowalski, 1976], the major invention of DMT has been the important concept of an approximation of an operator. In the study of semantics of knowledge representation formalisms, elements of lattices represent objects of interest.

Operators transform such objects into others according to the contents of a given knowledge base. Consequently, fixpoints of such operators are then objects that cannot be updated any more – informally speaking, the knowledge base can neither add information to a fixpoint nor remove information from it.

In classical approaches to fixpoint-based semantics, the underlying algebraic structure is the complete lattice of the set $\mathcal{V}_2 = \{v : A \to \{\mathbf{t}, \mathbf{f}\}\}$ of all two-valued interpretations over some vocabulary A ordered by the truth ordering $\leq_t$ with

$$v_1 \leq_t v_2 \text{ if and only if } \forall a \in A : v_1(a) = \mathbf{t} \implies v_2(a) = \mathbf{t}.^1$$

Consequently, an operator O on this lattice $(\mathcal{V}_2, \leq_t)$ takes as input a two-valued interpretation $v \in \mathcal{V}_2$ and returns a revised interpretation $O(v) \in \mathcal{V}_2$. The intuition of the operator is that the revised interpretation $O(v)$ incorporates additional knowledge that is induced by the knowledge base associated to O from interpretation v. Based on this intuition, fixpoints of O correspond to the models of the knowledge base.

To study fixpoints of operators O, DMT investigate fixpoints of their *approximating operators* $\mathcal{O}$. When O operates on two-valued interpretations $\mathcal{V}_2$, its approximation $\mathcal{O}$ operates on *three-valued* interpretations $\mathcal{V}_3 = \{v : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}\}$. The three truth values $\mathbf{t}$ (true), $\mathbf{f}$ (false), and $\mathbf{u}$ (undefined) can be ordered by the information ordering $\leq_i$. This ordering intuitively assigns a greater information content to the classical truth values $\{\mathbf{t}, \mathbf{f}\}$ than to undefined $\mathbf{u}$; more formally, we have $\mathbf{u} <_i \mathbf{t}$ and $\mathbf{u} <_i \mathbf{f}$ and $\leq_i$ is the reflexive transitive closure of $<_i$. The partially ordered set $(\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}, \leq_i)$ forms a complete meet-semilattice with the meet operation $\sqcap_i$.[2] This meet can be read as *consensus* and assigns $\mathbf{t} \sqcap_i \mathbf{t} = \mathbf{t}$, $\mathbf{f} \sqcap_i \mathbf{f} = \mathbf{f}$, and returns $\mathbf{u}$ otherwise. The ordering $\leq_i$ can be generalized to three-valued interpretations in a pointwise fashion:

$$v_1 \leq_i v_2 \text{ if and only if } \forall a \in A : v_1(a) \in \{\mathbf{t}, \mathbf{f}\} \implies v_1(a) = v_2(a).^3$$

Again, the resulting algebraic structure is a complete meet-semilattice; its $\leq_i$-maximal elements are exactly the two-valued interpretations $\mathcal{V}_2$, which form an $\leq_i$-antichain. Intuitively, in that complete meet-semilattice, a single three-valued interpretation

$$v : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$$

serves to approximate a set $[v]_2 = \{w \in \mathcal{V}_2 \mid v \leq_i w\}$ of two-valued interpretations. For example, for the vocabulary $A = \{a, b, c\}$, the three-valued interpretation $v = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{f}\}$ approximates the set $\{w_1, w_2\}$ of two-valued interpretations where $w_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{f}\}$ and $w_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}\}$.

[1] $(\mathcal{V}_2, \leq_t)$ is isomorphic to $(2^A, \subseteq)$ via $v \mapsto v^{-1}(\mathbf{t}) = \{a \in A \mid v(a) = \mathbf{t}\}$.

[2] A complete meet-semilattice is such that every non-empty finite subset has a greatest lower bound, the meet; and every non-empty directed subset has a least upper bound. A subset is directed iff any two of its elements have an upper bound in the set.

[3] $(\mathcal{V}_3, \leq_i)$ is isomorphic to $(\{M \subseteq A \cup \{\neg a \mid a \in A\} \mid a \in M \implies \neg a \notin M\}, \subseteq)$ via the mapping $v \mapsto \{a \in A \mid v(a) = \mathbf{t}\} \cup \{\neg a \mid a \in A, v(a) = \mathbf{f}\}$.

In a similar vein, a three-valued operator $\mathcal{O} : \mathcal{V}_3 \to \mathcal{V}_3$ *approximates* a two-valued operator $O : \mathcal{V}_2 \to \mathcal{V}_2$ if and only if

1. for all $v \in \mathcal{V}_2$, we have $\mathcal{O}(v) = O(v)$ ($\mathcal{O}$ agrees with O on two-valued v), and
2. for all $v_1, v_2 \in \mathcal{V}_3$, $v_1 \leq_i v_2 \implies \mathcal{O}(v_1) \leq_i \mathcal{O}(v_2)$ ($\mathcal{O}$ is $\leq_i$-monotone).

DMT [Denecker *et al.*, 2000] showed that in this case fixpoints of $\mathcal{O}$ approximate fixpoints of O. More specifically, for every fixpoint v_2 of O, there is a fixpoint v_3 of $\mathcal{O}$ such that $v_2 \in [v_3]_2$. Moreover, an approximating operator $\mathcal{O}$ always has a fixpoint, which need not be the case for two-valued operators O. In particular, $\mathcal{O}$ has an $\leq_i$-least fixpoint, which approximates *all* fixpoints of O.

In subsequent work, DMT [Denecker *et al.*, 2004] presented a general, abstract way to define the most precise approximation of a given operator $O : \mathcal{V}_2 \to \mathcal{V}_2$. Most precise here refers to a generalisation of $\leq_i$ to operators, where for $\mathcal{O}_1, \mathcal{O}_2 : \mathcal{V}_3 \to \mathcal{V}_3$, they define $\mathcal{O}_1 \leq_i \mathcal{O}_2$ iff for all $v \in \mathcal{V}_3$ it holds that $\mathcal{O}_1(v) \leq_i \mathcal{O}_2(v)$. Specifically, DMT then show that the most precise – called the *ultimate* – approximation of O is given by the operator $\mathcal{U}_O : \mathcal{V}_3 \to \mathcal{V}_3$ that maps a given $v \in \mathcal{V}_3$ to

$$\mathcal{U}_O(v) : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \quad \text{with} \quad a \mapsto \begin{cases} \mathbf{t} & \text{if } w(a) = \mathbf{t} \text{ for all } w \in \{O(x) \mid x \in [v]_2\} \\ \mathbf{f} & \text{if } w(a) = \mathbf{f} \text{ for all } w \in \{O(x) \mid x \in [v]_2\} \\ \mathbf{u} & \text{otherwise} \end{cases}$$

This definition is remarkable since previously, approximations of operators had to be devised by hand rather than automatically derived. DMT [Denecker *et al.*, 2004] give additional definitions introducing stable semantics that are only of minor interest here and will be introduced in a special form later.

AFT on AFs

AFT can be used for defining semantics of AFs as follows [Strass, 2013a]. The stable semantics for AFs can be understood as a two-valued semantics given by the fixpoints of an operator (going back to Pollock [1987]) on two-valued interpretations.

Definition 2.1 *For each* AF $F = (A, R)$*, the operator* $U_F : \mathcal{V}_2 \to \mathcal{V}_2$ *yields – for a given interpretation* $v : A \to \{\mathbf{t}, \mathbf{f}\}$ *– a new interpretation*

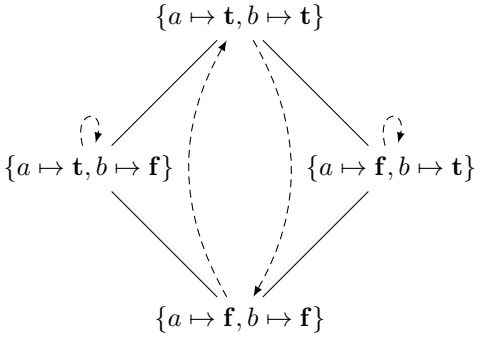
$$U_F(v) : A \to \{\mathbf{t}, \mathbf{f}\} \quad \textit{with} \quad a \mapsto \begin{cases} \mathbf{f} & \textit{if } \exists b \in A : v(b) = \mathbf{t}, (b, a) \in R \\ \mathbf{t} & \textit{otherwise} \end{cases}$$

Intuitively, all arguments that are attacked in F by some argument that is true in v are set to false in U_F and set to true otherwise, that is, if unattacked by all $\mathbf{t}$ arguments of v. (So the U is for "unattacked".) It is easy to see that the fixpoints of this operator exactly correspond to stable extensions [Strass, 2013a, Proposition 4.4].

Proposition 2.2 *Let $F = (A, R)$ be an AF and $v : A \to \{\mathbf{t}, \mathbf{f}\}$ be an interpretation. Then $v = U_F(v)$ iff the set $v^{-1}(\mathbf{t}) = \{a \in A \mid v(a) = \mathbf{t}\}$ is a stable extension of F.*

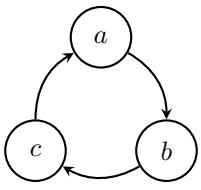
Example 2.3 *Consider the* AF *$F_1 = (A_1, R_1)$ with $A_1 = \{a, b\}$ and $R_1 = \{(a, b), (b, a)\}$:*

Below, we depict the complete lattice $(\{v : A_1 \to \{\mathbf{t}, \mathbf{f}\}\}, \leq_t)$ of two-valued interpretations over A_1 ordered by the truth ordering as a Hasse diagram (i.e. straight lines show direct $\leq_t$-neighbours), and how the operator U_{F_1} assigns its points to others (dashed arrows).

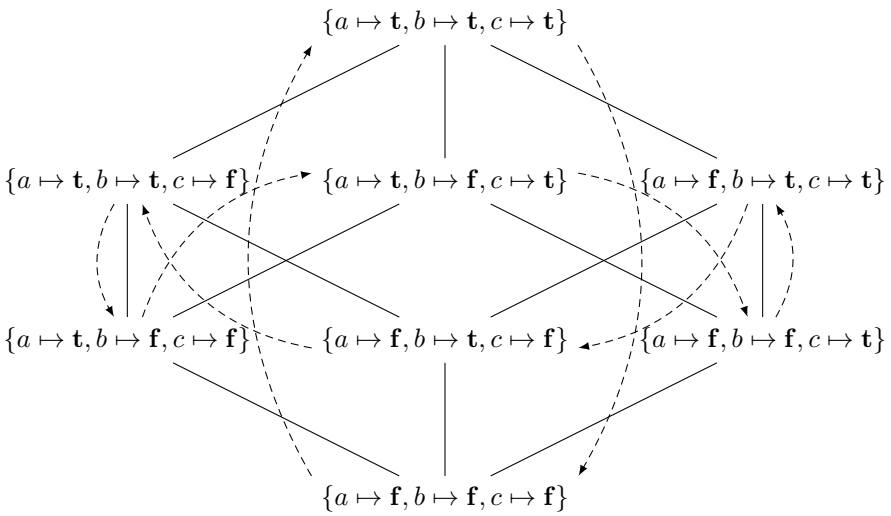


It can be seen from the diagram that the operator has two fixpoints, $\{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}\}$ and $\{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}\}$. They correspond one-to-one to the stable extensions $\{a\}$ and $\{b\}$ of the AF F_1.

Example 2.4 *In contrast, consider the* AF *$F_2 = (A_2, R_2)$ with $A_2 = \{a, b, c\}$ and $R_2 = \{(a, b), (b, c), (c, a)\}$:*



Again, we depict the complete lattice $(\{v : A_2 \to \{\mathbf{t}, \mathbf{f}\}\}, \leq_t)$ and how the operator U_{F_2} assigns its points to others.



The picture makes it obvious that U_{F_2} has no fixpoint, in accordance with the fact that F_2 has no stable extension.

Using the definitions of Denecker, Marek and Truszczyński, it is easy to obtain the ultimate approximation of U_F. (See also [Strass, 2013a, Proposition 4.1].)

Corollary 2.5 *Given an interpretation $v : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, the three-valued operator $\Upsilon_F : \mathcal{V}_3 \to \mathcal{V}_3$ yields a new interpretation*

$$\Upsilon_F(v) : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \quad \textit{with} \quad a \mapsto \begin{cases} \mathbf{f} & \textit{if } \exists b \in A : v(b) = \mathbf{t}, (b, a) \in R \\ \mathbf{t} & \textit{if } \forall b \in A : (b, a) \in R \implies v(b) = \mathbf{f} \\ \mathbf{u} & \textit{otherwise} \end{cases}$$

For any given AF F, the fixpoints of U_F constitute the stable semantics of F. The ultimate approximation Υ_F approximates U_F, thus the semantics induced by Υ_F then intuitively approximate AF stable semantics. More specifically, the following result is straightforward [Strass, 2013a, Section 4]:[4]

Proposition 2.6 *Let $F = (A, R)$ be an AF and $v : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ be an interpretation.*

1. *v is complete for F iff $v = \Upsilon_F(v)$.*

[4]Given an AF $F = (A, R)$, an extension $E \subseteq A$ uniquely determines a three-valued interpretation v_E by letting $v_E(a) = \mathbf{t}$ if $a \in E$, $v_E(a) = \mathbf{f}$ if a is attacked by E in F, and $v_E(a) = \mathbf{u}$ otherwise. Similarly, a three-valued interpretation $v : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ uniquely determines an extension $E_v = \{a \mid v(a) = \mathbf{t}\}$. This allows us to switch freely between extensions and interpretations.

2. *v is admissible for F iff $v \leq_i \Upsilon_F(v)$.*

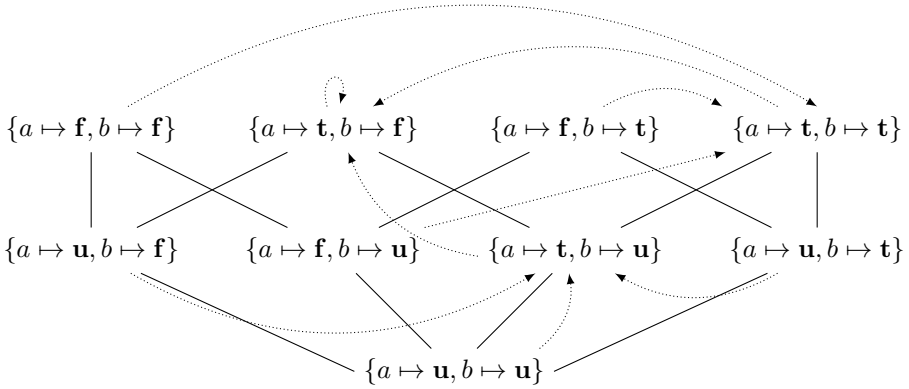
3. *v is preferred for F iff v is $\leq_i$-maximal admissible.*

4. *v is grounded for F iff v is the $\leq_i$-least fixpoint of Υ_F.*

In the next section, we will use approximation fixpoint theory and this result to define the semantics of ADFs in a straightforward way.

Example 2.7 *Consider the AF $F_3 = (A_3, R_3)$ with $A_3 = \{a, b\}$ and $R_3 = \{(a, b)\}$:*

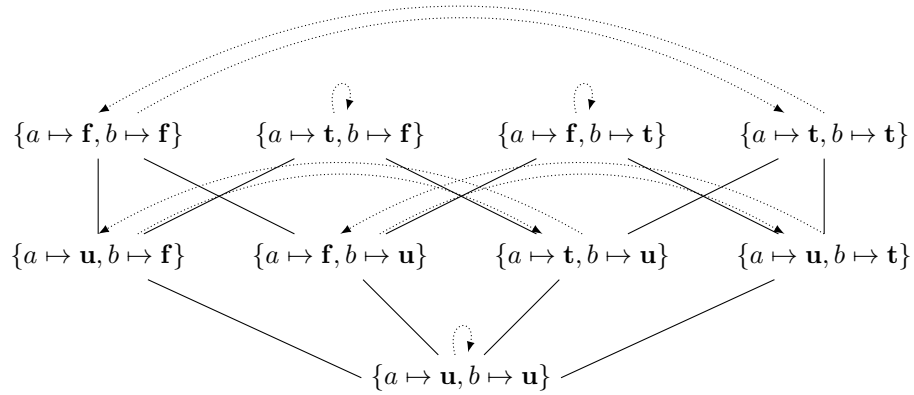
Below, we depict the associated meet-semilattice $(\{v : A_3 \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}\}, \leq_i)$ of the set of all three-valued interpretations over A_3 ordered by the information ordering, and how the operator Υ_{F_3} maps those interpretations to others.



The picture shows how the grounded semantics can be obtained by following the dotted line starting in the $\leq_i$-least element up to the operator's single fixpoint. (In fact, it obviates that all (sufficiently long) sequences of operator applications lead to the fixpoint, showing that this interpretation really is the intended meaning of F_3.)

Example 2.8 *Reconsider the AF $F_1 = (A_1, R_1)$ from Example 2.3 with $A_1 = \{a, b\}$ and $R_1 = \{(a, b), (b, a)\}$:*

Again, we show the complete meet-semilattice $(\{v : A_1 \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}\}, \leq_i)$ *along with the mappings of the operator* Υ_{F_1} *(dotted arrows).*



In this picture, the operator U_{F_1} *re-appears in the top row of all two-valued interpretations. Those form a complete lattice with respect to* $\leq_t$*, but an antichain with respect to* $\leq_i$*. Likewise, the two fixpoints of* U_{F_1} *re-appear as fixpoints of* Υ_{F_1} *in the top row. The additional fixpoint of* Υ_{F_1} *consequently constitutes the grounded semantics of* F_1*.*

As we have seen, the operator Υ_F arises naturally from a straightforward application of ultimate approximation [Denecker *et al.*, 2004] to an operator proposed by Pollock [1987]. It is interesting to observe that the assignments of the operator correspond precisely to what has independently been defined as "legal argument labellings" [Caminada and Gabbay, 2009].

3 ADFs: Syntax and Semantics

Like an AF, an abstract dialectical framework (ADF) is a directed graph whose nodes represent arguments, statements or positions. One can think of the nodes as arbitrary items which can be accepted or not. The links represent dependencies. However, unlike a link in an AF, the meaning of an ADF link can vary. The status of a node s only depends on the status of its parents (denoted $par(s)$), that is, the nodes with a direct link to s. In addition, each node s has an associated acceptance condition C_s specifying the exact conditions under which s is accepted. C_s is a function assigning to each subset of $par(s)$ one of the truth values $\mathbf{t}$, $\mathbf{f}$.[5] Intuitively, if for some $R \subseteq par(s)$ we have $C_s(R) = \mathbf{t}$, then s will be accepted provided the nodes in R are accepted and those in $par(s) \setminus R$ are not accepted.

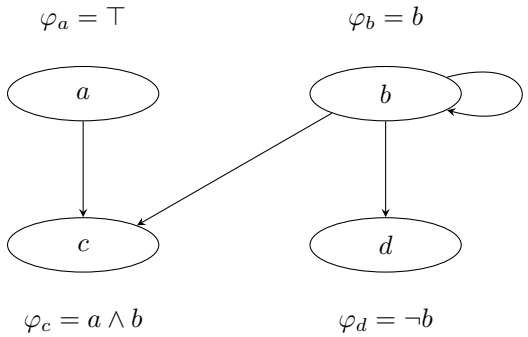
[5] In the original paper *in* and *out* were used. We prefer truth values here as they allow us to apply standard logical terminology.

Definition 3.1 *An* abstract dialectical framework *is a tuple* $D = (S, L, C)$ *where*

- S *is a set of statements (positions, nodes),*
- $L \subseteq S \times S$ *is a set of links,*
- $C = \{C_s\}_{s \in S}$ *is a set of total functions* $C_s : 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$*, one for each statement* s*.* C_s *is called acceptance condition of* s*.*

In many cases it is convenient to represent acceptance conditions as propositional formulas. For this reason we will frequently use a logical representation of ADFs (S, L, C) where C is a collection $\{\varphi_s\}_{s \in S}$ of propositional formulas.[6]

Example 3.2 *In the following* ADF*, which will act as running example throughout the chapter, we use formulas to specify acceptance conditions.*



Intuitively, φ_a *states that* a *should always be accepted. Condition* φ_b *expresses a kind of self-support, which can be utilized as a guess whether or not to accept* b*. Finally,* c *should be accepted if both* a *and* b *are, while* d *is attacked by statement* b*.*

Unless specified differently we will tacitly assume that the acceptance formulas specify the parents a node depends on implicitly. It is then not necessary to give the links in the graph explicitly. We thus can represent an ADF D as a tuple (S, C) where S and C are as above and L is implicitly given as $(a, b) \in L$ iff a appears in φ_b.

The different semantics of ADFs over statements S are based (via approximation fixpoint theory) on the notion of a two-valued model. A two-valued interpretation $v : S \to \{\mathbf{t}, \mathbf{f}\}$ – a mapping from statements to the truth values true and false – is a *two-valued model* (*model*, if clear from the context) of an ADF (S, C) whenever for all statements $s \in S$ we have $v(s) = v(\varphi_s)$, that

[6] More precisely, each acceptance condition C_s will be represented as a propositional formula φ_s over the vocabulary $par(s)$.

is, v maps exactly those statements to true whose acceptance conditions are satisfied under v.[7]

Approximation Fixpoint Theory on ADFs

We now come back to AFT and illustrate its role to define semantics for ADFs [Strass, 2013a; Brewka *et al.*, 2013]. As AFT deals with operator-based semantics and how to approximate them, the starting point is an operator for the two-valued semantics: the notion of an ADF model allows us to associate a two-valued operator to a given ADF.

Definition 3.3 *Let $D = (S, \{\varphi_s\}_{s\in S})$ be an ADF. The operator $G_D : \mathcal{V}_2 \to \mathcal{V}_2$ takes an input $v : S \to \{\mathbf{t}, \mathbf{f}\}$ and returns an updated interpretation*

$$G_D(v) : S \to \{\mathbf{t}, \mathbf{f}\} \quad \textit{with} \quad s \mapsto v(\varphi_s)$$

In words, the operator takes a two-valued interpretation v and returns a two-valued interpretation $G_D(v)$ mapping each $s \in S$ to the truth value that is obtained by evaluating φ_s with v. It is easy to see that this operator characterises the ADF model semantics [Strass, 2013a, Proposition 3.4].

Proposition 3.4 *Let $D = (S, L, C)$ be an ADF and $v : S \to \{\mathbf{t}, \mathbf{f}\}$ be a two-valued interpretation. Then v is a (two-valued) model of D iff $v = G_D(v)$.*

Example 3.5 *For the ADF D from Example 3.2, Figure 2 depicts the complete lattice $(\{v : S \to \{\mathbf{t}, \mathbf{f}\}\}, \leq_t)$ and how the operator G_D assigns its points to others.*

Using the general operator-based definitions of Denecker, Marek and Truszczyński [Denecker *et al.*, 2004], it is again straightforward to determine the ultimate approximation of G_D. Recall from the section on approximation fixpoint theory (Section 2) that the set $\mathcal{V}_3$ of all three-valued interpretations over S forms a complete meet-semilattice with respect to the information ordering $\leq_i$. The consensus meet operation $\sqcap_i$ of this semilattice is given by $(v_1 \sqcap_i v_2)(s) = v_1(s) \sqcap_i v_2(s)$ for all $s \in S$. The least element of this semilattice is the interpretation $v_{\mathbf{u}} : S \to \{\mathbf{u}\}$ mapping all statements to undefined – the least informative interpretation. The ultimate approximation of the two-valued ADF operator G_D is now obtained as follows [Strass, 2013a, Lemma 3.12]:

Corollary 3.6 *Let D be an ADF. The operator $\Gamma_D : \mathcal{V}_3 \to \mathcal{V}_3$ is the ultimate approximation of G_D and is defined as follows: for an ADF D and a three-valued interpretation v, the revised interpretation $\Gamma_D(v)$ is given by*

$$\Gamma_D(v) : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \quad \textit{with} \quad s \mapsto \textstyle\bigsqcap_i \{w(\varphi_s) \mid w \in [v]_2\}$$

[7] In an earlier paper [Brewka *et al.*, 2013], there was the notion of a "three-valued model". The development and analysis of that concept has been discontinued.

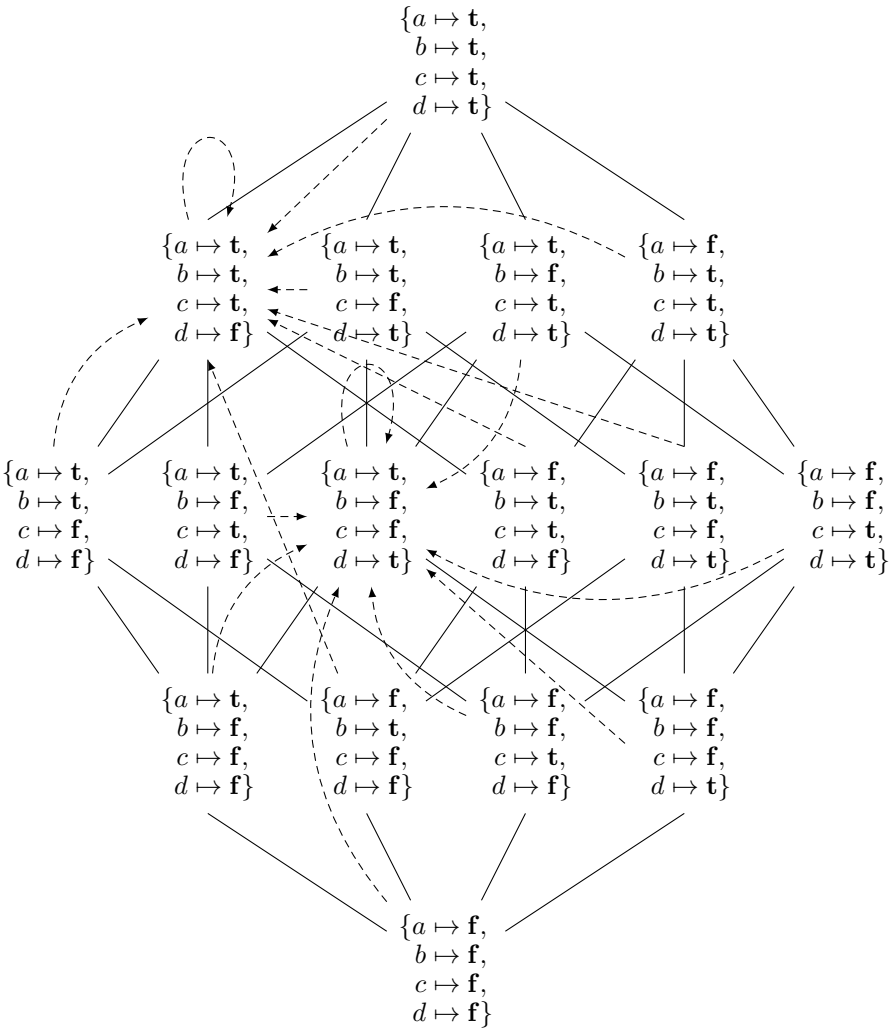
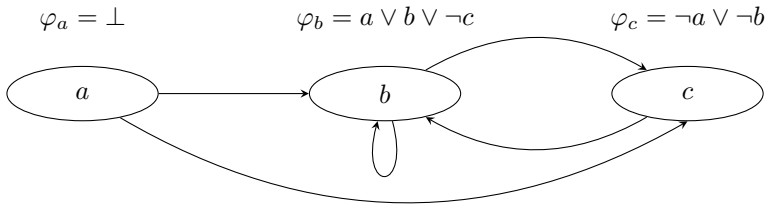


Figure 2: Complete lattice of two-valued interpretations for Example 3.2; dashed arrows visualise the assignments of the operator G_D. It can be readily seen that G_D has two fixpoints, whence D has two models (Proposition 3.4).

That is, for each statement s, the operator returns the consensus truth value for its acceptance formula φ_s, where the consensus takes into account all possible two-valued interpretations w that extend the input valuation v. If this v is two-valued, then $[v]_2 = \{v\}$, thus $\Gamma_D(v)(s) = v(\varphi_s) = G_D(v)(s)$ and Γ_D indeed approximates G_D.

Example 3.7 *Consider the* ADF $D_1 = (S_1, L_1, C_1)$ *given by* $S_1 = \{a, b, c\}$*, and* L_1 *and* C_1 *given as follows:*



Roughly, a cannot be accepted. Statement b supports itself, and is furthermore supported by a and attacked by c – more precisely, b can be accepted if a can be accepted or b can be accepted or c can be rejected. In turn, c is jointly attacked by a and b – c can only be rejected if both a and b are accepted, otherwise c is accepted. Figure 3 shows the associated complete meet-semilattice $(\{v : S_1 \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}\}, \leq_i)$ *along with the mappings of the operator* Γ_{D_1}*.*

It is now an easy corollary of Definition 2.6 to generalize the standard AF semantics to ADFs:

Definition 3.8 *Let* $D = (S, L, C)$ *be an* ADF *and* $v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ *be an interpretation.*

1. *v is* complete *for D iff* $v = \Gamma_D(v)$*.*
2. *v is* admissible *for D iff* $v \leq_i \Gamma_D(v)$*.*
3. *v is* preferred *for D iff v is* $\leq_i$*-maximal admissible.*
4. *v is* grounded *for D iff v is the* $\leq_i$*-least fixpoint of* Γ_D*.*

Incidentally, Brewka and Woltran [2010] already defined the operator Γ_D (manually) and used it to define the grounded semantics. Thus the grounded semantics can be seen as the greatest possible consensus between all acceptable ways of interpreting the ADF at hand. A three-valued interpretation is admissible for an ADF D iff it does not make an unjustified commitment that the operator Γ_D will subsequently revoke.

There is an alternative and perhaps slightly more accessible way of introducing the operator Γ_D. We will briefly pursue this way for illustration, and start

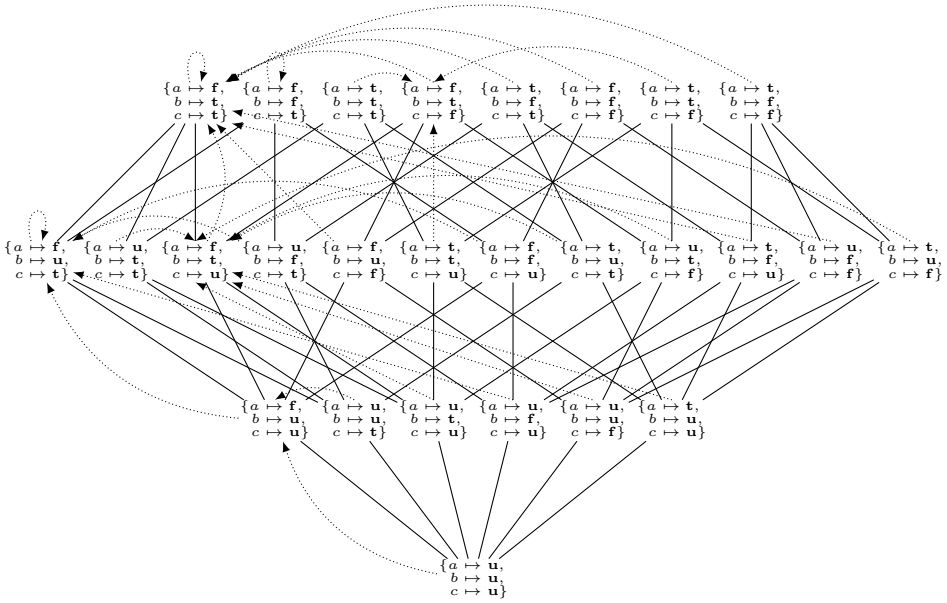


Figure 3: Complete meet-semilattice of three-valued interpretations over $S_1 = \{a, b, c\}$ under the information ordering for Example 3.7; dotted arrows visualise mappings of the operator Γ_{D_1}. It can be seen that Γ_{D_1} has a $\leq_i$-least fixpoint, which is situated right $\leq_i$-beneath its two-valued models, the other two fixpoints of Γ_{D_1}.

out with an additional definition. For a propositional formula φ over vocabulary S and a three-valued interpretation $v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, the *partial valuation of φ by v* is the formula

$$\varphi^v = \varphi[p/\top : v(p) = \mathbf{t}][p/\bot : v(p) = \mathbf{f}]$$

Intuitively, given a three-valued interpretation v and a formula φ, the partial evaluation of φ with v takes the two-valued part of v and replaces the evaluated variables with their truth values. For example, consider the propositional formula $\varphi = a \vee (b \wedge c)$ and the interpretation $v_1 = \{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}, c \mapsto \mathbf{u}\}$. Statement c with $v_1(c) = \mathbf{u}$ will remain in φ, while a and b are replaced, and we get $\varphi^{v_1} = \bot \vee (\top \wedge c)$. Now assume that an ADF $D = (S, \{\varphi_s\}_{s \in S})$ is given via acceptance formulas; for this D and a three-valued interpretation v, the revised

interpretation $\Gamma_D(v)$ is given by

$$\Gamma_D(v) : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \quad \text{with} \quad s \mapsto \begin{cases} \mathbf{t} & \text{if } \varphi_s^v \text{ is irrefutable} \\ \mathbf{f} & \text{if } \varphi_s^v \text{ is unsatisfiable} \\ \mathbf{u} & \text{otherwise} \end{cases}$$

An irrefutable formula is a formula that is satisfied under any two-valued interpretation (i.e. the formula is a tautology).

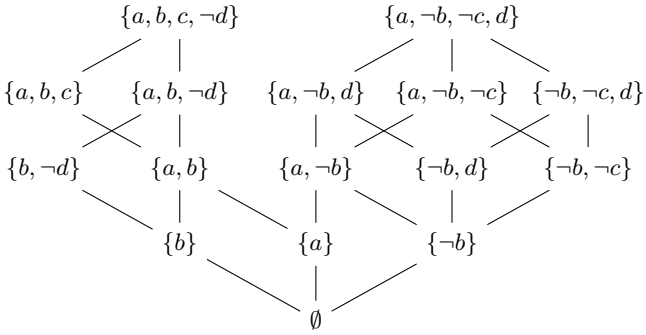
For reasons of brevity, we will sometimes shorten the notation of a three-valued interpretation $v = \{a_1 \mapsto t_1, \ldots, a_n \mapsto t_n,\}$ with statements $a_1, \ldots, a_n$ and truth values $t_1, \ldots, t_n$ to $v \mathrel{\hat{=}} \{a_i \mid v(a_i) = \mathbf{t}\} \cup \{\neg a_i \mid v(a_i) = \mathbf{f}\}$. For instance, $v = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{f}\} \mathrel{\hat{=}} \{a, \neg c\}$.

We now show some concrete interpretations and semantics for an example.

Example 3.9 *As we have seen before, for the* ADF *D from Example 3.2 we obtain the following two-valued models:*

- $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \mathrel{\hat{=}} \{a, b, c, \neg d\}$
- $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \mathrel{\hat{=}} \{a, \neg b, \neg c, d\}$

Unfortunately, due to its sheer size ($3^4 = 81$ interpretations), we cannot depict the semi-lattice $(\{S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}\}, \leq_i)$ and will henceforth resort to textual descriptions. The grounded interpretation of D is $v_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\} \mathrel{\hat{=}} \{a\}$. The admissible interpretations (ordered by $\leq_i$) of our example ADF *are as follows:*



We verify that $v_4 \mathrel{\hat{=}} \{a, \neg b, \neg c\}$ is admissible in the example ADF*. Statement a's acceptance condition is a tautology. This means that under any three-valued interpretation v' it holds that $\Gamma_D(v')(a) = \mathbf{t}$, and, in particular, $\Gamma_D(v_4)(a) = v_4(a) = \mathbf{t}$. Acceptance condition of statement b is the formula b. Such an acceptance condition (a single unnegated variable) implies that for any three-valued interpretation v' that assigns a value to b, it holds that $\Gamma_D(v')(b) = v'(b)$. If b is assigned $\mathbf{t}$ by v', then $\varphi_b^{v'}$ is a tautology, if b is assigned*

f, *then* $\varphi_b^{v'}$ *is unsatisfiable, and if* b *is assigned* **u** *by* v', *then* $\varphi_b^{v'} = b$ *is neither a tautology nor unsatisfiable. The acceptance condition of statement* c *is* $a \wedge b$. *Evaluating* φ_c *under* v_4 *gives* $\varphi_c^{v_4} = \top \wedge \bot \equiv \bot$, *and* $\Gamma_D(v_4)(c) = \mathbf{f} = v_4(c)$. *Finally,* $v_4(d) = \mathbf{u}$ *and* $\varphi_d = \neg b$. *Since for the undefined truth value it holds that* $\mathbf{u} \leq_i \mathbf{t}$ *and* $\mathbf{u} \leq_i \mathbf{f}$, *if a three-valued interpretation* v' *assigns undefined to a statement, then applying the operator* Γ_D *under* v' *cannot return a truth value with less information than* **u** *for that statement. For our example interpretation, we have* $v_4(d) \leq_i \Gamma_D(v_4)(d) = \mathbf{t}$.

The complete interpretations of our example ADF *are*

$$v_3 \mathrel{\hat{=}} \{a\}, \quad v_5 \mathrel{\hat{=}} \{a, b, c, \neg d\}, \quad v_6 \mathrel{\hat{=}} \{a, \neg b, \neg c, d\}.$$

The latter two, v_5 *and* v_6, *are the preferred interpretations.*

The definition of stable model semantics for ADFs [Brewka *et al.*, 2013] is based on ideas from Logic Programming (LP) where stable models strengthen the notion of minimal models by excluding self-justifying cycles of atoms. In LP, this is achieved by a test which picks a candidate model M, uses M to reduce the original logic program to a program without negative literals, and then checks whether M coincides with the (typically unique) least model of the reduced program. This way self-justifying cycles cannot appear. What we do for an ADF D is very similar: to check whether a two-valued model v of D is *stable* we do the following:

- we eliminate in D all nodes with v-value **f** and corresponding links,
- we replace eliminated nodes in acceptance conditions by $\bot$,
- we check whether nodes that are **t** in v coincide with those that are **t** in the grounded interpretation of the reduced ADF.

This is captured in the following definition [Brewka *et al.*, 2013, Definition 6]. (See also [Strass and Wallner, 2015, Proposition 2.4] for an alternative definition via AFT.)

Definition 3.10 *Let* $D = (S, L, C)$ *be an* ADF *with* $C = \{\varphi_s\}_{s \in S}$ *and* $v : S \to \{\mathbf{t}, \mathbf{f}\}$ *be a two-valued model of* D. *Define the reduced* ADF D^v *with* $D^v = (S^v, L^v, C^v)$, *where*

- $S^v = \{s \in S \mid v(s) = \mathbf{t}\}$
- $L^v = L \cap S^v \times S^v$
- $C^v = \{\varphi_s^v\}_{s \in S^v}$ *where for each* $s \in S^v$, *we set* $\varphi_s^v = \varphi_s[b/\bot : v(b) = \mathbf{f}]$.

Denote by w *the unique grounded interpretation of* D^v. *Now the two-valued model* v *of* D *is a* stable model *of* D *if and only if for all* $s \in S$, *we find that* $v(s) = \mathbf{t}$ *implies* $w(s) = \mathbf{t}$.

Note that a stable model of an ADF D is a model of D by definition (v is assumed to be a model). In the reduct for a model v, (i) only statements assigned to true by v are present, (ii) only links with both ends being statements assigned to true by v are considered, and (iii) in each acceptance formula of the remaining statements we replace statements $b \in S$ that v maps to false by their truth value, i.e., in these acceptance conditions variables assigned to false by v are replaced by $\bot$ (and the remaining statements/variables remain unmodified in the formulas). This definition straightforwardly expresses the intuition underlying stable models: if all statements the model v takes to be false are indeed false, we must find a constructive proof for all statements the model takes to be true.

Example 3.11 *Consider the* ADF D *given by*

$$\varphi_a = \top, \quad \varphi_b = \neg a \vee c, \quad \varphi_c = b.$$

It has two models: $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}\}$ *and* $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}\}$. *Let us check whether they are stable models. For* v_1*, the reduct,* D^{v_1}*, is equal to* D *(every statement is assigned to true by* v_1*, thus all statements and links remain in the reduct and no statement is replaced by* $\bot$ *in an acceptance condition). The grounded interpretation of* D *is* $v_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}\}$*, implying that* v_1 *is not stable in* D*, since the grounded interpretation of* D^{v_1} *is not equal to* v_1*.*[8]

For the other model of D*, the reduct* $D^{v_2} = (S^{v_2}, L^{v_2}, C^{v_2})$ *with* $S^{v_2} = \{a\}$*,* $L^{v_2} = \emptyset$*, and* $\varphi_a = \top$*. The grounded interpretation of* D^{v_2} *is* $v_4 = \{a \mapsto \mathbf{t}\}$*. The final condition of Definition 3.10,* $v_2(a) = \mathbf{t}$ *implies* $v_4(a) = \mathbf{t}$*, is satisfied, and, therefore,* v_2 *is a stable model of* D*. Further,* v_2 *is the only stable model of* D*, since we considered all models of* D*, only one being stable, and any other interpretation cannot be stable for* D*, since being a model is a prerequisite for being stable.*

Next, we illustrate that there are cases where an ADF has a model, but no stable model.

Example 3.12 *Consider the* ADF D *given by*

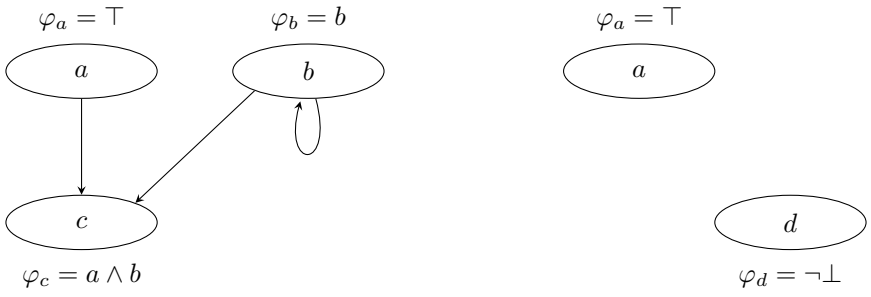
$$\varphi_a = c, \quad \varphi_b = c, \quad \varphi_c = a \leftrightarrow b.$$

[8] The definition of stable models in this chapter, taken from [Brewka *et al.*, 2013, Definition 6], supersedes the definition of stable models in the original paper on ADFs [Brewka and Woltran, 2010, Definition 6] in that the new definition corrects certain unintended results. For instance, v_1 in Example 3.11 is the only stable model according to the old definition, but this is not the case under the new definition. The model v_1 violates the basic intuition of stable semantics that all elements of a stable model should have a non-cyclic justification: in the model v_1 it holds that b is accepted because c is and vice versa (these two statements have supporting links to each other; see Section 6.1 for a formalization of attacking and supporting links between statements).

The only two-valued model of D is $v = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}\}$. Since c is true because a and b are and vice versa, the model contains unintended cyclic support and thus should not be stable. Indeed, for the reduct we get $D^v = D$. Let us compute the grounded semantics of D. We start with interpretation $w = \{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}\}$. Since none of the acceptance formulas is a tautology or an unsatisfiable formula, w is already a fixpoint of Γ_D and thus the grounded interpretation of D. Hence v is not a stable model and D has no stable models, just as intended. Since v is a minimal model of D the example illustrates that in Definition 3.10 we actually need the grounded semantics; requiring v to be among the (subset-inclusion or information) minimal two-valued models of the reduct is insufficient, in contrast to, e.g., stable semantics of logic programs.

For our running example, the concept of reduct is applied as follows.

Example 3.13 *The ADF from Example 3.2 has two two-valued models, namely $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\}$ and $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\}$. We obtain the reducts for each model of D as follows:*



Reduct D^{v_1} *Reduct D^{v_2}*

The grounded interpretation of reduct D^{v_1} is $\{a\}$, v_1 is thus not a stable model of D. For v_2, the reduct D^{v_2} has the grounded interpretation $\{a \mapsto \mathbf{t}, d \mapsto \mathbf{t}\}$. The model v_2 of D is thus the single stable model of D.

Well-known relationships between semantics defined on Dung AFs carry over to ADFs. This is formalized in the next theorem [Brewka *et al.*, 2013, Theorem 3].

Theorem 3.14 *Let D be an* ADF.

- *Each stable model of D is a two-valued model of D;*
- *each two-valued model of D is a preferred interpretation of D;*
- *each preferred interpretation of D is complete;*

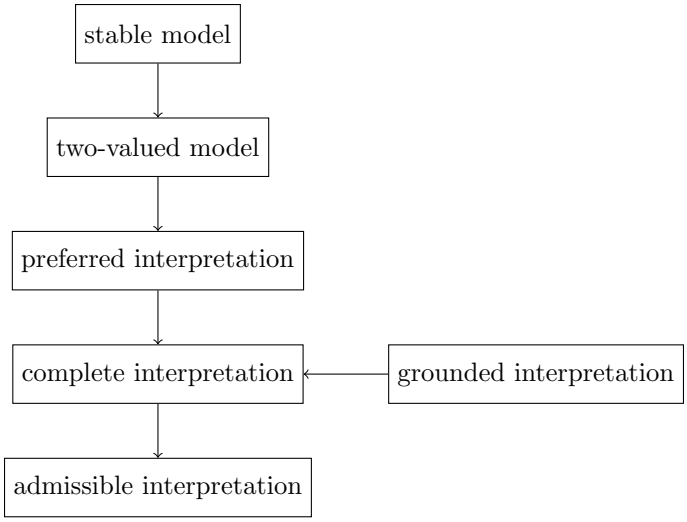


Figure 4: Relations between ADF *semantics*

- *each complete interpretation of D is admissible;*
- *the grounded interpretation of D is complete.*

We illustrate the relationships in Figure 4 where an arrow from a σ-interpretation to a τ-interpretation denotes that every σ-interpretation is a τ-interpretation. Further, again similarly as in AFs, any ADF possesses at least one admissible, complete, preferred, and grounded interpretation, while this is not guaranteed for models and stable models.

In addition to the semantical relationships generalizing those known from AFs, semantics on ADFs also directly generalize semantics for AFs. We first define for a given AF its associated ADF.

Definition 3.15 *For an* AF $F = (A, R)$*, define the* ADF associated to F *as* $D_F = (A, R, C)$ *with* $C = \{\varphi_a\}_{a \in A}$ *such that for each* $a \in A$*, the acceptance condition is given by*

$$\varphi_a = \bigwedge_{\substack{b \in A, \\ (b,a) \in R}} \neg b$$

Now we can formalize the way ADFs, and their semantics, generalize AFs in the next two theorems [Brewka *et al.*, 2013].

Theorem 3.16 *Let $F = (A, R)$ be an* AF *and D_F its associated* ADF*. For any two-valued interpretation v for A, the following are equivalent:*

(A) the set $v^{-1}(\mathbf{t}) = \{a \in A \mid v(a) = \mathbf{t}\}$ *is a stable extension of* F*,*

(B) v *is a stable model of* D_F*,*

(C) v *is a two-valued model of* D_F*.*

Note that for AF-based ADFs, there is no distinction between models and stable models. The intuitive explanation for this is that stable semantics on ADFs breaks cyclic supports, which cannot arise in AFs because they cannot (directly) express support.

More generally, we can also show that our definitions are indeed proper generalizations of Dung's notions for AFs as given in Proposition 2.6. The result is due to [Brewka *et al.*, 2013].

Theorem 3.17 *Let* F *be an* AF *and* D_F *its associated* ADF*. An interpretation is admissible, complete, preferred, grounded for* F *iff it is admissible, complete, preferred, grounded for* D_F*.*

On AFs, if v is a preferred interpretation (a stable model) for an AF F it holds that there is no preferred interpretation (stable model) $v' \neq v$ such that the set of statements assigned to true by v is a subset of the statements assigned to true by v', i.e., $\{s \mid v(s) = \mathbf{t}\} \not\subseteq \{s \mid v'(s) = \mathbf{t}\}$. On general ADFs, this property does not hold for preferred interpretations and two-valued models, i.e., there are ADFs with two preferred interpretations (models) v and v' such that $\{s \mid v(s) = \mathbf{t}\} \subseteq \{s \mid v'(s) = \mathbf{t}\}$.

Example 3.18 *Consider* ADF $D = (\{a\}, \{(a, a)\}, \{\varphi_a = a\})$*. Both* $v_1 = \{a \mapsto \mathbf{f}\}$ *and* $v_2 = \{a \mapsto \mathbf{t}\}$ *are models and preferred interpretations of* D*. It holds that* $\{s \mid v(s) = \mathbf{t}\} = \emptyset \subsetneq \{a\} = \{s \mid v'(s) = \mathbf{t}\}$.

On the other hand, for any ADF D with stable models v_1 and v_2, it holds that $v_1 \leq_t v_2$ implies $v_1 = v_2$ [Strass, 2013a, Proposition 3.8], that is, such strict relationships cannot occur between *stable* models. (This follows easily from AFT.)

4 ADFs as Modelling Tools

In this section we will provide various examples illustrating why – as we believe – ADFs are useful tools in formal argumentation. We discussed the term argumentation middleware in the introduction already. We now want to give a clearer picture what we actually mean by this. More precisely, we will discuss various graphical representations of argumentation scenarios users may find useful. In each case we define the semantics of the chosen representation by providing a formal translation to ADFs. The representation is thus equipped – via the translation – with the whole range of Dung semantics we have defined for ADFs. We also discuss how ADFs can serve as a tool for providing semantics to systems based on strict and defeasible inference rules, again via a translation.

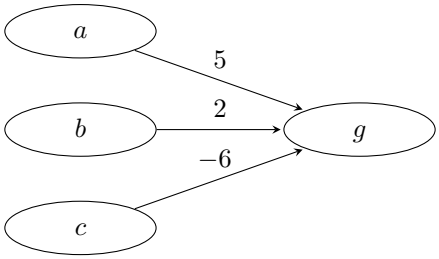


Figure 5: An argument graph with weighted links.

4.1 Weights and Preferences

In our informal discussion in the introduction we have already shown how graphical representations based on link types (+ for supporting, − for attacking) can be modeled using ADFs. The same is obviously true for links annotated with numerical weights. Throughout the chapter we will assume a positive weight represents support, a negative weight attack, in both cases with a given strength. An example can be found in Figure 5.

The figure uses a weighted graph to represent a simple argumentation scenario. We will provide the graph with a formal semantics based on translating it to an ADF. There are various ways of interpreting the numbers and of actually deriving specific ADF acceptance conditions from representations like this one. We first have to specify how the numbers should actually be used to decide whether a node is accepted or not. Recall that a link is active if its source node is accepted. A straightforward idea is to accept a node whenever the sum of the weights of all active links pointing to the node is positive. We will call this strategy *sum-of-weights* (sow). For node g in Figure 5 this amounts, as we will see, to the following acceptance condition: $(\neg c \wedge (a \vee b)) \vee (a \wedge b)$.

Secondly, we need to take care of those nodes which do not depend on other nodes, that is, nodes without incoming links. We will call these nodes *input nodes* and denote the input nodes of a graph G as $input(G)$. It is often useful to consider input nodes as parameters whose truth values can be chosen freely, with the aim to explore the consequences of a particular choice. Consequently, our translation will depend on the assignment of truth values to the input nodes.

Definition 4.1 *Let $G = (N, E, I)$ be a labelled graph with nodes N, edges E and (integer) labelling function $I : E \to \mathbb{Z}$. Let $A \subseteq input(G)$ be the subset of input nodes considered true (the other input nodes are considered false). The sum-of-weights translation of G under A is the* ADF *$D = (S, L, C)$ with $S = N$, $L = E$, and the acceptance condition C_s (represented as a formula ϕ_s) is defined as follows:*

$$\phi_s = \begin{cases} \top, & \text{if } s \in A \\ \bot, & \text{if } s \in input(G) \setminus A \\ \phi_{sow}(s), & \text{otherwise} \end{cases}$$

where the formula $\phi_{sow}(s)$ is the disjunction of all conjunctions of literals built from parent nodes of s which represent truth value assignments under which the sum of weights of active links is positive.

Let us check how the acceptance condition for node g in Figure 5 is obtained. The following table shows 8 possible assignments of truth values to g's parent nodes, together with the sum of values of active links:

a	b	c	
t	**t**	**t**	1
t	**t**	**f**	7
t	**f**	**t**	-1
t	**f**	**f**	5
f	**t**	**t**	-4
f	**t**	**f**	2
f	**f**	**t**	-6
f	**f**	**f**	0

The sum of weights of active links is positive in 4 of the 8 lines, the acceptance condition of g is the disjunction of the conjunctions corresponding to these lines, that is:

$$(a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge \neg c)$$

which can be simplified to $(\neg c \wedge (a \vee b)) \vee (a \wedge b)$, the formula presented earlier.

Of course, there are many more strategies how to evaluate the numbers. One possibility is to check whether the maximal positive weight of an active link is higher than the maximal negative weight of an active link. This leads to a different definition of acceptance conditions for non-input nodes. We leave the details to the reader and just mention that in Figure 5 the acceptance condition for g under this new strategy becomes $(\neg c \wedge (a \vee b))$.

Qualitative preferences can be handled in a similar manner. Let us first introduce prioritized argument graphs.

Definition 4.2 *A prioritized argument graph is a tuple $G = (S, L^+, L^-, >)$ where S is the set of nodes, L^+ and L^- are subsets of $S \times S$, the supporting and attacking links, and $>$ is a strict partial order (irreflexive, transitive, antisymmetric) on S representing preferences among the nodes.*

As before, we will translate prioritized argument graphs to ADFs. We illustrate the translation using an example. Assume we are given the graph in Figure 6.

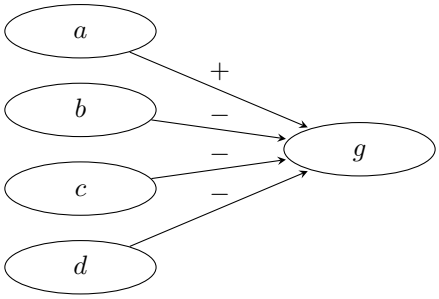


Figure 6: An argument graph with qualitative weights.

Assume further the preference ordering is $a > c$ and $g > d$, that is a is strictly preferred to c, g to d. We want to capture the following intuition: an attacker (represented by label $-$ in the graph) does not succeed if the attacked node is more preferred than the attacker, or if there is a more preferred supporting node (represented by label $+$ in the graph).

We treat input nodes as in Definition 4.1. The general scheme for deriving formulas expressing the corresponding acceptance condition ϕ_s for a node s with a non-empty set of parents is the following: we create a conjunction of implications, one for each attacker t of s which is not less preferred than s. The left side of the implication (the precondition) consists of the attacker t, the right side (conclusion) is the disjunction of all supporting nodes of s which are more preferred than t.

In the example above, the only attackers which are not less preferred than g are b and c. For b we obtain the implication $b \rightarrow \mathbf{f}$ (as there is no supporting node more preferred than b and the empty disjunction is equivalent to $\mathbf{f}$). For attacker c we obtain the implication $c \rightarrow a$, as a is more preferred than c. This yields the following acceptance condition for g: $(b \rightarrow \mathbf{f}) \wedge (c \rightarrow a)$ or, equivalently $\neg b \wedge (c \rightarrow a)$.

As a matter of fact, preferences are often not given in advance, as assumed in the example, but an issue of debate themselves. One way to model situations where the preference relation $>$ is established dynamically in the course of argumentation is the following. Let us assume some nodes represent (possibly conflicting) preference information, that is information about which pairs of nodes belong to $>$. The idea is to guess a (stable, preferred, grounded) interpretation M and then to verify whether M can be generated in a way satisfying the preference relation it contains. To do so we extract the preference information from the relevant nodes in M. We then check whether M can be reconstructed under this (now static) preference information using the techniques described above. We thus verify whether the preferences represented in the model itself were taken into account adequately.

Definition 4.3 *An argument graph with dynamic preferences is a tuple*

$$G = (S, L^+, L^-, P)$$

where S is the set of nodes, L^+ and L^- are subsets of $S \times S$, the supporting and attacking links, and $P : S \to S \times S$ is a partial function.

The function P assigns preference information to some of the nodes in S. If $P(a) = (b, c)$ then node a carries the information that b is preferred over c. For a three-valued interpretation M we use $>_M$ to denote the smallest strict partial order on S containing the set $\{(b, c) \mid P(a) = (b, c), M(a) = \mathbf{t}\}$. Note that $>_M$ may be undefined, e.g. if M contains two nodes with conflicting preference information. The semantics of argument graphs with dynamic preferences is now defined as follows:

Definition 4.4 *Let $G = (S, L^+, L^-, P)$ be an argument graph with dynamic preferences, A a subset of its input nodes. E is a (stable, preferred, grounded) interpretation of G under A iff $>_E$ is a strict partial order and E is a (stable, preferred, grounded) interpretation of the prioritized argument graph $D_E = (S, L^+, L^-, >_E)$ under A.*

We thus guess an interpretation E of the intended type, extract from E the corresponding strict partial order on S, and check whether E is among the intended interpretations of the (non-dynamic) prioritized argument graph which is based on the extracted preference information. The evaluation of the prioritized graph is based on the translation to ADFs described earlier in this section. For further details see [Brewka *et al.*, 2013].

4.2 Proof Standards

Proof standards are well known and play an important role in legal reasoning. They are based on the intuitive idea that decisions or verdicts which have drastic consequences, say for a defendant, should be based on stronger, less doubtful criteria than decisions with limited consequences, say a small fine. Farley and Freeman [Farley and Freeman, 1995] introduced a model of legal argumentation which distinguishes four types of arguments (in decreasing order of strength):

- *valid* arguments based on deductive inference,
- *strong* arguments based on inference with defeasible rules,
- *credible* arguments where premises give some evidence,
- *weak* arguments based on abductive reasoning.

By using values $V = \{+v, +s, +c, +w, -v, -s, -c, -w\}$ we will distinguish pro and con links of the corresponding types in argument graphs, where the type of a link is inherited from the type of its source node.

Based on these argument types, Farley and Freeman define the following proof standards:

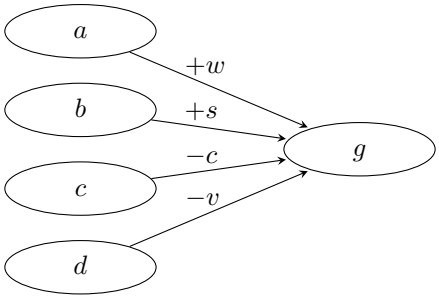


Figure 7: A Farley/Freeman argument graph.

- *Scintilla of Evidence*: at least one pro-argument is accepted.
- *Preponderance of Evidence*: at least one pro-argument is accepted, all accepted con arguments are outweighed by stronger accepted pro arguments.
- *Dialectical Validity*: there is at least one credible accepted pro-argument, none of the other side's arguments is accepted.
- *Beyond Reasonable Doubt*: there is at least one strong accepted pro-argument, none of the other side's arguments is accepted.
- *Beyond Doubt*: there is at least one valid active pro-argument, none of the other side's arguments is accepted.

Again we will show how these notions can be formalized using ADFs.

Consider the labelled graph in Figure 7. Let us focus on the acceptance condition for g, represented as a propositional formula. The condition obviously depends on g's proof standard. For scintilla of evidence it is sufficient that at least one pro-argument is accepted. There are two such arguments, a and b, the acceptance condition thus is $a \vee b$. For preponderance of evidence at least one pro-argument must be accepted, and in addition each accepted con-argument must be outweighed by a stronger pro-argument. In our case this means that if c is accepted, then the stronger pro-argument b must also be accepted, and d cannot be accepted, as there is no stronger pro-argument than the valid argument d. Taken together this yields the formula $(a \vee b) \wedge (c \rightarrow b) \wedge \neg d$. In a similar manner we obtain the formulas for g for the remaining proof standards, as shown in the following table:

Scintilla of evidence:	$a \vee b$
Preponderance of evidence:	$(a \vee b) \wedge (c \rightarrow b) \wedge \neg d$
Dialectical validity:	$b \wedge \neg c \wedge \neg d$
Beyond reasonable doubt:	$b \wedge \neg c \wedge \neg d$
Beyond doubt:	$\bot$

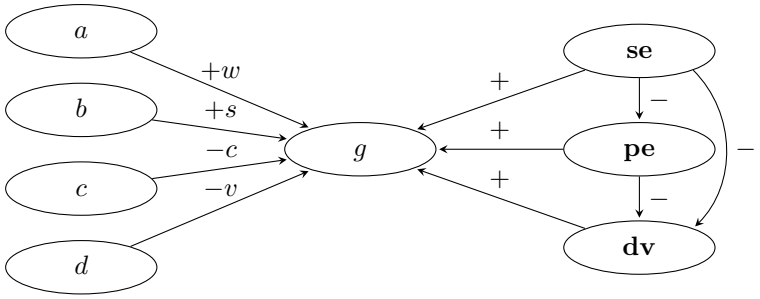


Figure 8: A graph with dynamic proof standards.

It is even possible to choose the proof standard dynamically. For ease of presentation let's focus on three proof standards, namely scintilla of evidence, preponderance of evidence and dialectical validity, represented as **se**, **pe** and **dv**, respectively.[9] Consider the graph in Figure 8 which should be viewed as part of a larger argument graph. The idea here is that scintilla of evidence is the default proof standard. If the corresponding node **se** is attacked from outside (e.g. since a crime was committed), then preponderance of evidence becomes the active proof standard. If also the corresponding node **pe** is attacked from outside (e.g. since the crime has serious consequences), then dialectical validity will be active. To model this intuition, the acceptance condition of node g becomes:

$$(\mathbf{se} \wedge (a \vee b)) \vee (\mathbf{pe} \wedge (a \vee b) \wedge (c \rightarrow b) \wedge \neg d) \vee (\mathbf{dv} \wedge b \wedge \neg c \wedge \neg d).$$

4.3 Carneades

Carneades [Gordon *et al.*, 2007] is an advanced model of argumentation based on a graphical representation of arguments and the propositions involved in them. Each proposition has an associated proof standard (scintilla of evidence, preponderance of evidence, clear and convincing evidence, beyond reasonable doubt, dialectical validity). There is some paraconsistency at work in the system as scintilla of evidence allows both a proposition and its negation to be accepted at the same time. The ADF graphs we will construct later will for this reason have separate nodes for each proposition p and its complement $\overline{p}$. A major restriction of Carneades is that cycles in the graph are not allowed (which means the system handles only cases where all Dung semantics coincide).

Let us start with some basic definitions underlying Carneades. Our presentation follows [Brewka and Gordon, 2010].

Definition 4.5 *An* argument *is a tuple $\langle P, E, c\rangle$ with premises P, exceptions E ($P \cap E = \emptyset$) and conclusion c. c and elements of P, E are literals.*

[9] The type of these nodes is irrelevant and thus left out.

An argument evaluation structure *(CAES) is a tuple* $\mathcal{S} = \langle \mathit{args}, \mathit{as}, \mathit{weight}, \mathit{standard} \rangle$*, where*

- *args is a set of arguments generating an acyclic argument graph,*
- *as is a consistent set of literals,*
- *weight assigns a real number to each argument, and*
- *standard maps propositions to a proof standard.*

The argument graph generated by a CAES is obtained as follows: each literal occurring in an argument *arg* becomes a node; each argument *arg* becomes a node; each premise of an argument *arg* is linked to the corresponding argument node *arg* via a link labelled with +, each exception via a link labelled with -; an additional link, labelled with $\mathit{weight}(\mathit{arg})$, connects *arg* and the conclusion of *arg*.

The central notions in Carneades are *applicability* of arguments and *acceptability* of propositions. These notions are defined via mutual recursion. Note that for the recursion to bottom out it is essential that Carneades is acyclic.

Definition 4.6 *We say an argument* $\langle P, E, c \rangle \in \mathit{args}$ *is* applicable *in* $\mathcal{S}$ *iff*

- $p \in P$ *implies* $p \in \mathit{as}$ *or* $[\overline{p} \notin \mathit{as}$ *and* p *acceptable in* $\mathcal{S}]$*, and*
- $p \in E$ *implies* $p \notin \mathit{as}$ *and* $[\overline{p} \in \mathit{as}$ *or* p *is not acceptable in* $\mathcal{S}]$*.*

Based on the applicability of arguments, we can define what it means for a proposition p to be *acceptable* in $\mathcal{S}$. As expected, acceptability depends on p's proof standard. The Carneades proof standards differ form those of Farley and Freeman. In particular, they depend on numerical values:

- $\mathit{standard}(p) = \mathit{se}$: there is an applicable argument for p,
- $\mathit{standard}(p) = \mathit{pe}$: p satisfies se, and the maximum weight assigned to an applicable argument pro p is greater than the maximum weight of an applicable argument con p,
- $\mathit{standard}(p) = \mathit{ce}$: p satisfies pe, and the maximum weight of an applicable pro argument exceeds a threshold α, and difference between the maximum weight of applicable pro arguments and the maximum weight of applicable con arguments exceeds a threshold β,
- $\mathit{standard}(p) = \mathit{bd}$: p satisfies ce, and the maximum weight of the applicable con arguments is less than a threshold γ,
- $\mathit{standard}(p) = \mathit{dv}$: there is an applicable argument pro p and no applicable argument con p.

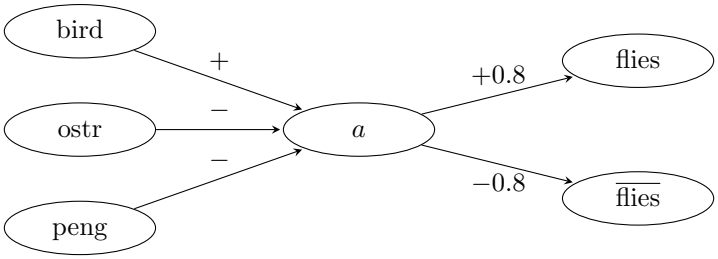


Figure 9: A Carneades argument represented graphically.

We now show how arguments and the generated argument graphs are represented using ADFs. The translation to ADFs is based on the techniques we have seen so far in this section. Consider the argument $a = \langle\{\text{bird}\}, \{\text{peng}, \text{ostr}\}, \text{flies}\rangle$ with $\mathit{weight}(a) = 0.8$. This argument is represented graphically as shown in Figure 9.

Apart from the duplication of propositions/complements the graphical representation corresponds to the original Carneades graph. Using techniques similar to the ones described earlier, we can properly define acceptance conditions such that an argument node is **t** in the ADF graph iff the argument is applicable, and a proposition node is **t** iff the proposition is acceptable. The acceptance condition of an argument node *arg* requires that all premises of *arg* are true, all exceptions false (assumptions can be handled by an easy preprocessing step). The acceptance condition of a proposition node depends on the proof standard and is modelled along the lines of what we have discussed earlier in this section. We leave the details to the reader. Note that we will resume our discussion of Carneades at the end of Section 5 where we show how the relevant acceptance conditions can be formalized in GRAPPA.

What has been gained by this reconstruction? Why is it useful? First of all, it shows the generality of ADFs. Secondly, it puts Carneades on safe formal ground. But in addition, and this is probably the main advantage, it allows us to give up the restriction of Carneades to acyclic argument graphs. Nothing in our translation rests on the assumption that Carneades is acyclic. The translation works perfectly well also for cyclic argument evaluation structures. The only difference is that the resulting ADF graph will have cycles as well. But handling cycles of this kind is part of the core functionality of ADFs, and they have a variety of different semantics to offer for this case, as we have seen in Section 3.

4.4 Rule-based Languages

A major strand of research in formal argumentation is concerned with using argumentation techniques to assign semantics to simple rule-based languages (see Chapter 6 of this handbook). Those languages are simple logic-inspired form-

alisms working with inference rules on a set of propositional literals. Inference rules can be strict, in which case the conclusion of the inference (a literal) must necessarily hold whenever all antecedents (also literals) hold. Inference rules can also be defeasible, which means that the conclusion *usually* holds whenever the antecedents hold. Here, the word "usually" suggests that there could be exceptional cases where a defeasible rule has not been applied [Pollock, 1987] (for example to avoid an imminent inconsistency).

Most of the existing works in this area translate rule-based languages to AFs by constructing arguments and identifying attacks. But this approach is not always without problems, as Caminada and Amgoud [Caminada and Amgoud, 2007] observed. (They even devised a set of *rationality postulates* for capturing the intended behavior of semantics for rule-based languages.) While there exist AF-based solutions to those problems [Wyner *et al.*, 2013], we concentrate here on one approach using ADFs as target language [Strass, 2013b; Strass, 2015b]. Translating to ADFs instead of AFs has the additional benefit of tackling the problem of cyclic justifications amongst arguments on the semantic level instead of the syntactic one (like it is done in the ASPIC approach [Caminada and Amgoud, 2007] among others). We only give intuitions here and refer the reader to the original paper(s) for details [Strass, 2013b; Strass, 2015b].

Inspired by the approach of Wyner et al. [Wyner *et al.*, 2013], Strass [Strass, 2013b; Strass, 2015b] directly uses the literals from the theory base as statements that express whether the literal holds. He also uses rule names as statements indicating that the rule is applicable. Additionally, for each rule r he creates a statement $\text{-}r$ indicating that the rule has not been applied. Not applying a rule is acceptable for defeasible rules, but unacceptable for strict rules since it would violate the closure postulate. This is enforced via integrity constraints saying that it may not be the case in any model that the rule body holds but the head does not hold: Technically, for a strict rule r, he introduces a conditional self-attack of $\text{-}r$; this self-attack becomes active if (and only if) the body of r is satisfied but the head of r is not satisfied, thereby preventing this undesirable state of affairs from getting included in a model. Defeasible rules offer some degree of choice, whence it is left to the semantics whether or not to apply them. This choice is modelled by a mutual attack cycle between r and $\text{-}r$. The remaining acceptance conditions are equally straightforward:

- Opposite literals attack each other.
- A literal is accepted whenever some rule deriving it is applicable, that is, all rules with head ψ support statement ψ.
- A strict rule is applicable whenever all of its body literals hold, that is, the body literals of r are exactly the supporters of r.
- Likewise, a defeasible rule is applicable whenever all of its body literals hold, and additionally the negation of its head literal must not hold.

Strass [2013b, 2015b] showed that the approach satisfies the rationality postulates of Caminada and Amgoud [2007]. Furthermore, this method has a mild computational complexity (with an at most quadratic blowup from rule-based theory to ADF formalization, while there can be exponential to infinite blowup in other approaches).

5 Graph-based Argument Processing

We have seen in Section 4 how ADFs can be used to provide graphical representations of argumentation scenarios with semantics. The different approaches were based on translations from some graphical representation to ADFs. In a nutshell, the GRAPPA approach [Brewka and Woltran, 2014] described in this section addresses the opposite question: is it possible to extend the formal techniques underlying ADFs in such a way that the semantics of various kinds of graphical representations can be defined directly for these representations, without the detour of a translation? More specifically, we will consider arbitrary (edge) labelled graphs. Such graphs are highly popular for visualizing argumentation scenarios, and indeed this chapter (and the handbook) is full of such representations. The goal of this section is to define various semantics directly for such labelled graphs.

Another way of looking at the approach is the following: Dung AFs actually can be seen as graphs where all edges have the same label, which is left implicit for this reason. In addition, all nodes have the same type of acceptance condition. Dung's seminal contribution can thus be characterized as defining various semantics for specific graphs with a single label and uniform acceptance conditions. Our goal is to generalize this to arbitrary labelled graphs with flexible, user-defined acceptance conditions.

GRAPPA requires two major changes. First of all, the acceptance conditions can no longer be propositional formulas built from parent nodes, as in ADFs. We rather have to define them in terms of the labels of active links in the graph, that is links whose source nodes are accepted (true). More precisely, since it may be relevant whether there are multiple active links with the same label, we have to consider multisets of labels. An acceptance condition will thus be a function assigning a truth value to each multiset of labels. Secondly, we have to modify the operator Γ_D for ADFs D as defined in Section 3 in such a way that the new acceptance conditions are taken into account adequately.

In the following we describe multisets as functions into the natural numbers. Intuitively, the number assigned to an element describes the number of occurrences of the element in the multiset.

Definition 5.1 *An acceptance function over a set of labels L is a function $c : (L \to \mathbb{N}) \to \{\mathbf{t}, \mathbf{f}\}$.*

The set of all acceptance functions over L is denoted F^L.

Definition 5.2 *A labelled argument graph (LAG) is a tuple $G = (S, E, L, \lambda, \alpha)$*

where

- S *is a set of nodes (statements),*
- E *is a set of edges (dependencies),*
- L *is a set of labels,*
- $\lambda : E \to L$ *assigns labels to edges,*
- $\alpha : S \to F^L$ *assigns* L*-acceptance-functions to nodes.*

The characteristic operator Γ_G of a LAG G basically does what the corresponding operator does for ADFs: it takes a three-valued (or, equivalently, partial) interpretation v and produces a new one v'. In doing so, it checks which truth values of nodes in S can be justified by v. This is done by considering all possible completions of v, more precisely the multisets of active labels induced by completions of v. These multisets are obtained by including an occurrence of a particular label for each occurrence of that label in a link which is active in the completion. If the acceptance function of s yields $\mathbf{t}$ under all completions (more precisely, for all multisets induced by any completion), then v' assigns $\mathbf{t}$ to s. If the acceptance function of s yields $\mathbf{f}$ under all completions, then v' assigns $\mathbf{f}$ to s. In all other cases the value remains undefined.

Here are the formal details. Note that we represent here three-valued interpretations v as sets of literals: nodes true in v appear positively in the set, nodes assigned false appear negated, and undefined nodes are left out.

Definition 5.3 *Let* $G = (S, E, L, \lambda, \alpha)$ *be a LAG,* v *a three-valued interpretation of* S*.* m_s^v*, the multiset of active labels of* $s \in S$ *in* G *under* v*, is defined as*

$$m_s^v(l) = |\{(e, s) \in E \mid e \in v, \lambda((e, s)) = l\}|$$

for each $l \in L$*.*

The characteristic operator Γ_G of G takes a three-valued interpretation v of S and produces a revised three-valued interpretation $\Gamma_G(v)$ of S.

Definition 5.4 *Let* $G = (S, E, L, \lambda, \alpha)$ *be a LAG,* v *a three-valued interpretation of* S*.* $\Gamma_G(v) = P_G(v) \cup N_G(v)$ *with*

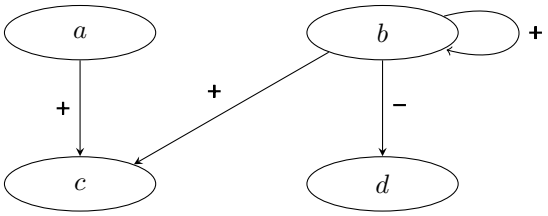
$$P_G(v) = \left\{ s \;\middle|\; \alpha(s)(m) = \mathbf{t} \text{ for each } m \in \{m_s^{v'} \mid v' \in [v]_c\} \right\}$$
$$N_G(v) = \left\{ \neg s \;\middle|\; \alpha(s)(m) = \mathbf{f} \text{ for each } m \in \{m_s^{v'} \mid v' \in [v]_c\} \right\}$$

With this new operator we can define the semantics of GRAPPA in exactly the same way as was done for ADFs:

Definition 5.5 *Let $G = (S, E, L, \lambda, \alpha)$ be a LAG, v a three-valued interpretation of S.*

- *v is a model of G iff v is total and $v = \Gamma_G(v)$,*
- *v is grounded in G iff v is the least fixed point of Γ_G,*
- *v is admissible in G iff $v \subseteq \Gamma_G(v)$,*
- *v is preferred in G iff v is $\subseteq$-maximal admissible in G,*
- *v is complete in G iff $v = \Gamma_G(v)$.*

Example 5.6 *This is a variation of Example 3.2. Consider the LAG with $S = \{a, b, c, d\}$ and $L = \{+, -\}$. The following graph shows the labels of each link.*



For simplicity, let us assume all nodes have the same acceptance condition requiring that all positive links must be active (that is the respective parents must be t) and no negative link is active.[10] *We obtain two models, namely $v_1 = \{a, b, c, \neg d\}$ and $v_2 = \{a, \neg b, \neg c, d\}$. The grounded interpretation is $v_3 = \{a\}$. The 16 admissible interpretations are exactly the same as for Example 3.9. Among the admissible interpretations $\{a, b, c, \neg d\}$ and $\{a, \neg b, \neg c, d\}$ are preferred. Complete interpretations are these two and in addition $\{a\}$.*

Now let us turn to stable semantics. The idea underlying stable semantics is to exclude self-justifying cycles. Again this semantics can be defined along the lines of the corresponding definition for ADFs in [Brewka *et al.*, 2013]: take a model v, reduce the LAG based on v and check whether the grounded extension of the reduced LAG coincides with the nodes true in v. Here is the definition:

Definition 5.7 *Let $G = (S, E, L, \lambda, \alpha)$ be a LAG, v a model of G, $S^v = v \cap S$. v is a stable model of G iff v restricted to S^v is the grounded interpretation of $G^v = (S^v, E^v, L, \lambda^v, \alpha^v)$, the v-reduct of G, where*

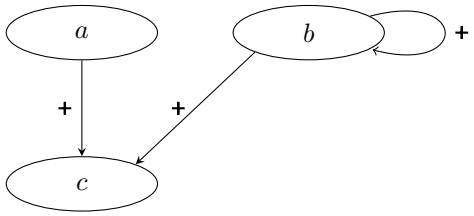
- *$E^v = E \cap (S^v \times S^v)$,*

[10] In the pattern language developed later in this section this can be expressed as $(\#_t(+) - \#(+) = 0) \wedge (\#(-) = 0)$.

- λ^v *is* λ *restricted to* S^v,[11]
- α^v *is* α *restricted to* S^v.

Observe that in α^v we did not have to alter the values of the function, i.e. the true and false multisets remain the same (although some of them might become "unused" since the number of parents shrinked). We will see later that this exactly matches the stable semantics for ADFs from [Brewka *et al.*, 2013]. For the moment, we continue our running example.

Example 5.8 *For Example 5.6 we obtained two models,* $v_1 = \{a, b, c, \neg d\}$ *and* $v_2 = \{a, \neg b, \neg c, d\}$. *In* v_1 *the justification for* b *is obviously based on a cycle. The* v_1*-reduct of our graph is*



It is easy to see that the grounded interpretation of the reduced graph is $\{a\}$, v_1 *is thus not a stable model, as intended. We leave it to the reader to verify that* v_2 *indeed is a stable model.*

Results about the semantics carry over from ADFs [Brewka *et al.*, 2013].

Proposition 5.9 *Let* G *be a LAG. The following inclusions hold:*

$$stb(G) \subseteq mod(G) \subseteq pref(G) \subseteq com(G) \subseteq adm(G),$$

where $stb(G)$,$mod(G)$, $pref(G)$, $com(G)$ *and* $adm(G)$ *denote the sets of stable models, models, preferred interpretations, complete interpretations and admissible interpretations of* G, *respectively. Moreover,* $pref(G) \neq \emptyset$, *whereas* $mod(G') = \emptyset$ *for some LAG* G'.

A remaining question is how to actually specify acceptance functions for GRAPPA. In [Brewka and Woltran, 2014] a specific pattern language has been developed for this purpose. This pattern language allows for the specification of conditions on multisets of labels. In the patterns one can refer to the number of total and active labels of specific types, to minimal/maximal numerical labels of active links. It is also possible to use simple arithmetics and relations.

More precisely, GRAPPA acceptance functions are specified using *acceptance patterns* over a set of labels L defined as follows:

[11] Given a function $f : M \to N$ and $M' \subseteq M$, f restricted to M' is the function $f' : M' \to N$ such that $f'(m) = f(m)$ for all $m \in M'$.

- A *term* over L is of the form $\#(l)$, $\#_t(l)$ (with $l \in L$), or min, min_t, max, max_t, sum, sum_t, $count$, $count_t$.

- A *basic acceptance pattern* (over L) is of the form $a_1t_1 + \cdots + a_nt_n \, R \, a$, where the t_i are terms over L, the a_is and a are integers and $R \in \{<, \leq, =, \neq, \geq, >\}$.

- An *acceptance pattern* (over L) is a basic acceptance pattern or a Boolean combination of acceptance patterns.

A GRAPPA instance then is a labelled argument graph with acceptance functions represented as acceptance patterns:

Definition 5.10 *A* GRAPPA instance *is a tuple $G = (S, E, L, \lambda, \pi)$ where S is a set of statements, E a set of edges, L a set of labels, λ an assignment of labels to edges, and π an assignment of acceptance patterns over L to all elements of S.*

We still need to specify what the acceptance function represented by a particular pattern assigned to a node s is. Recall that an acceptance function assigns a truth value in $\{\mathbf{t}, \mathbf{f}\}$ to a multiset of labels. We will define this function by specifying a satisfaction relation $\models$ between multisets and patterns: the basic idea is that a multiset receives value $\mathbf{t}$ iff it satisfies the corresponding pattern. The actual definition is slightly more complicated, though, as some of the terms (actually those indexed with t) are actually independent of the multiset, but depend on the node s, more precisely on the labels of links – active or not – with target s. For this reason, satisfaction of a pattern depends on both a multiset of labels and the node the pattern is assigned to via π. For a multiset of labels $m : L \to \mathbb{N}$ and $s \in S$ the value function val_s^m is:

$$
\begin{array}{ll}
val_s^m(\#l) & = m(l) \\
val_s^m(\#_t l) & = |\{(e,s) \in E \mid \lambda((e,s)) = l\}| \\
val_s^m(min) & = \mathbf{min}\{l \in L \mid m(l) > 0\} \\
val_s^m(min_t) & = \mathbf{min}\{\lambda((e,s)) \mid (e,s) \in E\} \\
val_s^m(max) & = \mathbf{max}\{l \in L \mid m(l) > 0\} \\
val_s^m(max_t) & = \mathbf{max}\{\lambda((e,s)) \mid (e,s) \in E\} \\
val_s^m(sum) & = \sum_{l \in L} m(l) \\
val_s^m(sum_t) & = \sum_{(e,s) \in E} \lambda((e,s)) \\
val_s^m(count) & = |\{l \mid m(l) > 0\}| \\
val_s^m(count_t) & = |\{\lambda((e,s)) \mid (e,s) \in E\}|
\end{array}
$$

$min_{(t)}$, $max_{(t)}$, $sum_{(t)}$ are undefined in case of non-numerical labels. For $\emptyset$ they yield the neutral element of the corresponding operation, i.e.

$$val_s^m(sum) = val_s^m(sum_t) = 0,$$
$$val_s^m(min) = val_s^m(min_t) = \infty,$$
$$val_s^m(max) = val_s^m(max_t) = -\infty.$$

Let m and s be as before. For basic acceptance patterns the *satisfaction relation* $\models$ is defined by

$$(m, s) \models a_1 t_1 + \cdots + a_n t_n R\, a \quad \text{iff} \quad \sum_{i=1}^{n} \big(a_i \; val_s^m(t_i)\big) \; R \; a.$$

The extension to Boolean combinations is as usual. The acceptance function represented by pattern p at node s then is the function assigning $\mathbf{t}$ to multiset m iff $(m, s) \models p$.

Example 5.11 *Let $L = \{\mathtt{++}, \mathtt{+}, \mathtt{-}, \mathtt{--}\}$ be a set of labels representing strong support, support, attack and strong attack, respectively. Assume a node s is accepted if its (active) support is stronger than its attack, where we measure strength by counting the respective links, hereby multiplying strong support/attack with a factor of 2. This can be specified using the following pattern for s:*

$$2(\#\mathtt{++}) + (\#\mathtt{+}) - 2(\#\mathtt{--}) - (\#\mathtt{-}) > 0.$$

We conclude this section by showing how the necessary patterns for Carneades argument graphs, which we discussed in Section 4.3, can be defined in GRAPPA. Recall that these graphs have two kinds of nodes, argument nodes and propositions nodes. The pattern for all argument nodes is

$$\big((\#_t\mathtt{+}) - (\#\mathtt{+}) = 0\big) \wedge \big((\#\mathtt{-}) = 0\big).$$

which says that all premises and none of the exceptions must be accepted. The patterns for proposition nodes depend on their proof standard. Recall that some of these standards have additional numerical parameters α, β and γ. The terms max and min represent the maximal, respectively minimal, label of an active link:

- scintilla of evidence: $\max > 0$
- preponderance of evidence: $\max + \min > 0$
- clear and convincing evidence: $(\max > \alpha) \wedge (\max + \min > \beta)$
- beyond reasonable doubt: $(\max > \alpha) \wedge (\max + \min > \beta) \wedge (-\min < \gamma)$
- dialectical validity: $(\max > 0) \wedge (\min > 0)$

This representation of the acceptance conditions underlying Carneades is not only extremely simple. It has the big advantage that it is uniform: the patterns for all nodes with the same proof standard are actually the same. This is different from representations of proof standards and other notions we discussed in Section 4 in ADFs where the acceptance condition for each node depends on its specific parents.

6 Computational Aspects

In the introduction we discussed, in an informal manner, relationships between statements (arguments) that are supporting or attacking, in the sense that a statement can have a positive or negative influence on the acceptance of another statement. General ADFs have a generic notion of links (dependencies) between statements. However, such links can be formally categorized into 4 groups, depending on whether they have an attacking or supporting nature (or both or neither). This leads to the notion of so-called bipolar ADFs (BADFs for short) which contain only attacking or supporting dependencies. We will introduce them, based on the original definition of [Brewka and Woltran, 2010], in Section 6.1, together with the formalization of attacking and supporting links. Such BADFs are a subclass of general ADFs, yet have appealing computational properties. They generalize AFs in a direct manner, but are strictly "in-between" AFs and general ADFs w.r.t. their corresponding expressiveness. Results relating to expressiveness are presented in Section 6.2. Further, many frameworks arising in argumentation in AI, other than AFs, can be translated to BADFs [Polberg, 2016] (partially under semantics not discussed in this chapter).

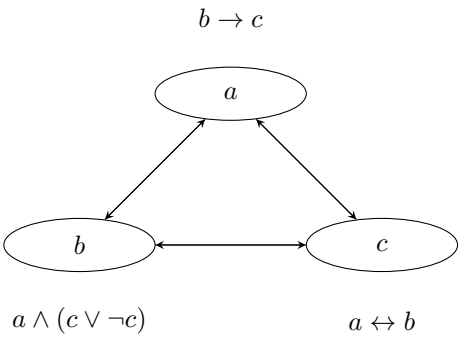
From a computational perspective, BADFs have the following interesting properties: they have the same worst-time complexity as AFs for many semantics, while general ADFs typically exhibit higher computational complexity. We summarize these results in Section 6.3, followed by Section 6.4 that gives pointers to recent systems for computing reasoning tasks on ADFs and BADFs.

6.1 Bipolar ADFs

As we have seen in previous sections, the concept of acceptance condition is quite powerful. A natural question is to what extent different restrictions of acceptance conditions may form interesting subclasses of ADFs. One such subclass are bipolar ADFs, as already defined in [Brewka and Woltran, 2010]. This class relies on the concept of attacking and supporting links which are defined as follows.

Let $D = (S, L, C)$ be an ADF. Formally, a link $(r, s) \in L$ is

- supporting in D iff for all $R \subseteq par(s)$, we have $C_s(R) = \mathbf{t}$ implies $C_s(R \cup \{r\}) = \mathbf{t}$;
- attacking in D iff for all $R \subseteq par(s)$, we have $C_s(R \cup \{r\}) = \mathbf{t}$ implies $C_s(R) = \mathbf{t}$.



L^+	L^-
(a, b)	(b, a)
(c, b)	(c, b)
(c, a)	

Figure 10: An ADF *with link types.*

We use $L^+ \subseteq L$ to denote all supporting and $L^- \subseteq L$ to denote all attacking links of L in an ADF $D = (S, L, C)$.

Example 6.1 *In Figure 10 we see an example* ADF $D = (S, L, C)$ *with* $S = \{a, b, c\}$ *and acceptance conditions* $\varphi_a = b \to c$, $\varphi_b = a \wedge (c \vee \neg c)$, *and* $\varphi_c = a \leftrightarrow b$. *On the right of that figure the link types are shown. Let us investigate why some of the links are supporting or attacking. Looking at the acceptance condition of* a, φ_a, *and the parents of* a *then we have the following relevant sets of statements (shown as two-valued interpretations):*

$$\begin{array}{rrcl} v_1 \mathrel{\hat{=}} & \{\neg b, \neg c\} & \models & \varphi_a \\ v_2 \mathrel{\hat{=}} & \{b, \neg c\} & \not\models & \varphi_a \\ v_3 \mathrel{\hat{=}} & \{\neg b, c\} & \models & \varphi_a \\ v_4 \mathrel{\hat{=}} & \{b, c\} & \models & \varphi_a \end{array}$$

We see, e.g., that the link (c, a) *is supporting, because whenever* c *is added to a subset of parents that is mapped to* $\mathbf{t}$ *by* C_a *(switched to true in every model of* φ_a*) then the new set (interpretation) is again mapped to true by acceptance condition* C_a *(is a model of* φ_a*). More concretely,* v_1, v_3, *and* v_4 *are models of acceptance condition* φ_a. *Switching the truth value of* c *to true in each of them, results in* v_3 *and* v_4 *(assigning* c *to true in* v_1 *and* v_3 *results in both cases with* v_3, *and assigning* c *to true in* v_4 *is again equal to* v_4*). Both* v_3 *and* v_4 *are models of* φ_a. *This means* (c, a) *is a supporting link. Similarly, link* (b, a) *is attacking because whenever we remove* b *from a set of parents of* a *that is mapped to* $\mathbf{t}$ *by* C_a *we get a set that is likewise mapped to* $\mathbf{t}$ *by* C_a.

Links (a, b) which are *both* attacking and supporting are so-called redundant links. The reason to call such a link redundant is that switching the truth value of a in any interpretation does not change the evaluation of acceptance condition φ_b w.r.t. the original interpretation and the modified interpretation. A link that is *neither* attacking nor supporting is called dependent.

Example 6.2 *Continuing Example 6.1, the link* (c, b) *is a redundant link. This link is both attacking and supporting. Redundancy means that the evaluation of* φ_b *is independent of the value of* c *(formula* φ_b *only depends on the truth value of* a*). In contrast, the links* (b, c) *and* (a, c) *are dependent links. For instance,* $\{\neg a, \neg b\} \models \varphi_c$ *and* $\{a, \neg b\} \not\models \varphi_c$ *taken together show that* (a, c) *is not supporting in this* ADF*. To see that* (a, c) *is not attacking, consider* $\{a, b\} \models \varphi_c$ *and* $\{\neg a, b\} \not\models \varphi_c$*.*

An ADF $D = (S, L, C)$ is bipolar (a BADF) if all links in L are supporting or attacking or both, i.e., $L = L^+ \cup L^-$. For example, our running example ADF from Example 3.2 is a BADF. Further, for any AF F its associated ADF D_F is bipolar, in fact each link in D_F is attacking.

Bipolar ADFs are still a quite expressible class; they allow acceptance conditions not only to express simply attack and support (for example $\neg a_1 \wedge \cdots \wedge \neg a_n \wedge s_1 \wedge \cdots \wedge s_m$ expressing that a statement is attacked by statements a_i and supported by statements s_j), but more advanced relations, like e.g. $((\neg a_1 \vee s_1) \wedge (\neg a_2 \vee s_2)) \vee \neg a_3$; in fact, all examples given in Section 4 are also bipolar ADFs. We would like to mention here that bipolar ADFs behave differently than the prominent class of bipolar AFs [Cayrol and Lagasquie-Schiex, 2013]. Indeed, several concepts of support relations have been discussed in the literature (abstract, deductive, necessary, and evidential support), thus a detailed discussion is beyond the scope of this chapter, and we refer the reader to works relating ADFs to formalisms including support [Polberg and Oren, 2014; Polberg, 2016]. However, what is important to state is that bipolar ADFs treat support and attack as equally strong concepts. Given the generality of bipolar ADFs which allow to "mix" support and attack as exemplified above, a distinct handling of support and attack in ADFs, e.g. as separated concepts in the language instead of a property of links and acceptance conditions, would require a lot of additional machinery.

Acceptance conditions in BADFs are, in fact, not only interesting for defining ADFs. The study of the concept of *bipolar Boolean functions* has meanwhile found applications outside of ADFs. Baumann and Strass (2016) have analyzed the integer sequence that arises when considering for each positive integer n the number of bipolar Boolean functions in n arguments. The resulting sequence is novel and has been added to the Online Encyclopedia of Number Sequences[12]. In further related work, Alviano, Faber, and Strass [Alviano *et al.*, 2016] applied the concept of bipolar Boolean functions to aggregates in answer set programming and obtained a novel class of aggregates whose model checking problems (according to the semantics of Pelov et al. [Pelov *et al.*, 2007] and Son and Pontelli [Son and Pontelli, 2007]) can be decided in deterministic polynomial time. They even identify a class that goes beyond bipolar Boolean functions but still retains polynomial-time decidability; this might constitute an interesting avenue for research that extends the bipolarity concept of ADFs.

[12] `https://oeis.org/A245079`

6.2 Expressiveness and Realizability

Expressiveness of a formalism $\mathcal{F}$ (i.e. the set of structures available in a formalism) with a semantics σ over a vocabulary A can be defined as the set of interpretation-sets over A that elements of $\mathcal{F}$ (the knowledge bases $\mathsf{kb} \in \mathcal{F}$ of that formalism) can produce. Formally, the *signature* of a formalism $\mathcal{F}$ w.r.t. semantics σ is the set

$$\Sigma_{\mathcal{F}}^{\sigma} = \{\sigma(\mathsf{kb}) \mid \mathsf{kb} \in \mathcal{F}\}$$

Intuitively, expressiveness is a basic measure of the capabilities of formalism $\mathcal{F}$ under σ, because it characterizes what "can and cannot be done" with $\mathcal{F}$ under semantics σ [Gogic *et al.*, 1995]. Whenever we have two formalisms, say $\mathcal{F}_1$ and $\mathcal{F}_2$, that share a semantics σ and we find that $\Sigma_{\mathcal{F}_1}^{\sigma} \subsetneq \Sigma_{\mathcal{F}_2}^{\sigma}$, then this intuitively means that $\mathcal{F}_2$ is strictly more expressive than $\mathcal{F}_1$: all sets $V \subseteq \mathcal{V}_3$ that can be realized with $\mathcal{F}_1$ can be realized with $\mathcal{F}_2$, and there is at least one set $V \subseteq \mathcal{V}_3$ that can be realized with $\mathcal{F}_2$ but not with $\mathcal{F}_1$.

For AFs, BADFs and ADFs under various semantics, their relative expressiveness is summarized in the following result [Strass, 2015c; Strass, 2015a; Linsbichler *et al.*, 2016a].

Theorem 6.3 *For $\sigma \in \{adm, com, prf, mod\}$, we find that*

$$\Sigma_{AF}^{\sigma} \subsetneq \Sigma_{BADF}^{\sigma} \subsetneq \Sigma_{ADF}^{\sigma}.$$

For the stable model semantics stb, we find that

$$\Sigma_{AF}^{mod} = \Sigma_{AF}^{stb} \subsetneq \Sigma_{BADF}^{stb} = \Sigma_{ADF}^{stb}.$$

Furthermore, for the model semantics we have

$$\Sigma_{ADF}^{mod} = \mathcal{V}_2 = \{v : A \to \{\mathbf{t}, \mathbf{f}\}\},$$

that is, ADFs *under the model semantics are universally expressive.*

Example 6.4 *We give example sets of interpretations that can be used to witness $\Sigma_{AF}^{prf} \subsetneq \Sigma_{BADF}^{prf} \subsetneq \Sigma_{ADF}^{prf}$. Consider $S = \{a, b, c\}$ and interpretations $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{f}\}$, $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{t}\}$, and $v_3 = \{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}\}$. To see that $\{v_1, v_2, v_3\} \in \Sigma_{BADF}^{prf}$, consider the* ADF *over S with acceptance conditions $\varphi_a = \neg b \vee \neg c$, $\varphi_b = \neg a \vee \neg c$, and $\varphi_c = \neg a \vee \neg b$. It is easy to verify that this* ADF *is bipolar and that $\{v_1, v_2, v_3\}$ constitute its preferred interpretations. On the other hand, from results in [Dunne* et al.*, 2015] it follows that there is no* AF *with preferred extensions $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$. In fact, this is quite easy to see: consider there would exist an* AF *F with those three preferred extensions. Then, there cannot be an attack in F between a and b, and moreover $\{a, b\}$ defends itself in F; the same holds for the pairs a, c, and b, c. But then, $\{a, b, c\}$ has to be conflict-free in F and defends itself, and thus $\{a, b\}$ (and likewise, $\{a, c\}$ and $\{b, c\}$) cannot be preferred in F; a contradiction.*

For $\Sigma^{prf}_{BADF} \subsetneq \Sigma^{prf}_{ADF}$, we use an example given in [Linsbichler et al.*, 2016b, Theorem 8]: consider $S' = \{a, b\}$ and interpretations $v_4 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}\}$, $v_5 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}\}$, and $v_6 = \{a \mapsto \mathbf{f}, b \mapsto \mathbf{u}\}$. For $X' = \{v_4, v_5, v_6\}$ we have $X' \in \Sigma^{prf}_{ADF}$, but $X' \notin \Sigma^{prf}_{BADF}$. For general* ADF*s, one example* ADF *is $D' = (S', L', \{\varphi_a = a, \varphi_b = a \leftrightarrow b\})$. All three interpretations v_4, v_5, and v_6 are preferred interpretations of D'. This* ADF *D' is not bipolar (due to φ_b, see Example 6.2). There is no* BADF *that has X' exactly as its preferred interpretations.*[13]

While this shows that BADFs can do strictly more than AFs, and in turn ADFs can do strictly more than BADFs (with the exception of the stable model semantics), there is little information on *what exactly* these signatures look like. Work on precisely characterizing signatures has been carried out for AFs [Dunne *et al.*, 2015]; the results can be found in Chapter 17 of this handbook. There has also been work on characterizing realizability for ADFs under two-valued [Strass, 2015a] and three-valued [Pührer, 2015; Linsbichler *et al.*, 2016a] semantics.

Finally, initial results on characterizing the representational *succinctness* of these formalisms have recently been obtained. Succinctness not only takes into account *what* formalisms can realize, but also *to what representational cost*, that is, what amount of space is needed to represent the smallest knowledge base realizing some desired set of interpretations. Again, the capabilities of different formalisms can be compared with respect to this measure [Gogic *et al.*, 1995]. As one promising result on ADFs, it turned out that even BADFs are exponentially more succinct than normal logic programs [Strass, 2015a].

6.3 Computational Complexity

The computational complexity of ADFs is well-studied [Strass and Wallner, 2014; Strass and Wallner, 2015; Gaggl *et al.*, 2015; Brewka *et al.*, 2013; Polberg and Wallner, 2017; Wallner, 2014]; for an overview we refer the reader to Chapter 13 of this volume. For the reader's convenience we repeat here the main results. For a specified semantics σ, the main reasoning tasks for ADFs to solve are:

- Credulous acceptance of a statement: is statement s assigned to true in at least one interpretation under semantics σ?
- Skeptical acceptance of a statement: is statement s assigned to true in all interpretations under semantics σ?
- Interpretation verification: is a given interpretation an interpretation under semantics σ?
- Interpretation existence: is there an interpretation under semantics σ?

[13] For an automated way to check whether for a given set of three-valued interpretations there is an ADF, BADF, or AF that has exactly this set as its σ-interpretations, one can use the system UNREAL [Linsbichler *et al.*, 2016b], available at `http://www.dbai.tuwien.ac.at/proj/adf/unreal/`.

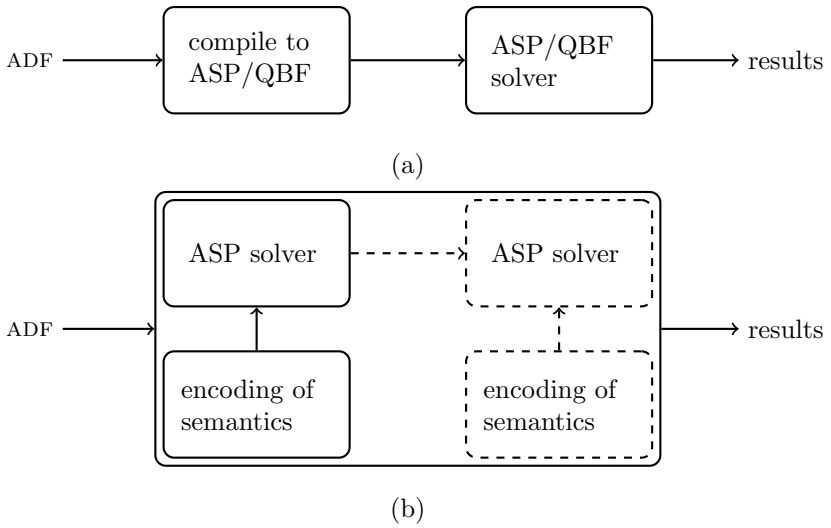


*Figure 11: Workflow for systems based on (a) instance-based compilation (*QADF*,* GrappaVis*, and* YADF*), and (b) static encodings (*DIAMOND *and* GrappaVis*)*

- Non-trivial interpretation existence: is there an interpretation under semantics σ assigning true or false to some statement?

Briefly put, complexity of reasoning tasks on general ADFs is situated one level higher in the polynomial hierarchy compared to the corresponding tasks on AFs. For BADFs complexity of reasoning stays at the same level as reasoning on AFs for most reasoning tasks, if the link type (attack or support) for each link is known (part of the input). Thus, BADFs offer more modeling capabilities than AFs while having the same (worst-case) computational cost as AFs for many reasoning tasks.

6.4 Systems

Systems for implementing reasoning on ADFs rely on declarative encodings in answer-set programming (ASP) [Brewka *et al.*, 2011] or quantified Boolean satisfiability, and utilize available solvers for these languages [Gebser *et al.*, 2011; Lonsing and Biere, 2010]. Most prominently, the DIAMOND family[14] [Strass and Ellmauthaler, 2017; Ellmauthaler and Strass, 2016; Ellmauthaler and Strass, 2014; Ellmauthaler and Strass, 2013] consists of ASP-based systems for reasoning on ADFs. In each DIAMOND version an ADF is encoded via ASP facts and, when augmented with static encodings for semantics, several reasoning tasks can be solved by computing answer-sets of the resulting ASP. Depending on the complexity of the reasoning task and used options in DIAMOND one call (in

[14] `http://diamond-adf.sourceforge.net/`

some family members two calls) to an ASP-solver are carried out to solve the given problem instance. DIAMOND includes dedicated BADF-specific encodings that make use of BADFs' upper complexity bounds.

The system QADF[15] [Diller *et al.*, 2015] uses solvers for quantified Boolean formulas (QBFs) to perform reasoning on ADFs. In QADF, in contrast to DIAMOND, each ADF instance is compiled to a QBF incorporating both the input ADF and the chosen semantics, i.e., the encodings for the semantics are not static.

GRAPPAVIS[16] [Heißenberger, 2016] is a system implementing GRAPPA (see Section 5) and incorporates both instance-based compilation of GRAPPA input into declarative ASP encodings and static encodings for the semantics utilizing in both cases one ASP solver call.

The system YADF[17] [Brewka *et al.*, 2017] is an ASP-based system for ADFs, based on the encodings for GRAPPA used in GRAPPAVIS. This system compiles ADF instances into one program to call an ASP solver (once).

The basic workflows for DIAMOND, QADF, GRAPPAVIS, and YADF are shown in Figure 11. With this figure we illustrate that QADF, GRAPPAVIS, and YADF implement algorithms that take an instance of an ADF, compile this instance, together with the chosen semantics and reasoning task, to one instance of an ASP or a QBF. On the other hand, DIAMOND and GRAPPAVIS implement algorithms that take an instance of an ADF, add to this instance a static encoding for the semantics and reasoning task, and give these to an ASP solver (with calling such a solver once or twice, depending on the task). The difference between (a) and (b) is that in (a) ADF and semantics have to be compiled together into one input for the solver, while for (b) semantics can be encoded separately (and modified separately).

A technique to cope with the high computational complexity of reasoning on ADFs was proposed by Linsbichler (2014). The technique is based on splitting the input ADF into partitions and solving one partition and transforming and solving the other partitions accordingly.

7 Conclusion

In this chapter, we have reviewed the argumentation formalism of abstract dialectical frameworks (ADFs). In contrast to Dung style frameworks, ADFs allow for a much more general specification of the interrelationship between the arguments. We have discussed how standard semantics like admissible, grounded, complete, preferred and stable can be generalized to ADFs by making use of the well known approximation fixpoint theory due to Denecker, Marek and Truszczyński [Denecker *et al.*, 2004].

Alternative approaches to defining ADF semantics can be found in the works of Polberg and colleagues [Polberg *et al.*, 2013; Polberg, 2014a; Polberg, 2014b;

[15] `http://www.dbai.tuwien.ac.at/proj/adf/qadf/`
[16] `http://www.dbai.tuwien.ac.at/proj/adf/grappavis/`
[17] `http://www.dbai.tuwien.ac.at/proj/adf/yadf/`

Polberg, 2015]. Likewise, further well-known semantics for AFs have been generalized to ADFs, e.g. naive, stage, and the cf2 family of semantics [Gaggl and Strass, 2014] and an alternative, symmetric version of the naive semantics [Strass and Wallner, 2015].

A further subclass of ADFs, related to a certain notion of acyclicity and different from BADFs, is investigated in [Polberg, 2015; Polberg, 2016]. Other authors have analyzed the relationship of ADFs and logic programs [Strass, 2013a; Alviano and Faber, 2015] and in the course of that have defined new ADF semantics, like approximate stable models [Strass, 2013a], F-stable models [Alviano and Faber, 2015], and the grounded fixpoint semantics [Bogaerts *et al.*, 2015]. The whole ADF formalism has even been lifted to the probabilistic case [Polberg and Doder, 2014].

We also addressed the modelling capabilities of ADFs; for a thorough discussion on the relation between ADFs and other argumentations frameworks, see also [Polberg, 2017]. A further application of ADFs in the context of legal reasoning can be found in [Al-Abdulkarim *et al.*, 2014; Al-Abdulkarim *et al.*, 2016]. The use of ADFs in text exploration has been investigated in [Cabrio and Villata, 2016]. Finally, we discussed the GRAPPA approach which makes use of ADF-like semantics in a flexible graph-based formalism. GRAPPA is the formal system underlying a mobile argumentation app developed by Pührer [2017].

Acknowledgements

We thank the anonymous reviewers for various comments which helped to significantly improve the chapter. This research has been supported by the German Research Foundation (DFG) (project BR 1817/7-2), the Austrian Science Fund (FWF) (projects I2854, P25521, and P30168-N31), and by Academy of Finland under grants 251170 COIN and 284591.

BIBLIOGRAPHY

[Al-Abdulkarim *et al.*, 2014] Latifa Al-Abdulkarim, Katie Atkinson, and Trevor J. M. Bench-Capon. Abstract dialectical frameworks for legal reasoning. In Rinke Hoekstra, editor, *Proceedings of the 27th Conference on Legal Knowledge and Information Systems (JURIX)*, volume 271 of *Frontiers in Artificial Intelligence and Applications*, pages 61–70. IOS Press, 2014.

[Al-Abdulkarim *et al.*, 2016] Latifa Al-Abdulkarim, Katie Atkinson, and Trevor J. M. Bench-Capon. A methodology for designing systems to reason with legal cases using abstract dialectical frameworks. *Artif. Intell. Law*, 24(1):1–49, 2016.

[Alviano and Faber, 2015] Mario Alviano and Wolfgang Faber. Stable model semantics of abstract dialectical frameworks revisited: A logic programming perspective. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2684–2690. AAAI Press, 2015.

[Alviano *et al.*, 2016] Mario Alviano, Wolfgang Faber, and Hannes Strass. Boolean functions with ordered domains in Answer Set Programming. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 879–885. AAAI Press, 2016.

[Amgoud and Cayrol, 2002] Leila Amgoud and Claudette Cayrol. A reasoning model based on the production of acceptable arguments. *Ann. Math. Artif. Intell*, 34(1-3):197–215, 2002.

[Amgoud and Vesic, 2011] Leila Amgoud and Srdjan Vesic. A new approach for preference-based argumentation frameworks. *Ann. Math. Artif. Intell*, 63:149–183, 2011.

[Baroni *et al.*, 2011] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. AFRA: Argumentation framework with recursive attacks. *Int. J. Approx. Reasoning*, 52(1):19–37, 2011.

[Baumann and Strass, 2016] Ringo Baumann and Hannes Strass. On the number of bipolar Boolean functions. *International Journal of Integer Sequences*, 2016. Submitted.

[Bench-Capon, 2003] Trevor J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.*, 13(3):429–448, 2003.

[Bogaerts *et al.*, 2015] Bart Bogaerts, Joost Vennekens, and Marc Denecker. Grounded fixpoints and their applications in knowledge representation. *Artif. Intell.*, 224:51–71, 2015.

[Brewka and Gordon, 2010] Gerhard Brewka and Thomas F. Gordon. Carneades and Abstract Dialectical Frameworks: A Reconstruction. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo R. Simari, editors, *Proceedings of the 3rd International Conference on Computational Models of Argument (COMMA)*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 3–12. IOS Press, 2010.

[Brewka and Woltran, 2010] Gerhard Brewka and Stefan Woltran. Abstract Dialectical Frameworks. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczyński, editors, *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 102–111. AAAI Press, 2010.

[Brewka and Woltran, 2014] Gerhard Brewka and Stefan Woltran. GRAPPA: A semantical framework for graph-based argument processing. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 153–158. IOS Press, 2014.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[Brewka *et al.*, 2013] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes P. Wallner, and Stefan Woltran. Abstract Dialectical Frameworks Revisited. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 803–809. AAAI Press / IJCAI, August 2013.

[Brewka *et al.*, 2014] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of Dung Frameworks and Their Role in Formal Argumentation. *IEEE Intelligent Systems*, 29(1):30–38, 2014.

[Brewka *et al.*, 2017] Gerhard Brewka, Martin Diller, Georg Heissenberger, Thomas Linsbichler, and Stefan Woltran. Solving advanced argumentation problems with answer-set programming. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1077–1083. AAAI Press, 2017.

[Cabrio and Villata, 2016] Elena Cabrio and Serena Villata. Abstract dialectical frameworks for text exploration. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016)*, pages 85–95. SciTePress, 2016.

[Caminada and Amgoud, 2007] Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artif. Intell.*, 171(5–6):286–310, 2007.

[Caminada and Gabbay, 2009] Martin W. A. Caminada and Dov M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2-3):109–145, 2009.

[Cayrol and Lagasquie-Schiex, 2013] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: Towards a better understanding. *Int. J. Approx. Reasoning*, 54(7):876–899, 2013.

[Coste-Marquis *et al.*, 2012] Sylvie Coste-Marquis, Sébastien Konieczny, Pierre Marquis, and Mohand Akli Ouali. Weighted attacks in argumentation frameworks. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 593–597. AAAI Press, 2012.

[Denecker *et al.*, 2000] Marc Denecker, Victor W. Marek, and Miroslaw Truszczyński. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

[Denecker *et al.*, 2004] Marc Denecker, Victor W. Marek, and Miroslaw Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Inf. Comput.*, 192(1):84–121, 2004.

[Diller *et al.*, 2015] Martin Diller, Johannes Peter Wallner, and Stefan Woltran. Reasoning in abstract dialectical frameworks using quantified boolean formulas. *Argument & Computation*, 6(2):149–177, 2015.

[Dunne *et al.*, 2011] Paul E. Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artif. Intell.*, 175(2):457–486, 2011.

[Dunne *et al.*, 2015] Paul E. Dunne, Wolfgang Dvořák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. *Artif. Intell.*, 228:153–178, 2015.

[Ellmauthaler and Strass, 2013] Stefan Ellmauthaler and Hannes Strass. The DIAMOND system for argumentation: Preliminary report. In Michael Fink and Yuliya Lierler, editors, *Proceedings of the Sixth International Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP)*, September 2013.

[Ellmauthaler and Strass, 2014] Stefan Ellmauthaler and Hannes Strass. The diamond system for computing with abstract dialectical frameworks. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA)*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 233–240. IOS Press, 2014.

[Ellmauthaler and Strass, 2016] Stefan Ellmauthaler and Hannes Strass. DIAMOND 3.0 – A native C++ implementation of DIAMOND. In Pietro Baroni, editor, *Proceedings of the Sixth International Conference on Computational Models of Argument (COMMA)*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 471–472, Potsdam, Germany, September 2016. IOS Press.

[Farley and Freeman, 1995] Arthur M. Farley and Kathleen Freeman. Burden of proof in legal argumentation. In *Proceedings of the 5th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 156–164, 1995.

[Gaggl and Strass, 2014] Sarah Alice Gaggl and Hannes Strass. Decomposing Abstract Dialectical Frameworks. In Simon Parsons, Nir Oren, and Chris Reed, editors, *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA)*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 281–292. IOS Press, 2014.

[Gaggl *et al.*, 2015] Sarah Alice Gaggl, Sebastian Rudolph, and Hannes Strass. On the computational complexity of naive-based semantics for abstract dialectical frameworks. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2985–2991. IJCAI/AAAI, 2015.

[Gebser *et al.*, 2011] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The Potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.

[Gogic *et al.*, 1995] Goran Gogic, Henry Kautz, Christos Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 862–869. Morgan Kaufmann, 1995.

[Gordon *et al.*, 2007] Thomas F. Gordon, Henry Prakken, and Douglas Walton. The carneades model of argument and burden of proof. *Artif. Intell.*, 171(10-15):875–896, 2007.

[Heißenberger, 2016] G. Heißenberger. A system for advanced graphical argumentation formalisms. Master's thesis, TU Wien, 2016. Available at http://www.dbai.tuwien.ac.at/proj/adf/grappavis/.

[Hunter, 2013] Anthony Hunter. A probabilistic approach to modelling uncertain logical arguments. *Int. J. Approx. Reasoning*, 54(1):47 – 81, 2013.

[Linsbichler *et al.*, 2016a] Thomas Linsbichler, Jörg Pührer, and Hannes Strass. Characterizing realizability in abstract argumentation. In Gabriele Kern-Isberner and Renata Wassermann, editors, *Proceedings of the 16th International Workshop on Non-Monotonic Reasoning (NMR)*, April 2016.

[Linsbichler *et al.*, 2016b] Thomas Linsbichler, Jörg Pührer, and Hannes Strass. A uniform account of realizability in abstract argumentation. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 252–260. IOS Press, 2016.

[Linsbichler, 2014] Thomas Linsbichler. Splitting abstract dialectical frameworks. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA)*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 357–368. IOS Press, 2014.

[Lonsing and Biere, 2010] Florian Lonsing and Armin Biere. DepQBF: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010.

[Martínez *et al.*, 2008] Diego C. Martínez, Alejandro Javier García, and Guillermo Ricardo Simari. An abstract argumentation framework with varied-strength attacks. In Gerhard Brewka and Jérôme Lang, editors, *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 135–144. AAAI Press, 2008.

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

[Nielsen and Parsons, 2007] Søren Nielsen and Simon Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In Nicolas Maudet, Simon Parsons, and Iyad Rahwan, editors, *Proc. ArgMAS*, volume 4766 of *Lecture Notes in Computer Science*, pages 54–73. Springer, 2007.

[Nouioua, 2013] Farid Nouioua. AFs with necessities: Further semantics and labelling characterization. In Weiru Liu, V. S. Subrahmanian, and Jef Wijsen, editors, *Scalable Uncertainty Management - 7th International Conference, SUM 2013*, volume 8078 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2013.

[Oren and Norman, 2008] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In Philippe Besnard, Sylvie Doutre, and Anthony Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 276–284. IOS Press, 2008.

[Pelov *et al.*, 2007] Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming*, 7(3):301–353, 2007.

[Polberg and Doder, 2014] Sylwia Polberg and Dragan Doder. Probabilistic abstract dialectical frameworks. In Eduardo Fermé and João Leite, editors, *Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 8761 of *Lecture Notes in Artificial Intelligence*, pages 591–599. Springer, 2014.

[Polberg and Oren, 2014] Sylwia Polberg and Nir Oren. Revisiting support in abstract argumentation systems. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA)*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 369–376, 2014.

[Polberg and Wallner, 2017] Sylwia Polberg and Johannes P. Wallner. Preliminary report on complexity analysis of extension-based semantics of abstract dialectical frameworks. Technical Report DBAI-TR-2017-103, TU Wien, 2017.

[Polberg *et al.*, 2013] Sylwia Polberg, Johannes P. Wallner, and Stefan Woltran. Admissibility in the Abstract Dialectical Framework. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent System (CLIMA)*, volume 8143 of *Lecture Notes in Artificial Intelligence*, pages 102–118. Springer, 2013.

[Polberg, 2014a] Sylwia Polberg. Extension-based semantics of abstract dialectical frameworks. In Sébastien Koniecny and Hans Tompits, editors, *Proceedings of the 15th International Workshop on Non-Monotonic Reasoning, NMR 2014*, pages 273–282, 2014.

[Polberg, 2014b] Sylwia Polberg. Extension-based semantics of abstract dialectical frameworks. In Ulle Endriss and João Leite, editors, *Proceedings of the 7th European Starting AI Researcher Symposium, STAIRS 2014*, pages 240–249, 2014.

[Polberg, 2015] Sylwia Polberg. Revisiting extension-based semantics of abstract dialectical frameworks. Technical Report DBAI-TR-2015-88, TU Wien, 2015.

[Polberg, 2016] Sylwia Polberg. Understanding the abstract dialectical framework. In Loizos Michael and Antonis C. Kakas, editors, *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016*, volume 10021 of *Lecture Notes in Computer Science*, pages 430–446, 2016.

[Polberg, 2017] Sylwia Polberg. Intertranslatability of abstract argumentation frameworks. Technical Report DBAI-TR-2015-104, TU Wien, 2017.

[Pollock, 1987] John L Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987.

[Pührer, 2015] Jörg Pührer. Realizability of three-valued semantics for abstract dialectical frameworks. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3171–3177. AAAI Press, 2015.

[Pührer, 2017] Jörg Pührer. ArgueApply: A mobile app for argumentation. In Marcello Balduccini and Tomi Janhunen, editors, *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2017)*, volume 10377 of *Lecture Notes in Computer Science*, pages 250–262. Springer, 2017.

[Simari and Loui, 1992] Guillermo R. Simari and Ronald P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell.*, 53(2–3):125 – 157, 1992.

[Son and Pontelli, 2007] Tran Cao Son and Enrico Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7(3):355–375, 2007.

[Strass and Ellmauthaler, 2017] Hannes Strass and Stefan Ellmauthaler. goDIAMOND 0.6.6 – ICCMA 2017 system description, 2017. Second International Competition on Computational Models of Argumentation: `http://www.dbai.tuwien.ac.at/iccma17/`.

[Strass and Wallner, 2014] Hannes Strass and Johannes P. Wallner. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 101–110. AAAI Press, 2014.

[Strass and Wallner, 2015] Hannes Strass and Johannes Peter Wallner. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. *Artif. Intell.*, 226:34–74, 2015.

[Strass, 2013a] Hannes Strass. Approximating Operators and Semantics for Abstract Dialectical Frameworks. *Artificial Intelligence*, 205:39–70, 2013.

[Strass, 2013b] Hannes Strass. Instantiating Knowledge Bases in Abstract Dialectical Frameworks. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, volume 8143 of *Lecture Notes in Artificial Intelligence*, pages 86–101. Springer, 2013.

[Strass, 2015a] Hannes Strass. Expressiveness of two-valued semantics for abstract dialectical frameworks. *J. Artif. Intell. Res. (JAIR)*, 54:193–231, 2015.

[Strass, 2015b] Hannes Strass. Instantiating rule-based defeasible theories in abstract dialectical frameworks and beyond. *J. Log. Comput.*, 2015.

[Strass, 2015c] Hannes Strass. The relative expressiveness of abstract argumentation and logic programming. In Sven Koenig and Blai Bonet, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1625–1631. AAAI Press, 2015.

[Thimm, 2012] Matthias Thimm. A probabilistic semantics for abstract argumentation. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 750–755. IOS Press, 2012.

[van Emden and Kowalski, 1976] Maarten H. van Emden and Robert A. Kowalski. The Semantics of Predicate Logic as a Programming Language. *J. ACM*, 23(4):733–742, 1976.

[Wallner, 2014] Johannes P. Wallner. *Complexity Results and Algorithms for Argumentation - Dung's Frameworks and Beyond.* PhD thesis, TU Vienna, Institute of Information Systems, 2014.

[Wyner *et al.*, 2013] Adam Wyner, Trevor J. M. Bench-Capon, and Paul E. Dunne. On the instantiation of knowledge bases in abstract argumentation frameworks. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, volume 8143 of *Lecture Notes in Artificial Intelligence*, pages 34–50. Springer, 2013.

Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass
Leipzig University
Computer Science Institute
Intelligent Systems Department
Augustusplatz 10–11
04109 Leipzig, Germany

Johannes Wallner, Stefan Woltran
Vienna University of Technology
Institute of Information Systems
Database and Artificial Intelligence Group 184/2
Favoritenstraße 9–11
A-1040 Vienna, Austria