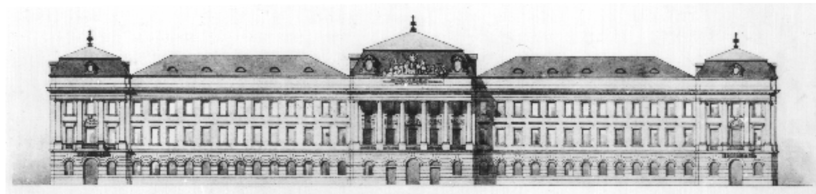
dbai

TECHNICAL REPORT



INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG DATENBANKEN UND ARTIFICIAL INTELLIGENCE

Extension–Based Semantics of Abstract Dialectical Frameworks

DBAI-TR-2014-85

Sylwia Polberg

Institut für Informationssysteme
Abteilung Datenbanken und
Artificial Intelligence
Technische Universität Wien
Favoritenstr. 9
A-1040 Vienna, Austria
Tel: +43-1-58801-18403
Fax: +43-1-58801-18493
sekret@dbai.tuwien.ac.at
www.dbai.tuwien.ac.at

DBAI TECHNICAL REPORT
2014

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DBAI TECHNICAL REPORT
DBAI TECHNICAL REPORT DBAI-TR-2014-85, 2014

Extension–Based Semantics of Abstract Dialectical Frameworks

Sylwia Polberg [1]

Abstract. One of the most common tools in abstract argumentation are the argumentation frameworks and their associated semantics. While the framework is used to represent a given problem, the semantics define methods of solving it, i.e. they describe requirements for accepting and rejecting arguments. The basic available structure is the Dung framework, AF for short. It is accompanied by a variety of semantics including grounded, complete, preferred and stable. Although powerful, AFs have their shortcomings, which led to development of numerous enrichments. Among the most general ones are the abstract dialectical frameworks, also known as the ADFs. They make use of the so–called acceptance conditions to represent arbitrary relations. This level of abstraction brings not only new challenges, but also requires addressing problems inherited from other frameworks. One of the most controversial issues, recognized not only in argumentation, concerns the support cycles. In this paper we introduce a new method to ensure acyclicity of the chosen arguments and present a family of extension–based semantics built on it. We also continue our research on the semantics that permit cycles and fill in the gaps from the previous works. Moreover, we provide ADF versions of the properties known from the Dung setting. Finally, we also introduce a classification of the developed sub–semantics and relate them to the existing labeling–based approaches.

[1]Institute for Information Systems 184/2, Technische Universität Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria. E-mail: polberg@dbai.tuwien.ac.at

Acknowledgements: The author is funded by the Vienna PhD School of Informatics.

Contents

1 Introduction

Argumentation, in one form or another, is present in our daily lives. Over the last years, it has become an influential subfield of artificial intelligence, with applications ranging from legal reasoning [8] or dialogues and persuasion [29, 36] to medicine [26, 27] or eGovernment [2]. Till today, various formalisms and classifications of types of argumentation have been created [37]. One of them is *abstract argumentation*. It has became especially popular thanks to the research of Phan Minh Dung [23]. Although the framework he has developed was relatively limited, as it took into account only the conflict relation between the arguments, it inspired a search for more general models. Throughout the years, many different argumentation frameworks were created, ranging from the ones employing various measures of arguments or relations strengths and preferences [1, 22, 7, 30] to ones that focus on researching new types of interactions between the framework elements [4, 18, 31, 32, 34]. An overview of available structures can be found in [12]. One of the most general enrichments of the latter type are the abstract dialectical frameworks, ADFs for short [13]. Instead of extending the Dung's frameworks with elements representing new types of relations each time it is needed, they make use of so–called acceptance conditions to express arbitrary interactions between the arguments. However, a framework is just a way of representing a problem and cannot be considered a suitable argumentation tool without properly developed semantics.

The semantics of a framework are meant to represent what is considered rational. We may require the chosen opinion to be e.g. consistent, defensible, providing counterarguments for what we cannot accept and so on. Given many of the advanced semantics, such as grounded or complete, we can observe that they return same results when faced with a "pretty" Dung's framework [23], i.e. free from any types of cycles, directed or not. The differences between the approaches become more visible when we work with more complicated examples. Sometimes we end up with a case where none of the available semantics return sets of arguments we could consider satisfactory. This gave rise to new approaches, each trying to tackle this issue. For example, for handling indirect attacks and defenses we have prudent and careful semantics [21, 20]; for the problem of even and odd attack cycles we can resort to some of the SCC–recursive semantics [6]; while for treatment of self attackers, sustainable and tolerant semantics were developed [9]. Introducing a new type of relation, for example support, creates additional problems such as support cycles and being supported and attacked by the same argument. Many of these issues can be seen as on the inside, i.e. "what can I consider rational?". On the other hand, some can be understood as on the "outside", e.g. "what can be considered a valid attacker, what should I defend from?". Various examples of such behavior exist even in the Dung setting. An admissible extension is conflict–free and defends against attacks carried out by any other argument in the framework. Then, we can say that self–attackers are not rational and limit the set of arguments we have to protect our choice from. If we move to a bipolar setting, new restrictions can be introduced – for example, we can now demand that we only defend from arguments not in support cycles, thus again trimming the set of attackers. From this perspective semantics can be seen as a two–person discussion, describing what "I can claim" and "what my opponent can claim". This is also the point of view that we follow in this paper and that is used to create our semantics classification. Please note that this sort of dialogue perspective can already be found in argumentation [24, 28], although it is used in a slightly different context.

Various extension–based semantics for ADFs have already been proposed in the original paper [13]. Unfortunately, some of them were defined only for a particular ADF subclass called bipolar and were suitable for certain types of situations. Therefore, only three of them – conflict–free, model and grounded – remain. The research in [11, 38] resulted in establishing a family of semantics we can qualify as labeling–based. Although they resolve the problems of the initial formulations, they have their own drawbacks. They are described in terms of e.g. fixpoints of a three–valued characteristic operator, which is based on consensus of acceptance conditions. In this formulation, it is not always visible at the first glance how defense and other notions known from the Dung setting behave in ADFs. Moreover, verifying an existing interpretation rather than constructing one from some initial data can result in an argument affecting his own status in face of self–dependencies, which is not always a desirable property when a framework can express support. Finally, shifting from two–valued to three–valued setting is more than just a structural change. While in the extension–based semantics we often aim to accept as many arguments as the rationality allows, in labeling setting knowing that something is true is equally important to knowing it is false. Thus, one makes use of information maximality rather than subset maximality, which in a bipolar setting creates differences not present in AFs. Although we find this method to be suitable for the labeling intuitions, we are missing semantics that would still let us focus on argument's acceptance.

The most controversial type of self–dependency concerns the so–called support cycles and is handled differently from formalism to formalism. Among the best known structures are the Bipolar Argumentation Frameworks (BAFs for short) [16, 18], Argumentation Frameworks with Necessities (AFNs) [32] and Evidential Argumentation Systems (EASs) [34]. While AFNs and EASs discard support cycles, BAFs seem to leave the question open. In ADFs cycles are permitted unless the intuition of a given semantics is clearly against it. This variety is not an error in any of the structures; it is caused by the fact that a standard Dung semantics can be interpreted in several ways in a setting that allows more types of relations. Moreover, since one can find arguments both for and against any of the cycle treatments, it should not come as a surprise that there is no consensus as to what approach is the best.

The aim of this paper is to introduce a family of extension–based semantics and to specialize them to handle the problem of support cycles, as it seems to be the biggest difference between the approaches of the current frameworks that allow positive relations. Consequently, we present methods for ensuring acyclicity in ADFs. Furthermore, a classification our sub–semantics in the inside–outside fashion that we have described before is introduced. We also recall our previous research on admissibility in [35] and show how it fits into our system. Our results also include which known properties, such as Fundamental Lemma, carry over from the Dung framework. Finally we provide an initial analysis of similarities and differences between the extension and labeling–based semantics in the context of produced extensions.

The report is structured as follows. In Sections 2, 3 and 4 we provide a short recap on AFs, BAFs, AFNs, EASs and ADFs. Then we introduce the new extension–based semantics in Section 5 and analyze their behavior in Section 6. We close the paper with an analysis of similarities and differences between the newly created and existing, labeling–based approach.

2 Dung's Argumentation Frameworks

Let us start from the basics: the abstract argumentation framework by Dung [23].

Definition 2.1. *A **Dung's abstract argumentation framework** (AF for short) is a pair (A, R), where A is a set of **arguments** and $R \subseteq A \times A$ represents an **attack** relation.*

AFs can be simply represented as directed graphs. We will now briefly recall the available semantics, for more details we refer the reader to [3].

Definition 2.2. *Let $AF = (A, R)$ be a Dung's framework. We say that an argument $a \in A$ is **defended** by a set E in AF[1], if for each $b \in A$ s.t. $(b, a) \in R$, there exists $c \in E$ s.t. $(c, b) \in R$. A set $E \subseteq A$ is:*

- ***conflict–free** in AF iff for each $a, b \in E$, $(a, b) \notin R$.*
- ***naive** in AF iff it is maximal w.r.t. set inclusion conflict–free.*
- ***admissible** in AF iff it is conflict–free and defends all of its members.*
- ***preferred** in AF iff it is maximal w.r.t. set inclusion admissible.*
- ***complete** in AF iff it is admissible and all arguments defended by it are contained in it.*
- ***stable** in AF iff it is conflict–free and for each $a \in A \setminus E$ there exists an argument $b \in E$ s.t. $(b, a) \in R$.*

The stable semantics is somewhat different than the rest in the sense that depending on the given framework, it might not produce any extensions. This problem is addressed with maximizing the amount of covered arguments [15]:

Definition 2.3. *Let E^+ be the set of arguments attacked by E. $E^+ \cup E$ is the **range** of E. A conflict–free set is stable iff $E^+ = A \setminus E$. A complete extension E is **semi–stable** iff its range is maximal w.r.t. set inclusion.*

We close the list with the grounded semantics. It basically represents the knowledge that we can only build from the initial (i.e. unattacked) arguments, i.e. starting with an empty set we first include the initial arguments, then add all elements defended by the set and continue until nothing more is added. The formal definition is given by the means of the characteristic function of AF:

Definition 2.4. *The **characteristic function** $F_{AF} : 2^A \to 2^A$ is defined as: $F_{AF}(E) = \{a \mid a$ is defended by E in $AF\}$. The **grounded extension** is the least fixed point of F_{AF}.*

Furthermore, other semantics can also be described in terms of the characteristic function; for example, a conflict–free set E is admissible iff $E \subseteq F_{AF}(E)$ and complete iff $E = F_{AF}(E)$.

Please note there is also an alternative way to compute the grounded extension:

[1] Defense is often substituted with acceptability: say that a is acceptable w.r.t. E if E is defends a.

Proposition 2.5. *The unique **grounded extension** of AF is defined as the outcome E of the following algorithm. Let us start with $E = \emptyset$:*

1. *put each argument $a \in A$ which is not attacked in AF into E; if no such argument exists, return E.*
2. *remove from AF all (new) arguments in E and all arguments attacked by them (together with all adjacent attacks) and continue with Step 1.*

What we have described above forms a family of so–called extension–based semantics. We now continue with the labeling–based ones, which are thoroughly explained in [14].

Definition 2.6. *A three–valued labeling is simply a total function $Lab : A \rightarrow \{in, out, undec\}$*[2].

At the heart of the admissibility–based semantics lies the concept of legality:

Definition 2.7. *We say that an in–labeled argument is **legally in** iff all its attackers are labeled out. An out–labeled argument is **legally out** iff at least one its attacker is labeled in. Finally, an $undec$–labeled argument is **legally undec** iff not all of its attackers are labeled out and it does not have an attacker that is labelled in.*

Definition 2.8. *We then say that a labeling Lab is:*

- ***admissible** in AF iff each in–labeled argument is legally in and each out–labeled argument is legally out.*
- ***complete** in AF if it is admissible and every $undec$–labeled argument is legally $undec$.*
- ***preferred** in AF if it is complete and the set of arguments labeled in is maximal w.r.t. set inclusion.*
- ***grounded** in AF if it is complete and the set of arguments labeled in is minimal w.r.t. set inclusion.*
- ***semi–stable** in AF if it is complete and the set of elements mapped to $undec$ is minimal w.r.t set inclusion.*
- ***stable** in AF if it is complete and the set of elements mapped to $undec$ is empty.*

The correspondence between the labeling–based and extension–based has already been studied in [14, 3]:

Theorem 2.9. *Let $E \subseteq A$ be a σ–extension of AF, where $\sigma \in \{$admissible, complete, grounded, preferred, stable, semi–stable$\}$. Then $(E, E^+, A \setminus (E \cup E^+))$ is a σ–labeling of AF.*

Let Lab be a σ–labeling of AF, where $\sigma \in \{$admissible, complete, grounded, preferred, stable, semi–stable $\}$. Then $in(Lab)$ is a σ–extension of AF.

[2]Sometimes the $\mathtt{t}, \mathtt{f}$ and $\mathbf{u}$ notation is also used.

Remark. Depending on the semantics, there can be more than one labeling corresponding to a given extension. Let E^- be the set of arguments that attack E. Obviously, E defends its members iff $E^- \subseteq E^+$. Therefore, for a labeling to be admissible it suffices that the set of out arguments contains E^-; on the other hand, due to legality it cannot map more than E^+. This gives us a certain freedom in assignments. On the other hand, for example stable semantics possesses a one to one correspondence between the labelings and extensions.

Finally, we would like to recall several important lemmas and theorems from the original paper on AFs [23]. The so–called Fundamental Lemma is as follows:

Lemma 2.10. ***Dung's Fundamental Lemma*** *Let E be an admissible extension, a and b arguments that are defended by E. Then $E' = E \cup \{a\}$ is admissible and b is defended by E'.*

The next two formulations show some relations between the existing semantics.

Theorem 2.11. *Every stable extension is a preferred extension, but not vice versa.*

Theorem 2.12. *The following holds:*

1. *Every preferred extension is a complete extension, but not vice versa.*

2. *The grounded extension is the least w.r.t. set inclusion complete extension.*

3. *The complete extensions form a complete semilattice w.r.t. set inclusion.* [3]

Example 2.13. *Consider the Dung framework $AF = (A, R)$ with $A = \{a, b, c, d, e\}$ and the attack relation $R = \{(a, b),\ (c, b),\ (c, d),\ (d, c),\ (d, e),\ (e, e)\}$, as depicted in Figure 1. It has eight conflict–free extensions in total, namely $\{a, c\}$,$\{a, d\}$, $\{b, d\}$, $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$ and $\emptyset$. As b is attacked by an unattacked argument, it cannot be defended against it and will not be in any admissible extension. From this $\{a, c\}$, $\{a, d\}$ and $\{a\}$ are complete. We end up with two preferred extensions, $\{a, c\}$ and $\{a, d\}$. However, only $\{a, d\}$ is stable, and $\{a\}$ is the grounded extension.*

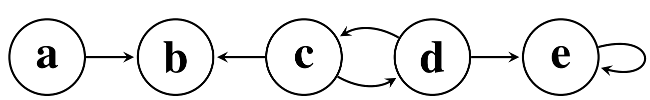


Figure 1: Sample Dung framework

3 Argumentation Frameworks with Support

Although the Dung's framework is a powerful tool, it has its shortcomings. Having only a binary attack at hand limits what can be modeled naturally, and what requires additional modifications which can make the representation of a problem and verifying the answer more complicated. Not

[3]A partial order $(A, \leq)$ is a complete semilattice iff each nonempty subset of A has a glb and each increasing sequence of S has a lub.

surprisingly, this framework has been generalized in various ways in order to address its deficiencies (an overview can be found in [19]). In the context of this report, the enrichments that permit new types of relations are most interesting.

Although many studies focused on developing the attack relation, with time it was acknowledged that a positive interaction between arguments beyond defense also needs to be expressed. Initially, there was hope that since Dung's framework has one abstract attack, one type of support would be sufficient [16]. However, various arguments and examples against this claim have been given, and more specialized forms of support have been researched. Currently the most recognized frameworks following the Dung representation are the Bipolar Argumentation Framework BAF [18], Argumentation Framework with Necessities AFN [32] and Evidential Argumentation System EAS [34]. The approaches towards modeling support can be classified in two ways. First of all we have the BAF style, more in line with meta–argumentation, where we can create coalition arguments or, depending on the type of positive relation that is used, we derive advanced conflicts and evaluate the resulting framework in a Dung manner. Although this study does not discuss certain problems of a bipolar setting such as support cycles, it provides a valuable insight into the consequences of using positive relations. The other approach, more visible in AFNs and EASs, treats support as a fully valued interaction and adapts semantics in an appropriate manner, rather than trying to translate the structure back into the Dung setting. We will go through them step by step. Although the translation of these structures into ADFs is a matter of ongoing work and not a topic we want to discuss in this report, the differences between the frameworks will further exemplify the directions of the semantics we have taken in ADFs.

3.1 Bipolar Argumentation Frameworks

The original bipolar argumentation framework BAF [16] studied a relation we will refer to as **abstract support**:

Definition 3.1. *A **bipolar argumentation framework** is a tuple (A, R, S), where A is a set of **arguments**, $R \subseteq A \times A$ represents the **attack** relation and $S \subseteq A \times A$ the **support**. It is also assumed that $R \cap S = \emptyset$*[4].

The biggest difference between this abstract relation and any other interpretation of support, or even conflict, is the fact that it did not affect the acceptability of an argument. By this, we understand that an argument did not require any form of support and was able to stand "on its own". The positive interaction was used to derive additional indirect forms of conflict, which were later used to enhance the semantics from the Dung setting. The first developed type was the supported attack. Later, in [17] the *secondary attack* was also introduced (first referred to as diverted).

Definition 3.2. *We say that an argument a **support attacks** argument b, if there exists some argument c s.t. there is a sequence of supports from a to c (i.e. $aS...Sc$) and cRb. We say that a **secondary attacks** b if there is some argument c s.t. $cS...Sb$ and aRc.*

[4] This requirement is dropped in later works [18].

These additional notions are now used to form stronger version of known semantics. Please note that the definition of defense is the same as in the Dung setting (i.e. requires direct attack).

Definition 3.3. *Let (A, R, S) be a BAF. We say that $B \subseteq A$ is* ***+conflict–free*** *iff $\nexists a, b \in B$ s.t. a (directly or indirectly) attacks b. B is* ***safe*** *iff $\nexists b \in A$ s.t. b is at the same time (directly or indirectly) attacked by B and either there is a sequence of supports from an element of B to b, or $b \in B$. B is closed under S iff $\forall b \in B, a \in A$, if bSa then $a \in B$. Then B is:*

- ***d–admissible*** *iff it is +conflict–free and defends all its elements*
- ***s–admissible*** *iff it is safe and defends all its elements*
- ***c–admissible*** *iff it is +conflict–free, closed for S and defends all its elements*
- ***d–/s–/c–preferred*** *iff it is maximal w.r.t. set inclusion d–/s–/c–admissible*
- ***stable*** *iff it is +conflict–free and $\forall b \notin B$, b is (directly or indirectly) attacked by B.*

The weak dependency between an argument and its supporter led to development of more specific interpretations, most notably the deductive, necessary and evidential support. The first one remained in the BAF setting, while the latter two were developed in different frameworks. We say that a *deductively supports* b if acceptance of a implies the acceptance of b [10] and not acceptance of b implies non acceptance of a. Although originally used rather for coalitions and meta–argumentation purposes, it is also studied in a standard setting in [18]. The deductive behavior of support in BAFs is achieved by introducing another type of indirect conflict, namely the *mediated attack*. Further study also motivated the *super–mediated attack*.

Definition 3.4. *There is a* ***mediated attack*** *from a to b iff there is some argument c s.t. there is a sequence of supports from b to c and aRc. There is a* ***super–mediated attack*** *from a to b iff there is some argument c s.t. a direct or supported attacks c and b supports c.*

Finally, it is easy to see that BAFs do not make any special acyclicity assumptions as to the support relation[5]. Thus, cyclic arguments are considered valid attackers that can be used both by us and by the opponent.

3.2 Argumentation Frameworks with Necessities

The necessary support in its binary form was first developed in [33]. We say that an argument a *necessary supports* b if we need to assume a in order to accept b. The developed semantics were built around the supported and secondary attacks and discarded any support cycles. However, they not always returned intended results. Therefore, we would like to focus on the more recent formulation that was presented in [32], this time with a set form of support.

[5]Only in the case of stable semantics the framework is assumed to be acyclic

Definition 3.5. *An **argumentation framework with necessities** is a tuple (A, R, N), where A is the set of **arguments**, $R \subseteq A \times A$ represents (binary) **attacks**, and $N \subseteq (2^A \setminus \emptyset) \times A$ is the **necessity relation**.*

Given a set $B \subseteq A$ and an argument a, BNa should be read as "at least one element of B needs to be present in order to accept a". The AFN semantics are built around the notions of coherence:

Definition 3.6. *We say that a set of arguments B is **coherent** iff every $b \in B$ is powerful, i.e. there exists a sequence $a_0, .., a_n$ of some elements of B s.t $a_n = b$, there is no $C \subseteq A$ s.t. CNa_0, and finally for $1 \leq i \leq n$ it holds that for every set $C \subseteq A$ if CNa_i, then $C \cap \{a_0, ..., a_{i-1}\} \neq \emptyset$. A coherent set B is **strongly coherent** iff it is conflict–free.*

Although it may look a bit complicated at first, the definition of coherence grasps the intuition that we need to provide sufficient acyclic support for the arguments we want to accept. Defense in AFNs is understood as the ability to provide support and to counter the attacks from any coherent set. Using these notions, the AFN semantics are built in a way corresponding to Dung semantics.

Definition 3.7. *Let (A, R, N) be an AFN. We say that a set $B \subseteq A$ **defends** a, if $B \cup \{a\}$ is coherent and for every $c \in A$, if cRa then for every coherent set $C \subseteq A$ containing c, BRC. The set of arguments **deactivated** by B is defined by $B^+ = \{a \mid BRA \text{ or there is } E \subseteq A \text{ s.t. } ENa \text{ and } B \cap E = \emptyset\}$. Finally, we have that B is:*

- ***admissible** iff it is strongly coherent and defends all of its arguments.*
- ***preferred** iff it is maximal w.r.t. set inclusion admissible.*
- ***complete** iff it is admissible and contains any argument it defends.*
- ***stable** iff B is complete and $B^+ = A \setminus B$.*

It is easy to see that, through the notion of coherency, AFNs discard cyclic arguments both on the "inside" and the "outside". This means we cannot accept them in an extension and they are not considered as valid attackers.

3.3 Evidential Argumentation Systems

The last type of support we will consider here is the the *evidential support* [34]. It distinguishes between standard and *prima facie* arguments. The latter are the only ones that are valid without any support. Every other argument that we want to accept needs to be supported by at least one prima facie argument, be it directly or not. While the acyclicity in the necessary support required us to trace back to either an attacker or an initial argument, the evidential support restricts this even further by allowing us to go back to only a subgroup of the initial arguments, marked as prima facie.

Definition 3.8. *An **evidential argumentation system** (EAS) is a tuple (A, R, E) where A is a set of **arguments**, $R \subseteq (2^A \setminus \emptyset) \times A$ is the **attack relation**, and $E \subseteq (2^A \setminus \emptyset) \times A$ is the **evidential support**. We assume that $\nexists x \in 2^A, y \in A$ s.t. xRy and xEy. The prima facie arguments are represented with a single one $\eta \in A$ referred to as environment or evidence. Consequently, $\nexists (x, y) \in R$ where $\eta \in x$; and $\nexists x$ where $(x, \eta) \in R$ or $(x, \eta) \in E$.*

The difference between the structures of EAS and AFN lies in the fact that the former reads *"sets A and B support an argument x"* as *"all elements of A or all elements of B are required to assume x"*, while the latter as *"at least one element of A and at least one element of B is required to assume x"*. The idea that the valid arguments (and attackers) need to trace back to the environment is captured with the notions of e–support and e–supported attack[6]. From now on we assume an EAS $EF = (A, R, E)$.

Definition 3.9. *An argument $a \in A$ has **evidential support** (e–support) from a set $X \subseteq A$ iff:*

1. *$a = \eta$; or*
2. *There is a non-empty $T \subseteq X$ such that TEa and $\forall x \in X$, x has evidential support from $X \setminus \{a\}$*

An argument a is minimally evidentially supported by (or has minimal evidential support from) a set X if there is no set X' such that $X' \subset X$ and a is evidentially supported by X'.

Remark. Note that by this definition η has evidential support from any set.

Definition 3.10. *A set $X \subseteq A$ carries out an **evidence supported attack** (e–supported attack) on a iff $(X', a) \in R$ where $X' \subseteq X$, and for all $x \in X'$, x has evidential support from X.*

We can now continue with EAS semantics. The notion of

Definition 3.11. *An argument a is **acceptable** with respect to a set of arguments $X \subseteq A$ iff*

- *a is evidentially supported by X; and*
- *for any evidence supported attack by a set T against a, it is the case that X carries out an evidence supported attack against $T' \subseteq T$ such that $T \setminus T'$ does not carry out an evidence supported attack on a.*

Definition 3.12. *A set of arguments $X \subseteq A$ is:*

- ***conflict–free** iff there is no $a \in X$ and $X' \subseteq X$ such that $X'Ra$.*
- ***admissible** iff it is conflict–free and all elements of X are acceptable w.r.t. X.*
- ***preferred** iff it is maximal w.r.t. set inclusion admissible.*

[6]The presented definition is slightly different from the one available in [34]. The new version was obtained through personal communication with the author in order to address a technical issue of the original formulation.

- ***stable*** *iff it is conflict–free, e–supports all of its members, and for any e–supported argument $a \notin X$, X e–support attacks either a or every set of arguments minimally supporting a.*

From the fact that every valid argument needs to be grounded in the environment it clearly results that EAS semantics are acyclic both on the inside and outside. In a certain sense this requirement is even stronger than in AFNs, as one is allowed to come back to only to a single special argument rather than any initial one.

4 Abstract Dialectical Frameworks

Abstract dialectical frameworks have been defined in [13] and till today various results as to their semantics, instantiation and complexity have already been published in [11, 35, 38, 39, 40]. The main goal of ADFs is to be able to express arbitrary relations and avoid the need of extending AFs by a new relation sets each time they are needed. This is achieved by the means of the so–called acceptance conditions. They define what sets of arguments related to a given argument should be present for it to be accepted or rejected.

Definition 4.1. *An **abstract dialectical framework** (ADF) as a tuple (S, L, C), where S is a set of abstract **arguments** (nodes, statements), $L \subseteq S \times S$ is a set of **links** (edges) and $C = \{C_s\}_{s \in S}$ is a set of **acceptance conditions**, one condition per each argument.*

Originally, the acceptance conditions were defined in terms of functions:

Definition 4.2. *Let $par(s)$ denote the set of **parents** of an argument s; it consists of those $p \in S$ for which (p,s) $\in L$. Then an **acceptance condition** is given by a total function $C_s : 2^{par(s)} \rightarrow \{in, out\}$.*

Alternatively, one can also use the propositional formula representation [25], i.e. with $C = \{\varphi_s\}_{s \in S}$, which will be more convenient for our purpose. As we will be making use of both extension and labeling–based semantics, we need to provide a short background on interpretations first (more details can be found in [11, 35]). Please note that links represent just connections between arguments, the burden of saying what is the nature of this connection falls to the acceptance conditions. Moreover, the parents of an argument can be easily extracted from the conditions. Thus, we will use the shortened notation and assume an ADF $D = (S, C)$ through the rest of this paper.

4.1 Interpretations and decisiveness

A two (three–valued) interpretation is simply a mapping that assigns truth values (respectively $\{\mathbf{t}, \mathbf{f}\}$ and $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$) to arguments. We will be making use both of partial (i.e. defined only for a subset of S) and full ones. The truth values can be compared with respect to truth ordering, i.e. $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$, or precision (information) ordering: $\mathbf{u} \leq_i \mathbf{t}$ and $\mathbf{u} \leq_i \mathbf{f}$. The latter will be used in the context of labeling semantics. The pair $(\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}, \leq_i)$ forms a complete meet–semilattice with the meet operation $\sqcap$ assigning values in the following way: $\mathbf{t} \sqcap \mathbf{t} = \mathbf{t}$, $\mathbf{f} \sqcap \mathbf{f} = \mathbf{f}$ and $\mathbf{u}$ in all other cases. It can naturally be extended to interpretations: given two interpretations v and v' on S, we say that v' contains more information, denoted $v \leq_i v'$, iff $\forall_{s \in S}\, v(s) \leq_i v'(s)$. Similar follows for the meet operation. In case v is three and v' two–valued, we say that v' extends v. This means that elements mapped originally to $\mathbf{u}$ are now assigned either $\mathbf{t}$ or $\mathbf{f}$. The set of all two–valued interpretations extending v is denoted $[v]_2$.

Example 4.3. *Let $v = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{f}, d : \mathbf{u})$ be a three–valued interpretation. We have two extending interpretations, namely $v' = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{f}, d : \mathbf{t})$ and $v'' = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{f}, d : \mathbf{f})$. Clearly, it holds that $v \leq_i v'$ and $v \leq_i v''$. However, v' and v'' are incomparable w.r.t. $\leq_i$.*

Let now $w = \{a : \mathbf{f}, b : \mathbf{f}, c : \mathbf{f}, d : \mathbf{t})$ be another three–valued interpretation. The meet of v and w gives us a new interpretation $w' = \{a : \mathbf{u}, b : \mathbf{u}, c : \mathbf{f}, d : \mathbf{u})$: as the assignments of a, b and d differ between v and w, the resulting value is $\mathbf{u}$. On the other hand, c is in both cases $\mathbf{f}$ and thus retains its value.

We will use v^x to denote a set of arguments mapped to x by v, where x a given truth–value.

Given an acceptance condition C_s for some argument $s \in S$ and an interpretation v, we define a shorthand $v(C_s)$ as $C_s(v^{\mathbf{t}} \cap par(s))$. For a given propositional formula φ and an interpretation v defined over all of the atoms of the formula, $v(\varphi)$ will just stand for the value of the formula under v. However, apart from knowing the "current" value of an acceptance condition for some interpretation, we would also like to know if this interpretation is "final". By this we understand that no new information will cause the value to change. This is captured by the notion of decisiveness, which are at the core of the extension–based ADF semantics.

Definition 4.4. *Given a two–valued interpretation v defined over a set $A \subseteq S$ of arguments, a **completion** of v to a set Z where $A \subseteq Z$ is an interpretation v' defined on Z in a way that $\forall a \in A\; v(a) = v'(a)$. By a $\mathbf{t}/\mathbf{f}$ completion we will understand v' that maps all arguments in $Z \setminus A$ respectively to $\mathbf{t}/\mathbf{f}$.*

Remark. We would like to draw the attention to the similarity between the concepts of completion and extending interpretation. Basically, given a three–valued interpretation v defined over S, the set $[v]_2$ precisely corresponds to the set of completions to S of the two–valued part of v. However, if we used the notion of an extension instead of a completion in a two–valued setting, it could be easily mistaken for the extension understood as set of arguments, not as an interpretation. Therefore, we will use our notation to avoid such collisions.

Definition 4.5. *We say that v is **decisive** for an argument $s \in S$ iff for any two (respectively two or three–valued) completions $v_{par(s)}$ and $v'_{par(s)}$ of v to $A \cup par(s)$, it holds that $v_{par(s)}(C_s) = v'_{par(s)}(C_s)$. We say that s is **decisively out/in** wrt v if v is decisive and all of its completions evaluate C_s to respectively out, in.*

Example 4.6. *Let $(\{a, b, c, d\}, \{a : b \to d, b : a \wedge c, c : \bot, d : d\})$ be the ADF depicted in Figure 2. Example of a decisively in interpretation for a is $v = \{b : \mathbf{f}\}$. It simply means that knowing that b is false, not matter the value of d, the implication is always true and thus the acceptance condition is satisfied. From the more technical side, it is the same as checking that both completions to*

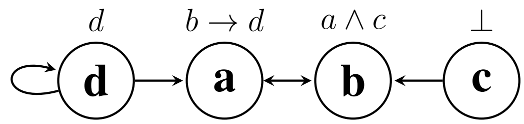


Figure 2: Sample ADF

$\{b, d\}$, namely $\{b : \mathbf{f}, d : \mathbf{t}\}$ and $\{b : \mathbf{f}, d : \mathbf{f}\}$ satisfy the condition. Example of a decisively out interpretation for b is $v' = \{c : \mathbf{f}\}$. Again, it suffices to falsify one element of a conjunction to know that the whole formula will evaluate to false.

Remark. Please note that the existence of an interpretation that satisfies the acceptance condition of an argument a (i.e. there is a set of parents s.t. condition is in) implies the existence of a decisively in interpretation for a and vice versa. Moreover, if an argument is decisively out/in w.r.t. an interpretation, it holds that its acceptance condition is out/in. It basically results from the definition of a completion and decisiveness. Finally, if an argument is decisively in/out w.r.t. some interpretation, then it is decisively out w.r.t. any of its completions, not necessarily the ones that are defined for all parents.

Please note that although decisiveness in the interpretation form is more convenient for our purposes, the set version of this idea was already developed in the original paper [13] for the grounded semantics. Thus, one can choose between the representations depending on which one is more suitable. The set of arguments that was decisively in/out w.r.t. some set of accepted (A) and rejected (R) arguments was retrieved via the acc/reb functions:

Definition 4.7. *Let $A, R \subseteq S$. Then $acc(A, R) = \{r \in S \mid A \subseteq S' \subseteq (S \backslash R) \Rightarrow C_r(S' \cap par(s)) = in\}$ and $reb(A, R) = \{r \in S \mid A \subseteq S' \subseteq (S \backslash R) \Rightarrow C_r(S' \cap par(s)) = out\}$.*

We will now show that the set and interpretation approaches represent the same concept. Since we are interested in extensions, i.e. single status assignments to arguments, we can assume that $A \cap R = \emptyset$. Then we have that an argument $r \in S$ is in $acc(A, R)$, if for all possible subsets of arguments that contain the accepted ones (A) and not including any of the rejected ones (thus are from $S \setminus R$) the acceptance condition is met. This is precisely checking if an argument is decisively in w.r.t. an interpretation v, where $v^{\mathbf{t}} = A$ and $v^{\mathbf{f}} = R$. Clearly, $reb(A, R)$ is just finding arguments that are decisively out w.r.t. v. We will come back to this representation when recalling the ADF grounded semantics.

4.2 Acyclicity

We will now explain how to check for positive dependency cycles. Please note we refrain from calling it support cycles in this context in order not to confuse it with certain definitions of support (evidential, necessary etc.) studied in other generalizations of the Dung's framework that we have recalled in Section 3.

The informal understanding of a cycle is simply whether acceptance of an argument depends on this argument. A natural way to analyze this situation would be to "track" the evaluation of a given argument, e.g. in order to accept a we need to accept b, to accept b we need to accept c and so on. This simple case becomes more complicated when disjunction is introduced. We then receive a number of such "paths", some of them ending with cycles, some not. Moreover, they might be conflicting with each other, and we can have a situation where all acyclic evaluations are attacked and a cycle is forced. Our idea is to "unwind" the arguments and construct such paths for them along with storing the arguments that can conflict it.

Let us now introduce the formal definitions. Given an argument $s \in S$ and $x \in \{in, out\}$, by $min\_dec(x, s)$ we will the denote the set of minimal two–valued interpretations that are decisively x for s. By minimal we understand that both $v^{\mathbf{t}}$ and $v^{\mathbf{f}}$ are minimal w.r.t. set inclusion.

Definition 4.8. *Let $A \subseteq S$ be a nonempty set of arguments. A* ***positive dependency function*** *on A is a function pd assigning every argument $a \in A$ an interpretation $v \in min\_dec(in, a)$ s.t. $v^t \subseteq A$ or $\mathcal{N}$ for null iff no such interpretation can be found.*

Definition 4.9. *An* ***acyclic positive dependency evaluation*** *ace^a for $a \in A$ based on a given pd–function pd is a pair $((a_0, ..., a_n), B)$,* [7] *where $B = \bigcup_{i=0}^{n} pd(a_i)^{\mathbf{f}}$ and $(a_0, ..., a_n)$ is a sequence of distinct elements of A s.t.: 1) $\forall_{i=0}^{n} pd(a_i) \neq \mathcal{N}$, 2) $a_n = a$, 3) $pd(a_0)^{\mathbf{t}} = \emptyset$, and 4) $\forall_{i=1}^{n}$, $pd(a_i)^{\mathbf{t}} \subseteq \{a_0, ..., a_{i-1}\}$. We will refer to the sequence part of the evaluation as* ***pd–sequence*** *and to the B as the* ***blocking set****.*

We will say that an argument a is **pd–acyclic** in A iff there exist a pd–function on A and a corresponding acyclic pd–evaluation for a. Furthermore, we will simply write than an argument has an acyclic pd–evaluation on A if there is some pd–function on A from which we can produce the evaluation. There are two ways we can "attack" an acyclic evaluation. Either we accept an argument that needs to be rejected in order for the evaluation to hold (i.e. it is in the blocking set), or we are able to discard an argument from the pd–sequence. This leads to the following, more abstract formulation:

Definition 4.10. *Let $A \subseteq S$ be some set of arguments and $a \in A$ s.t. a has an acyclic pd–evaluation $ace^a = ((a_0, ..., a_n), B)$ in A. We say that a two–valued interpretation v* ***blocks*** *ace^a iff $\exists b \in B$ s.t. $v(b) = \mathbf{t}$ or $\exists a_i \in \{a_0, ..., a_n\}$ s.t. $v(a_i) = \mathbf{f}$.*

Remark. The idea of a pd–evaluation is strongly related to the concept of a powerful sequence from AFNs. The difference lies in the fact that in AFNs we have binary support, thus blocking a sequence corresponds precisely to attacking its members. Since ADFs are not limited in this way, preventing a sequence might not always break conflict–freeness, and hence the blocking set needs to be stored.

Remark. A pd–evaluation can be self–blocking, i.e. some members of the pd–sequence are present in the blocking set. Although an evaluation like that will never be accepted in an extension, it can make a difference in what we consider a valid attacker.

[7] Please note that it is not required that $B \subseteq A$

Let us now show on an example why we require minimality on the chosen interpretations and why do we store the blocking set:

Example 4.11. *Let us assume an ADF* $(\{a, b, c, d\}, \{a : \top, b : c \vee \neg a, c : b \vee \neg d, d : \top\})$*, as depicted in Figure 3. For the argument* b *there exist the following decisively in interpretations:* $v_1 = \{a : \mathbf{f}\}, v_2 = \{c : \mathbf{t}\}, v_3 = \{a : \mathbf{f}, c : \mathbf{f}\}, v_4 = \{a : \mathbf{f}, c : \mathbf{t}\}, v_5 = \{a : \mathbf{t}, c : \mathbf{t}\}$*. Only the first two are minimal. Considering* v_5 *would give us a wrong view that* b *depends positively on* c*, which is not a desirable reading. The minimal ones for* c *are* $w_1 = \{b : \mathbf{t}\}$ *and* $w_2 = \{d : \mathbf{f}\}$*. Since* a *and* d *are initial arguments, a minimal decisively in interpretation for them is naturally empty* $z = \{\}$*.*

There are four pd–functions on defined on $\{a, b, c, d\}$*.* $pd_1 = \{a : z, b : v_1, c : w_1, d : z\}$*,* $pd_2 = \{a : z, b : v_1, c : w_2, d : z\}$*,* $pd_3 = \{a : z, b : v_2, c : w_1, d : z\}$ *and* $pd_4 = \{a : z, b : v_2, c : w_2, d : z\}$*. For* b *we obtain the following acyclic pd–evaluations:* $((b), \{a\})$ *(from* pd_1 *and* pd_2*) and* $((c, b), \{d\})$ *(from* pd_4*). Consequently, for* c *we receive* $((c), \{d\})$ *and* $((b, c), \{a\})$*. It is easy to see we cannot obtain any acyclic evaluations for* b *and* c *from* pd_3*.*

Let us now see if $\{a, b, c, d\}$ *is acyclic. The intuition is, that it is not – the presence of* a *and* d *"forces" a cycle between* b *and* c*. The acceptance conditions of all arguments are satisfied, thus this simple check is not good enough. If we were to follow the powerful approach from AFNs, we would see that all members of the pd–sequences are there. Hence, this way also does not suffice. Looking at the whole evaluations shows us that members of the blocking sets are accepted and only now we get the correct answer that the set contains arguments that are not acyclic.*

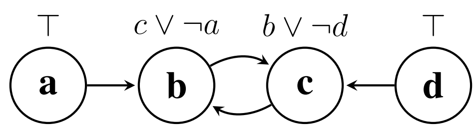


Figure 3: Sample ADF

We can now proceed to recall existing and introduce new semantics of the abstract dialectical frameworks.

5 Extension–Based Semantics of ADFs

Although various semantics for ADFs have already been defined in the original paper [13], only three of them – conflict–free, model and grounded (initially referred to as well–founded) – are still used (issues with the other formulations can be found in [11, 35, 38]). Moreover, the treatment of cycles and their handling by the semantics was not sufficiently developed. In this section we will address all of those issues. Before we continue, let us first motivate our choice on how to treat cycles. As we have shown in Section 3, the opinions on support cycles differ between the available frameworks. There is no consensus as to how they should be treated, as we can find examples both for and against their validity. Therefore, we would like to explore the possible approaches in the context of ADFs by developing appropriate semantics.

The classification of the sub–semantics that we will adopt in this paper is as follows. Bearing in mind the intuition we have presented in the introduction, appropriate semantics will receive an $xy-$ prefix, where $x, y \in \{a, c\}$. It will denote whether we demand acyclicity - a or not - c - on the "inside" (x) and on the "outside" (y). As the conflict–free (and naive) semantics focus only on what we can accept, we will drop the prefixing in this case. Although the model, stable and grounded fit into our classification (more details can be found in Section 6), they have a sufficiently unique naming and further annotations are not necessary. We are thus left with admissible, preferred and complete. The BAF approach follows the idea that we can accept arguments that are not acyclic in our opinion and we allow our opponent to do the same. The ADF semantics we have developed in [35] also shares this view. Therefore, they will receive the $cc-$ prefix. On the other hand, AFN and EAS semantics do not permit cycles both in extensions and as attackers. Consequently, the semantics following this line of reasoning wll be prefixed with $aa-$.

Please note we believe that also a non–uniform approach can be suitable for certain situations. By a non–uniform we mean not accepting cyclic arguments, but still treating them as valid attackers and so on (i.e. $ca-$ and $ac-$). Imagine a case with a suspect, prosecutor and a jury. The suspect can utter a self–supporting argument such as "I'm telling the truth!", which expressed properly can convince the jury and raise doubt. The prosecutor has to disprove the suspect's claim with sufficient evidence and a clear, acyclic chain of reasoning. Depending on whom we identify with, the requirements shift and hence we can have semantics that allow cycles on the "inside", but not on the "outside", and vice versa. Following this line of thought we introduce both uniform and non–uniform sub–semantics when required.

Remark. Please note that such non–uniform approaches can also be found in logic programming, one can for example compare the supported and stable models.

5.1 Conflict–free and naive semantics

In the Dung setting, conflict–freeness meant that the elements of an extension could not attack one another. This is also the common interpretation in various other AF generalizations, including the bipolar ones such as AFNs and EASs [32, 34]. Providing an argument with the required support is then a separate condition. In ADFs, where we lose the set representation of relations in favor of abstraction, not including "attackers" and accepting "supporters" is combined into one notion. This represents the intuition of "arguments that can stand together" presented in [5].

Definition 5.1. *A set of arguments $E \subseteq S$ is a* ***conflict–free extension*** *of D if for all $s \in E$ we have $C_s(E \cap par(s)) = in$.*

The acyclic version of conflict–freeness is a bit more than just a pd–acyclic set; we have to make sure that the evaluation is unblocked. To meet the formal requirements, we first have to show how the notions of range and the $^+$ set are moved to ADFs.

Definition 5.2. *Let $E \subseteq S$ a conflict–free extension of D and v_E a partial two–valued interpretation built as follows:*

1. *let $M = E$ and for every $a \in E$ set $v_E(a) = \mathbf{t}$;*

2. *for every argument $b \in S \setminus M$ that is decisively out in v_E, set $v_E(b) = \mathbf{f}$ and add b to M;*
3. *now repeat the previous step until there are no new elements added to M.*

*By E^+ we understand the set of arguments $v_E^{\mathbf{f}}$ and we will refer to it as the **discarded set**. v_E now forms a **range interpretation** of E, where the usual range is denoted as E^R and equals $E \cup E^+$*[8].

However, the notions of the discarded set and the range are quite strong in the sense that they require an explicit "attack" on arguments that take part in dependency cycles. This is not always a desirable property. Depending on the approach we might not treat cyclic arguments as valid and hence want them "out of the way".

Definition 5.3. *Let $E \subseteq S$ a conflict–free extension of D and v_E^a a partial two–valued interpretation built as follows:*

1. *Let $M = E$. For every $a \in M$ set $v_E^a(a) = \mathbf{t}$.*
2. *For every argument $b \in S \setminus M$ s.t. every acyclic pd–evaluation of b in S is blocked by v_E^a, set $v_E^a(b) = \mathbf{f}$ and add b to M.*
3. *Repeat the previous step until there are no new elements added to M.*

*By E^{a+} we understand the set of arguments mapped to $\mathbf{f}$ by v_E^a and refer to it as **acyclic discarded set**. We refer to v_E^a as **acyclic range interpretation** of E.*

It is easy to see that there is a subset relation between the two versions of the discarded set:

Lemma 5.4. *Let E be a conflict–free set. Then $E^+ \subseteq E^{a+}$.*

Proof Idea. Let us look at the construction of the standard and acyclic range interpretations v_E and v_E^a. In the first step, they are the same, i.e. contain only $\mathbf{t}$ assignments for the elements of E. If an interpretation has the power to decisively out the argument, then of course it "blocks" any in (and thus decisively in) interpretations of this argument. Hence, any acyclic pd–evaluation of this argument and ones built with it can be easily prevented. Thus, it is easy to see that at every step of the construction the acyclic range interpretation has in total falsified at least as many arguments as the standard one, i.e.$v_E^{\mathbf{f}} \subseteq v_E^a$. Consequently, $E^+ \subseteq E^{a+}$.

With this at hand, we can now define an acyclic version of conflict–freeness:

Definition 5.5. *A conflict–free extension E is a **pd–acyclic conflict–free extension** of D iff for every argument $a \in E$, there exists an unblocked pd–acyclic evaluation on E w.r.t. v^E.*

Remark. As we are dealing with a conflict– free extension, all the arguments of a given pd–sequence are naturally $\mathbf{t}$ both in v_E and v_E^a. Therefore, in order to ensure that an evaluation is unblocked it suffices to check whether $E \cap B = \emptyset$. Consequently, in this case it does not matter w.r.t. which version of range we are verifying the evaluations.

[8] It can be equivalently seen as $v^{\mathbf{t}} \cup v^{\mathbf{f}}$ or simply as the set of arguments for which v_E is defined.

Definition 5.6. *The **naive** and **pd–acyclic naive** extensions are defined as respectively maximal w.r.t. set inclusion conflict–free and pd–acyclic conflict–free extensions.*

Example 5.7. *Let us now look at the ADF* $(\{a, b, c\}, \{a : \neg c \vee b, b : a, c : c\})$ *depicted in Figure 4. The conflict–free extensions are* $\emptyset, \{a\}, \{c\}, \{a, b\}$ *and* $\{a, b, c\}$*. Since there exists no acyclic evaluation for* c*, it cannot appear in any pd–acyclic conflict–free extension. Thus, only* $\emptyset, \{a\}$ *and* $\{a, b\}$ *qualify for acyclic type. The naive and pd–acyclic naive extensions are respectively* $\{a, b, c\}$ *and* $\{a, b\}$*.*

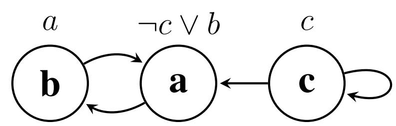


Figure 4: Sample ADF

5.2 Model and stable semantics

The concept of a model basically follows the intuition that if something can be accepted, it should be accepted:

Definition 5.8. *A conflict–free extension* E *is a **model** of* D *if* $\forall\ s \in S, \quad C_s(E \cap par(s)) = in$ *implies* $s \in E$*.*

Although this definition is simple, several of its properties should be explained. First of all, given some model candidate E, checking whether a condition of some argument s is satisfied does not verify if an argument depends on itself or if it ”outs” a previously included member of E. This means that an argument we should include may break conflict–freeness of the set. On the other hand, an argument that is not in w.r.t. E, can due to dependency cycles appear in a model $E \subset E'$. Consequently, it is clear to see that model semantics is not universally defined and the produced extensions might not be maximal w.r.t. subset inclusion. Finally, we would like to make a note concerning the arguments that are not included in a model. We can see that they were either inconsistent, ”attacked” by the set, or they could not be accepted as at least one argument necessary for their acceptance was missing. Especially the latter is interesting; lack of support means two things – either we were able to trace back to an inconsistent or attacked argument, or we reached a positive dependency cycle. Looking at the model semantics from the ”defense” perspective, we are either able to attack (or cut off the support) of our attacker, or the attacker is not valid due to a positive dependency cycle. This description clearly follows the idea of $ca-$ semantics; as we show in Lemma 6.9, this is indeed the case.

The model semantics was used as a mean to obtain the stable models. The main idea was to make sure that the model is acyclic. Unfortunately, the used reduction method was not adequate, as shown in [11]. However, the initial idea still holds, and the new stability is defined as follows:

Definition 5.9. *A model E is a* ***stable extension*** *iff it is pd–acyclic conflict–free.*

Although the produced extensions are now incomparable w.r.t. set inclusion, stability is still not universally defined.

Example 5.10. *Let us come back to the ADF $(\{a, b, c\}, \{a : \neg c \vee b, b : a, c : c\})$ depicted in Figure 4 and described in Example 5.7. The conflict–free extensions were $\emptyset, \{a\}, \{c\}, \{a, b\}$ and $\{a, b, c\}$. The first two are not models. It is easy to see that respectively $C_a(\emptyset) = in$ and $C_b(\{a\}) = in$, hence the model condition is not satisfied. Recall that $\emptyset, \{a\}$ and $\{a, b\}$ were the pd–acyclic conflict–free extensions. The only one that is also a model is $\{a, b\}$ and thus we obtain our single stable extension.*

5.3 Grounded semantics

Just like in the Dung setting, the grounded semantics preserves the unique–status property. Moreover, it is defined in the terms of a special operator:

Definition 5.11. *Let $\Gamma'_D(A, R) = (acc(A, R), reb(A, R))$, where $acc(A, R) = \{r \in S \mid A \subseteq S' \subseteq (S \backslash R) \Rightarrow C_r(S' \cap par(s)) = in\}$ and $reb(A, R) = \{r \in S \mid A \subseteq S' \subseteq (S \backslash R) \Rightarrow C_r(S' \cap par(s)) = out\}$. Then E is the* ***grounded model*** *of D iff for some $E' \subseteq S$, (E, E') is the least fix–point of Γ'_D.*

As we have explained in Section 4.1, acc and reb are nothing more than means of retrieving decisively in/out arguments via a set representation. We are now interested in the least fixpoint of the operator, which as noted in [13] can be reached by iterating Γ'_D starting with $(A, R) = (\emptyset, \emptyset)$. It is easy to see that at all steps $A \cap R = \emptyset$: as the sets are initially disjoint, we can see it as an interpretation, an clearly no argument can be at the same time decisively in and out w.r.t. this interpretation. Therefore, we propose an alternative way to compute the grounded extension, in line with Proposition 2.5:

Proposition 5.12. *Let v be an empty interpretation. For every argument $a \in S$ that is decisively in w.r.t. v, set $v(a) = \mathbf{t}$ and for every argument $b \in S$ that is decisively w.r.t. v, set $v(b) = \mathbf{f}$. Repeat the procedure until no further assignments can be done. The* ***grounded extension*** *of D is then $v^{\mathbf{t}}$.*

Remark. The grounded semantics follows the ac–approach. The extension is iteratively built from the initial data and thus is acyclic by nature. Moreover, the way arguments are rejected follows the standard way of discarding.

Example 5.13. *Let us come back again to the ADF $(\{a, b, c\}, \{a : \neg c \vee b, b : a, c : c\})$ depicted in Figure 4. We will now try to find its grounded extension. Let v be an empty interpretation. The only argument that has a satisfied acceptance condition, and thus the chance to be decisively in, is a. However, it is easy to see that if we accept c, the condition is outed. Hence, we obtain no decisiveness in this case. Since b and c are both out, we can check if they have a chance to be decisively out. Again, condition of b can be met if we accept a, and condition of c if we accept c;*

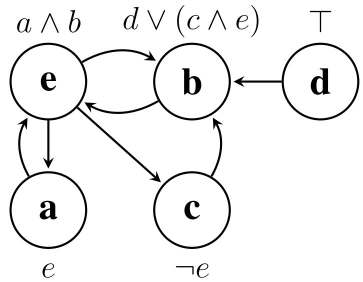


Figure 5: Sample ADF

as v does not define the status of a and c, we obtain no decisiveness again. Thus, $\emptyset$ is the grounded extension.

Let us now look at the ADF $(\{a, b, c, d, e\}, \{a : e, b : d \vee (c \wedge e), c : \neg e, d : \top, e : a \wedge b\})$ depicted in Figure 5. Assume an empty interpretation v. It is easy to see that only d is decisively in w.r.t. v and that there are no decisively out arguments. However, now that we have $d : \mathbf{t}$ assignment, b can be also decisively assumed. Again, no decisive outing occurs, and next round returns us no new assignments. Thus, the grounded extension is $\{b, d\}$.

5.4 Admissible and preferred semantics

The basic admissible semantics was developed in [35]. It basically followed the intuition that we need to be able to discard any counterarguments of our opponent and made no acyclicity assumptions:

Definition 5.14. *Deprecated: A conflict–free extension $E \subseteq S$ is* ***admissible*** *in D iff for any nonempty $F \subseteq S \setminus E$, if there exists an $a \in E$ s.t. $C_e(par(e) \cap (F \cup E)) = out$ then $F \cap E^+ \neq \emptyset$.*[9]

The new simplified version of the previous formulation, taking into account our classification, is now as follows:

Definition 5.15. *A conflict–extension $E \subseteq S$ is* ***cc–admissible*** *in D iff every element of E is decisively in w.r.t to its range interpretation v_E.*

It is important to understand how decisiveness encapsulates the defense known from the Dung setting. If an argument is decisively in, then basically any set of arguments that would have the power to out the acceptance condition is "prevented" by the interpretation. Hence, statements required for the acceptance of a are mapped to $\mathbf{t}$ and those that would make us reject a are mapped to $\mathbf{f}$. The former encapsulates the required support, while the latter contains the "attackers" known from the Dung setting.

[9]The new formulation is equivalent to this one and we see it as more elegant. However, we would like to recall this version to avoid confusion for readers familiar with our previous works.

Definition 5.16. *A conflict–free extension E is **ca–admissible** iff every argument in E is decisively in w.r.t. acyclic range interpretation v_E^a.*

When working with the semantics that have to be acyclic on the "inside", we not only have to defend the members, but also their acyclic evaluations:

Definition 5.17. *A ca(cc)–admissible extension E is **aa(ac)–admissible** iff it is pd–acyclic conflict–free and for every member of the extension there exists an acyclic pd–evaluation on E s.t. all members of its blocking set B are mapped to $\mathbf{f}$ by the acyclic (standard) range interpretation of E.*

The following example shows that decisiveness encapsulates defense of an argument, but not necessarily of its evaluation:

Example 5.18. *Recall the framework depicted in Figure 3 and described in Example 4.11, i.e. $(\{a, b, c, d\}, \{a : \top, b : c \vee \neg a, c : b \vee \neg d, d : \top\})$. We can see that $\{b\}$ is a pd–acyclic conflict–free extension. Its range interpretation is just $v = \{b : \mathbf{t}\}$ (both standard and acyclic). It is easy to see that b is not decisively in w.r.t. v; a completion $v' = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{f}\}$ falsifies the acceptance condition. Thus, as expected, it cannot be admissible. Let us now look at the set $\{b, c\}$. Again, it is pd–acyclic conflict–free; its range is simply $v = \{b : \mathbf{t}, c : \mathbf{t}\}$. Both arguments are decisively in w.r.t v; whether we utter a, d or both, it will not change the outcomes of the acceptance conditions. However, it is easy to see that if our opponent uses $\{a, d\}$, the arguments are still able to stand only due to a cyclic dependency. In a more technical way, given possible evaluations $((b), \{a\}), ((c, b), \{d\}), ((c), \{d\}), \{(b, c), \{a\})$ none of them is "defended", i.e. no blocking set is falsified by the range interpretation.*

Definition 5.19. *A set of arguments is **xy–preferred** iff it is maximal w.r.t. set inclusion xy–admissible.*

Please note that, out of all the sub–semantics, ca–admissible behaves slightly differently from the others. By this we mean that an argument discarded by the acyclic range interpretation can be in fact decisively in w.r.t. it unless the accepted arguments are acyclic. We case of such behavior is visible in Example 5.20.

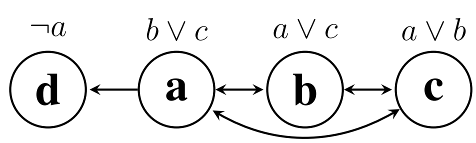


Figure 6: Sample ADF

Example 5.20. *Let us assume an ADF $(\{a, b, c, d\}$, $\{a : b \vee c,\ b : a \vee c,\ c : a \vee b, d : \neg a\})$ as depicted in Figure 6. $\emptyset$ is trivially admissible. A set $\{d\}$ is ca–admissible, as its attacker a has no acyclic pd–evaluation and is thus not treated as valid. It is easy to see that sets $\{a, b\}$, $\{a, c\}$,*

$\{b, c\}$ and $\{a, b, c\}$ are also ca–admissible, since cycles on the "inside" are permitted and none of the members is attacked in the first place. The last extension is $\{b, c, d\}$. Although the condition of a is of course satisfied, it still has no acyclic evaluation and is automatically discarded. Please note that the fact that the condition is met comes from the fact that the part of the cycle it depends from is accepted in the extension.

This behavior can be interpreted as not allowing the opponent to use cyclic reasoning, even if his opinions would be based on ours. In this sense, the method is very strict. A more refined ca–approach that would not have this opinion "isolation" consequence is left for future work – we believe it can be achieved by analyzing the acyclic evaluations that "end" not in unsupported arguments, but in ones present in the extension.

Example 5.21. *Let us recall the ADF $(\{a, b, c, d, e\}, \{a : e, b : d \vee (c \wedge e), c : \neg e, d : \top, e : a \wedge b\})$ depicted in Figure 5. $\emptyset, \{c\}, \{d\}, \{b, d\}, \{c, d\}, \{b, c, d\}$ and $\{a, b, d, e\}$ are the conflict–free extensions, with the acyclic ones being $\emptyset, \{c\}, \{d\}, \{b, d\}, \{c, d\}$ and $\{b, c, d\}$.*

The obvious cc–admissible extensions are $\emptyset, \{d\}$ and $\{b, d\}$ (follows from reasoning on the grounded extension in Example 5.13). The presence of d makes b acceptable independently of what happens to c and e, thus we do not have to analyze the conflict between them in this context. The last cc–admissible extension is $\{a, b, d, e\}$ and again, since d is present, the conflict can be disregarded. This is also the only cc–admissible extension that is not ac–admissible.

Let us now move to semantics acyclic on the "outside", starting with the ca approach. $\emptyset$, $\{d\}$ and $\{b, d\}$ naturally carry over, and so does $\{a, b, d, e\}$. However, now that we do not have to defend from cyclic attackers, $\{c\}$ comes into play. Since a and e have no acyclic pd–evaluation, they land in the discarded range of any set (assuming they are not accepted into the set beforehand). Thus, the $((c), \{e\})$ pd–evaluation of c is properly defended. Therefore, we have that $\{c\}, \{b, c, d\}$ and $\{c, d\}$ are also ca–admissible extensions. With the exception of $\{a, b, d, e\}$, all of those extensions are also aa–admissible.

The preferred extensions are $\{a, b, d, e\}$ for the cc aproach, $\{b, d\}$ for ac, $\{b, c, d\}$ for aa and finally $\{b, c, d\}$ and $\{a, b, d, e\}$ for the ca type.

To summarize, we have presented four types of admissibility and explained the motivation behind the definitions. The further properties of this semantics, such as whether it satisfies the Fundamental Lemma and how sub–semantics relate one to another, will be given in Section 6.

5.5 Complete semantics

Quoting Dung, "the notion of complete extensions captures the kind of confident rational agent who believes in every thing he can defend". When we move to a bipolar setting, completeness is understood as an approach in which we are required to accept an argument that we sufficiently support and are capable of defending. This represents an agent that has to believe anything he can conclude from his opinions. In a sense, it follows the model intuition that whatever we can accept, we should accept – however, here we are faced with a stronger version. Not only we use an admissible base instead of a conflict–free one, but also, due to defense, the added argument is obviously conflict–free both with itself and with the rest of the extension. This leads to a formulation where instead of checking if an argument is in, we want it to be decisively in.

Definition 5.22. *A cc(ac)–admissible extension E is **cc(ac)–complete** in D iff every argument in S that is decisively in w.r.t. the range interpretation of E is in E*[10].

Definition 5.23. *An aa–admissible extension E is **aa–complete** in D iff every argument in S that is decisively in w.r.t. the acyclic range interpretation v_E^a is in E*[11].

Due to the problems mentioned before, the ca version of the semantics is slightly different than the others:

Definition 5.24. *A ca–admissible extension E is **ca–complete** in D iff every argument $s \in S \setminus E^{a+}$ that is decisively in w.r.t. the acyclic range interpretation of v_E^a is in E.*

Example 5.25. *Recall the ADF $(\{a, b, c, d\}, \{a : b \vee c,\ b : a \vee c,\ c : a \vee b, d : \neg a\})$, depicted in Figure 6 and described in Example 5.20. The produced ca–admissible extensions were $\emptyset$, $\{d\}$,$\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, b, c\}$ and $\{b, c, d\}$. $\emptyset$ has an acyclic discarded set $\{a, b, c\}$, thus making d decisively in. Consequently, it cannot be complete. Similar follows for $\{b, c\}$ – acyclic discarded set is $\{a\}$ and we could have accepted d. $\{a, b\}$ and $\{a, c\}$ discard respectively $\{c, d\}$ and $\{b, d\}$. Therefore, although c in the first and b in the latter case are in fact decisively in, both sets are ca–complete. Same follows for $\{b, c, d\}$ – condition of a is satisfied, but it is contained in the discarded set, and thus completeness requirements are met.*

Example 5.26. *We will now show the extensions of all of the semantics and their sub–semantics on an example. Let $(\{a, b, c, d\}, \{a : \neg b, b : \neg a, c : b \wedge \neg d, d : d\})$ be an ADF, as depicted in Figure 7. Its possible extensions are listed in Table 1.*

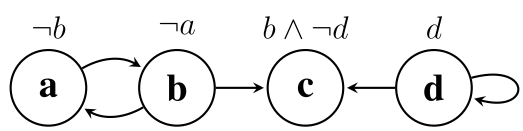


Figure 7: Sample ADF

6 Properties of Extension–Based Semantics

Various properties can be proved for our semantics and sub–semantics, obviously the study we provide here will not cover all of them. However, we will show how all sub–semantics of a given type relate one to another as well as recall the lemmas and theorems from the original paper on AFs [23]. Before we continue, we will make a note on some basic properties of the range interpretations:

[10]Please consult Lemma 6.4 to see that no further ”defense” of acyclicity in case of ac–completeness is required.
[11]Please consult Lemma 6.4 to see that no further ”defense” of acyclicity is required.

Table 1: Extensions of the ADF from Figure 7.

CF	C	$\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$
	A	$\emptyset$, $\{a\}$, $\{b\}$, ~~$\{d\}$~~, $\{b,c\}$, ~~$\{a,d\}$, $\{b,d\}$~~
MOD		~~$\emptyset$~~, $\{a\}$, ~~$\{b\}$, $\{d\}$~~, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$
STB		~~$\emptyset$~~, $\{a\}$, ~~$\{b\}$, $\{d\}$~~, $\{b,c\}$, ~~$\{a,d\}$, $\{b,d\}$~~
GRD		$\emptyset$, ~~$\{a\}$, $\{b\}$, $\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$~~
ADM	CC	$\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$, ~~$\{b,c\}$~~, $\{a,d\}$, $\{b,d\}$
	AC	$\emptyset$, $\{a\}$, $\{b\}$, ~~$\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$~~
	CA	$\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$
	AA	$\emptyset$, $\{a\}$, $\{b\}$, ~~$\{d\}$~~, $\{b,c\}$, ~~$\{a,d\}$, $\{b,d\}$~~
COMP	CC	$\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$, ~~$\{b,c\}$~~, $\{a,d\}$, $\{b,d\}$
	AC	$\emptyset$, $\{a\}$, $\{b\}$, ~~$\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$~~
	CA	$\emptyset$, $\{a\}$, ~~$\{b\}$~~, $\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$
	AA	$\emptyset$, $\{a\}$, ~~$\{b\}$, $\{d\}$~~, $\{b,c\}$, ~~$\{a,d\}$, $\{b,d\}$~~
PREF	CC	~~$\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$, $\{b,c\}$~~, $\{a,d\}$, $\{b,d\}$
	AC	~~$\emptyset$~~, $\{a\}$, $\{b\}$, ~~$\{d\}$, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$~~
	CA	~~$\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$~~, $\{b,c\}$, $\{a,d\}$, $\{b,d\}$
	AA	~~$\emptyset$~~, $\{a\}$, ~~$\{b\}$, $\{d\}$~~, $\{b,c\}$, ~~$\{a,d\}$, $\{b,d\}$~~

Proposition 6.1. *Let E be a standard and A a pd–acyclic conflict–free extension, with v_E, v^a_E, v_A and v^a_A their corresponding standard and acyclic range interpretations. Let $s \in S$ be an argument. The following holds:*

1. *If $v_E(s) = \mathbf{f}$, then $C_s(E \cap par(s)) = out$. It does not hold for v^a_E.*
2. *If $v_A(s) = \mathbf{f}$, then $C_s(A \cap par(s)) = out$. Same holds for v^a_A.*
3. *If $v^a_A(s) = \mathbf{f}$, then s is decisively out w.r.t. v^a_A.*

Proof Sketch.

1. Since the arguments mapped to $\mathbf{f}$ are decisively out w.r.t. v_E, then the acceptance condition is obviously out. The fact that it does not hold for the acyclic version can be already noted in Example 5.20.

2. The property of v_A is obvious by construction, the one for v^a_A follows from Point 3 (if something is decisively out, then it is out).

3. Assume that $v^a_A(s) = \mathbf{f}$, but s is not decisively out w.r.t. v^a_A. This means there exists a completion to $A \cup A^{a+} \cup par(s)$ of the acyclic range interpretation v' s.t. $C_s(v'^{\mathbf{t}}) = in$. As it is defined for all parents of s, s is decisively in w.r.t. it, and thus we can extract a minimal interpretation $v'_m in$. As the existence of v' was not "prevented", neither is $v'_m in$ – this basically means that no element of $v'^{\mathbf{t}}$ was falsified and no element of $v'^{\mathbf{f}}$ was mapped

to $\mathbf{t}$. Consequently, we either have that $v'^{\mathbf{t}} \subseteq A$ or that there exists an argument $x \in v'^{\mathbf{t}}$ for which v^a_A is undefined. If it is the first case, then all elements of $v'^{\mathbf{t}}$ have an unblocked pd–acyclic evaluation. Therefore, we can recombine their pd–sequences and blocking sets in a way we have a pd–acyclic evaluation that is not blocked by v^a_A and we reach a contradiction[12]. If it is the latter, then it means that x possessed and unblocked pd–acyclic evaluation and again we could have constructed one for s that was not blocked by v^a_A.

6.1 Admissible and preferred semantics

Let us now show the relations between the introduced admissible sub–semantics.

Lemma 6.2. *The following holds:*

1. *Every ac–admissible extension is cc–admissible*
2. *Every aa–admissible extension is ca–admissible*
3. *Every cc–admissible extension is ca–admissible*
4. *Every ac–admissible extension is aa–admissible*

Proof.

1. Follows from the Definition 5.17 of admissible semantics.
2. As above.
3. Let E be a conflict–free extension, v_E and v^a_E its standard and range interpretations. By Lemma 5.4, we know that $E^+ \subseteq E^{a+}$, i.e. that v^a_E is a completion of v_E to $E \cup E^{a+}$. Thus obviously if all arguments in E are decisively in w.r.t. v_E, they are also decisively in w.r.t. v^a_E. Thus, if E is cc–admissible, it is also ca–admissible.
4. Similar to 3.

□

Remark. The restrictions we put on the "inside" and "outside" affect the number of extensions we receive. The less we have on the inside, the more we can say. The more we have on the outside, the less our opponent is allowed to utter against us. Thus, not surprisingly, the ac approach can be seen as the most strict, while ca admits the most.

However, as it can be already observed in Example 5.26, the is–a relation between the extensions cannot be assumed in the case of the preferred sub–semantics. Although a given admissible extension of one type can also be of another, it does not mean their maximal elements are the same as well. Thus, we can only derive some inclusion relation, as depicted in Figure 8.

[12] Please note that it can obviously be the case that $A \cup \{s\}$ is no longer pd–acyclic conflict–free. However, we are performing the checks purely against A and its range interpretation, not the extended version.

Lemma 6.3. *Let xy and x'y' be two admissible sub–semantics, where $x, x', y, y' \in \{a, c\}$, s.t. every xy–admissible extension is also x'y'–admissible (see Lemma 6.2). Then every xy–preferred extension is contained in some x'y'–preferred extension.*

Proof Idea. The reasoning behind it is rather simple;. Given $x, x', y, y' \in \{a, c\}$, if xy–admissible extensions are x'y'–admissible, then also xy–preferred extensions are x'y'–admissible. Taking the maximal x'y'–admissible extensions, hence x'y'–preferred ones, ensures that every xy–preferred one is contained in at least one chosen set.

Before we continue with further analysis, we first have to show that our admissible sub–semantics satisfy the Fundamental Lemma. However, in the case of ca–admissibility, we can only assume a weaker version.

Lemma 6.4. *CC/AC/AA Fundamental Lemma: Let E be a cc(ac)–admissible extension, v_E its range interpretation and $a, b \in S$ two arguments decisively in w.r.t. v_E. Then $E' = E \cup \{a\}$ is cc(ac)–admissible and b is decisively in w.r.t. v'_E.*

Let E be an aa-admissible extension, v^a_E its acyclic range interpretation and $a, b \in S$ two arguments decisively in w.r.t. v^a_E. Then $E' = E \cup \{a\}$ is aa–admissible and b is decisively in w.r.t. v'_E.

Proof Sketch. Let us start with the cc case. First of all, it follows from Lemma 6.1 that neither a nor b could have been mapped to $\mathbf{f}$ by v_E. Thus, v'_E is a completion of v_E and whatever was decisively in w.r.t. v_E must remain this way w.r.t. v'_E. Consequently, all arguments in E' satisfy the cc–admissibility criterion. The same follows for b – if it was decisively in w.r.t. v_E, then it is also this way w.r.t. v'_E.

Let us move on to the ac approach. Since every ac–admissible is cc–admissible and we now know that the cc–semantics conforms the Fundamental Lemma, it holds that E' is at least cc–admissible. We also know that b remains decisively in. What remains is to show that E' preserves its acyclicity and defends the evaluations. We know that for every argument in E there is a "protected" evaluation and since a could not have been mapped to $\mathbf{f}$ by v_E, no conflict arises. Thus, we only have to make sure that a has an unblocked evaluation and that all elements of the blocking set of this evaluation are falsified by the range interpretation. As a is decisively in w.r.t. v_E, it means that there exists a minimal decisively in interpretation v' for a s.t. v_E (and thus v'_E) is its completion, i.e. $v'^{\mathbf{t}} \subseteq E$ and $v'^{\mathbf{f}} \subseteq E^+$. Since $v'^{\mathbf{t}} \subseteq E$, then for every element of v'^{tvt} there is a protected and unblocked acyclic evaluation. It is easy to see that we can combine their pd–sequences into one and append it with a. The union of their respective blocking sets and $v'^{\mathbf{f}}$ gives us the new blocking set. Thus, we have an appropriate evaluation that is clearly unblocked. Finally, since $v'^{\mathbf{f}} \subseteq E^+$, it is also "defended" and we can conclude that E' is ac–admissible.

Finally, we have the aa–admissible case. By Lemma 6.1 we know that neither a nor b could have been mapped to $\mathbf{f}$ by v^a_E. Thus, v'^a_E is a completion of v^a_E. Therefore, all arguments in E, a and b remain decisively in w.r.t. v'^a_E and E' is at least ca–admissible. What remains to show is that E' preserves its acyclicity and defends the evaluations, which follows the reasoning we presented above in the ac case.

Lemma 6.5. *Weak CA Fundamental Lemma: Let E be a ca–admissible extension, v^a_E its acyclic range interpretation and $a \in S \setminus E^{a+}$ an argument decisively in w.r.t. v_E. Then $E' = E \cup \{a\}$ is ca–admissible.*

Proof Sketch. As E may contain cycles, it can be the case that a is decisively in w.r.t. v^a_E and at the same time $v^a_E(a) = \mathbf{f}$. Therefore, we only take into account such arguments a that are not discarded. As a result, v'_E is a completion of v_E and we can use the proof of the cc part of the Fundamental Lemma (i.e. Lemma 6.4).

Remark. The part of the Fundamental Lemma that concerns the argument b is not provided in the CA version. It does hold that it will preserve its decisiveness, however, it might be the case that even if it is not in the acyclic discarded set of E, it might be in E'.

6.2 Complete and grounded semantics

We can now analyze the complete sub–semantics. Not surprisingly, the correspondence between the extensions depends on the "outside", i.e. w.r.t which range interpretation the decisiveness of arguments is evaluated. In other words, arguments that are decisively in w.r.t. the acyclic range interpretation might not necessarily be decisively in w.r.t. the standard one. Hence, although every ac–admissible extension is aa–admissible, not every ac–complete extension is aa–complete. It can already be observed in Example 5.26. We can observe similar results in the case of cc and ca–complete semantics. Thus, we are left only with the following properties, depicted in Figure 8.

Lemma 6.6. *It holds that:*

1. *Every ac–complete extension is cc–complete*

2. *Every aa–complete extension is ca–complete.*

Proof.

1. Let E be an arbitrary ac–admissible extension. By Lemma 6.2, it is also cc–admissible. If E is ac–complete, but not cc–complete, it would mean that at the same time all arguments decisively in w.r.t. v_E are in E and there are is an argument decisively in w.r.t. v_E but not in E. We reach a contradiction.

2. Let E be an arbitrary aa–admissible extension. By Lemma 6.2, it is also ca–admissible. If all arguments in S that are decisively in w.r.t. v^a_E are in E, then of course so are the ones contained in $S \setminus E^{a+}$. Thus, the ca–completeness criterion is satisfied.

□

We can now continue with an ADF version of Theorem 2.12 from the Dung setting:

Theorem 6.7. 1. *Every xy–preferred extension is an xy–complete extension for $x, y \in \{a, c\}$, but not vice versa.*

2. *The grounded extension might not be an aa(ca)–complete extension.*

3. *The grounded extension is the least w.r.t. set inclusion ac(cc)–complete extension.*

Proof.

1. Let us first show the cc/ac/aa case. Assume an xy–preferred extension E is not xy–complete. This means that there exists some argument $a \in S$ s.t. a is decisively in w.r.t. v_E/v_E^a (depending on the case) but is not in E. By the Fundamental Lemma 6.4, $E \cup \{a\}$ is cc/ac/aa admissible. Obviously $E \subset E \cup \{a\}$, which means E could not have been preferred in the first place. We reach a contradiction. A similar reasoning follows for the ca–case, just with $a \in S \setminus E^{a+}$.

2. It is a result of the fact that these sub–semantics do not treat cyclic attackers as valid, while the grounded semantics does not make this assumption. Let $(\{a, b\}, \{(b, a), (b, b)\}, \{a : \neg b, b : b\})$ be a simple ADF where a is attacked by a self–supporting argument b. The aa–complete extension would be $\{a\}$ and the ca–complete ones would be $\{a\}$ and $\{b\}$, while the grounded one would be simply $\emptyset$.

3. We know that the empty set is ac(cc)–admissible. We can now proceed with adding the arguments that are decisively in w.r.t. its range interpretation and repeat the process (with the now extended set) until no arguments are added. By the Fundamental Lemma (i.e. Lemmas 6.4) we know that after each addition we will still have an ac(cc)–admissible extension. When no further arguments can be added, we clearly receive an ac(cc)–complete extension. It is also obviously a minimal one. What we have described now is just the construction of the grounded extension presented in Proposition 5.12.

□

6.3 Model and stable semantics

The relations between the semantics presented in [23] also carry on to some of the specializations and are shown in Figure 8. However, before we move to them we will prove one more relation:

Lemma 6.8. *Let E be a model. Then $E^{a+} = S \setminus E$.*

Proof Sketch. Assume it is not the case, i.e. there exists an argument $s \in S$ s.t. $C_s(E \cap par(s)) = out$ and at the same time $s \notin E^{a+}$. This means there exists an evaluation for this argument, say $((a_0, ..., a_{n-1}, s), B)$, that is unblocked by v_E^a. This means that $B \cap E = \emptyset$ and none of $a_0, ..., a_{n-1}, s$ is in E^{a+}. Consequently, all elements of the sequence have a corresponding unblocked evaluation. This means that a_0 has an unblocked evaluation, which by the definition is of the form $((a_0), B_0)$. It is easy to see that if $B_1 \cap E = \emptyset$, then $C_{a_0}(E \cap par(a)) = in$. Hence $a_0 \in E$, as E is a model. Now let us analyze a_1. A possible evaluation is of the form $((a_0, a_1), B_0 \cup B_1)$. As $a_0 \in E$ and $B_1 \cap E = \emptyset$, again, it is easy to see that it has to be the case that acceptance condition of a_1 is satisfied and thus $a_1 \in E$. We may now continue this chain of reasoning until we reach s and the conclusion that if it had an unblocked evaluation, its acceptance condition would have been in.

Lemma 6.9. *Every model is ca–complete, but not necessarily ca–preferred.*

Proof. Let E be a model. By Lemma 6.8, we know that v^a_E is defined for every argument in S. Hence, it is its own single completion and all accepted arguments are decisively in w.r.t. it. Consequently, ca–admissibility requirements are satisfied. By the fact that v^a_E is defined for S, ca–completeness follows naturally. However, even without this fact it is also easy to see that by definition of a model, all arguments that have a satisfied acceptance condition are in the set. Thus, whatever is out, cannot possibly be decisively in. Consequently, the completeness requirement is satisfied.

Since this semantics can produce extensions that are comparable w.r.t. set inclusion, then it is not surprising that a model might not be a ca–preferred extension. It is already visible in Example 5.10: models were sets $\{c\}$, $\{a, b\}$ and $\{a, b, c\}$, with only the last one being ca–preferred. □

Lemma 6.10. *Every stable extension is an aa–preferred extension, but not vice versa.*

Proof Sketch. Let us assume E is a stable extension, but not aa–preferred. First of all, E is a model and thus is a ca–complete extension. As v^a_E is defined on all arguments of S by Lemma 6.8, whatever is not in E is mapped to $\mathbf{f}$. As E is also pd–acyclic conflict–free, it means that the members of respective blocking sets are not in E. Consequently, they have to be mapped to $\mathbf{f}$ by v^a_E. We can conclude that the acyclic pd–evaluations are properly protected and E is aa–admissible. If E is not aa–preferred, it means that there exists an aa–admissible extension E' s.t. $E \subset E'$. However, that would mean that $E' \setminus E \subseteq E^{a+}$, i.e. that arguments in $E' \setminus E$ have all acyclic pd–evaluations blocked. It is easy to see that they cannot be later accepted in an aa–preferred extension, as acyclicity on the "inside" is required.

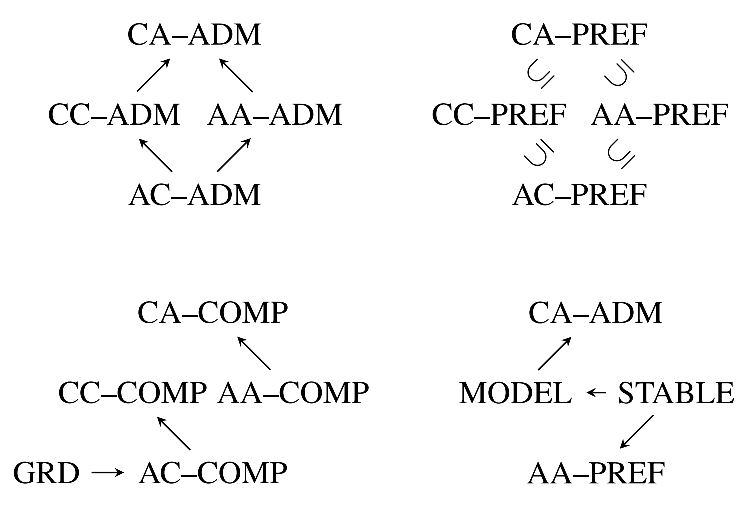


Figure 8: The relations between given extension–based sub–semantics. $x \rightarrow y$ should be read as extensions of x are extensions of y. $x \subseteq y$ should be read as any extension of x is contained in some extension of y.

7 Labeling–Based Semantics of ADFs

The two approaches towards labeling–based semantics of ADFs were developed in [11, 38]. They are based on the notion of a characteristic operator. While in the Dung setting the operator worked with sets, here three valued interpretations are used.

Definition 7.1. *Let V_S be the set of all three–valued interpretations defined on S, s and argument in S and v an interpretation in V_S. The* ***three–valued characteristic operator*** *of D is a function $\Gamma_D : V_S \to V_S$ s.t. $\Gamma_D(v) = v'$ with $v'(s) = \bigsqcap_{w \in [v]_2} C_s(par(s) \cap w^{\mathbf{t}})$.*

Remark. This operator working on three–valued interpretations is a more sophisticated version of the operator introduced in the original paper [13] and recalled in Section 5.3. This will become more visible when we describe the behavior of Γ_D in terms of decisiveness.

Recall that verifying the value of an acceptance condition under a set of extensions of a three–valued interpretation $[v]_2$ is just like testing its value against the completions of the two–valued part of v. Thus, an argument that is **t**/**f** in the $\Gamma_D(v)$ is decisively in/out w.r.t. the two–valued sub–interpretation of v.

Remark. It is easy to see that in a certain sense this operator allows self–justification and self–falsification. Take for example a self–supporter; if we generate an interpretation in which it is false then, obviously, it will remain false. Same follows if we assume it to be true. This results from the fact that the operator functions on interpretations defined on all arguments, thus allowing a self–dependent argument to affect its status. The same is true if we consider bigger positive dependency cycles.

The labeling–based semantics are now as follows:

Definition 7.2. *Let v be a three–valued interpretation for D and Γ_D its characteristic operator. We say that v is:*

- ***three–valued model*** *iff for all $s \in S$ we have that $v(s) \neq \mathbf{u}$ implies that $v(s) = v(\varphi_s)$;*
- ***admissible*** *iff $v \leq_i \Gamma_D(v)$;*
- ***complete*** *iff $v = \Gamma_D(v)$;*
- ***preferred*** *iff it is $\leq_i$–maximal admissible; and*
- ***grounded*** *iff it is the least fixpoint of Γ_D.*

The stable semantics is a slightly different case. Although formally we receive a set, not an interpretation, this makes no difference for stability. As nothing is left undecided, there is a one–to–one correspondence between the extensions and labelings. The current state of the art definition, presented in [38, 11] is based on the grounded semantics:

Definition 7.3. *Let M be a model of D and $D^M = (M, L^M, C^M)$ a reduct of D, where $L^M = L \cap (M \times M)$ and for $m \in M$ we set $C^M_m = \varphi_m[b/\mathbf{f} : b \notin M]$. Let gv be the grounded model of D^M. Model M is* ***stable*** *iff $M = gv^{\mathbf{t}}$.*

Table 2: Labelings of the ADF from Figure 7.

ADM	$\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{u},d:\mathbf{t}\},\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{f},d:\mathbf{t}\},\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{u},d:\mathbf{f}\},$ $\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{f},d:\mathbf{t}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{t},d:\mathbf{f}\},\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{u},d:\mathbf{u}\},$ $\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{u},d:\mathbf{t}\},\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{t}\},\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{f}\},$ $\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{u},d:\mathbf{u}\},\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{u},d:\mathbf{t}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{u},d:\mathbf{f}\},$ $\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{u}\},\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{u},d:\mathbf{f}\},\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{u},d:\mathbf{u}\}$
COMP	$\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{f}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{u},d:\mathbf{u}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{f},d:\mathbf{t}\},$ $\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{u}\},\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{u},d:\mathbf{f}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{t},d:\mathbf{f}\},$ $\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{t}\},\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{f},d:\mathbf{t}\},\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{u},d:\mathbf{u}\}$
PREF	$\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{f},d:\mathbf{t}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{t},d:\mathbf{f}\},\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{f}\},$ $\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{t}\}$
STB	$\{a:\mathbf{t},b:\mathbf{f},c:\mathbf{f},d:\mathbf{f}\},\{a:\mathbf{f},b:\mathbf{t},c:\mathbf{t},d:\mathbf{f}\}$
GRD	$\{a:\mathbf{u},b:\mathbf{u},c:\mathbf{u},d:\mathbf{u}\}$

Example 7.4. *Recall the framework from Example 5.26. The obtained labelings are visible in Table 2. As there are over twenty possible three–valued models, we will not list them.*

8 Comparison of Extensions and Labelings

We shall now compare the new extension–based semantics with the existing labeling–based ones. We will say that an extension E and a labeling v correspond if $v^{\mathbf{t}} = E$.

8.1 Conflict–free extensions and three–valued models

We will start by relating conflict–freeness and three–valued models. Please note that the intuitions of two–valued and three–valued models are completely different and should not be confused – it is just the naming that is somewhat unfortunate.

Theorem 8.1. *Let E be a conflict–free and A a pd–acyclic conflict–free extension. The $\mathbf{u}$–completions of v_E, v_A and v_A^a are three–valued models.*

Proof Idea. Follows straightforwardly from the definition of conflict–freeness and Proposition 6.1.

Theorem 8.2. *Let v be a three–valued model. $v^{\mathbf{t}}$ is a conflict–free set.*

Proof. Since v is a three–valued model, then for every $s \in S$ mapped to $\mathbf{t}$, $v(C_s) = in$. Since $v(C_s) = C_s(v^{\mathbf{t}} \cap par(s))$, conflict–freeness follows straightforwardly. □

8.2 Admissible semantics

We can now continue with the admissible semantics. First, we will tie the notion of decisiveness to admissibility, following the comparison of completions and extending interpretations that we have presented in Section 4.1.

Theorem 8.3. *Let v be a three–valued interpretation and v' its (maximal) two–valued sub–interpretation. v is admissible iff all arguments mapped to $\mathbf{t}$ are decisively in w.r.t. v' and all arguments mapped to $\mathbf{f}$ are decisively out w.r.t. v'.*

Proof. Assume v is admissible, but there exists an argument $s \in S$ mapped to $\mathbf{t}$ that is not decisively in w.r.t. v' or it is mapped to $\mathbf{f}$ and is not decisively out w.r.t. v'. This means there exists a completion v'_S of v' to S s.t. $C_s(par(s) \cap v_S'^{\mathbf{t}})$ is respectively out/in. Obviously, v'_S is also an extension of v, i.e. $v'_S \in [v]_2$. However, if this extension evaluated the condition of s to out/in, then obviously the operator could not have assigned s $\mathbf{t}/\mathbf{f}$ and we reach a contradiction.

Now assume a two–valued interpretation v' such as all arguments mapped to $\mathbf{t}/\mathbf{f}$ are decisively in/out, but its $\mathbf{u}$–completion v is not admissible. This means that $v \not\leq_i \Gamma_D(v)$. Consequently, there exists an argument s mapped to $\mathbf{t}/\mathbf{f}$ by v' that is assigned respectively $\mathbf{f}$ or $\mathbf{u}/\mathbf{t}$ or $\mathbf{u}$. This means that all/some extensions of the interpretation evaluate the condition of s to out/in. Obviously, it means that all/some completions of v' evaluated the condition of s to out/in. Therefore, the initial assignment could not have been decisive and we reach a contradiction. □

However, please note that this theorem does not imply that admissible extensions and labelings "perfectly" coincide. In labelings, we guess an interpretation, and thus assign initial values to arguments that we want to verify later. If they are self–dependent, it of course affects the outcome. In the extension based approaches, we distinguish whether this dependency is permitted. For example, in the ac–approach the accepted arguments cannot take part in support cycles, thus self–justification is not permitted. On the other hand, the iteratively built standard discarded set does not permit self–falsification. Therefore, most of the approaches will have a corresponding labeling, but not vice versa.

Theorem 8.4. *The following holds:*

1. *Let E be a cc–admissible extension. Then the $\mathbf{u}$–completion of v_E is an admissible labeling.*
2. *Let E be an aa(ac)–admissible extension. Then the $\mathbf{u}$–completion of v_E^a is an admissible labeling.*
3. *Not for every ca–admissible extension there exists a corresponding admissible labeling.*
4. *For every admissible labeling there exists a corresponding ca–admissible extension.*

Proof Sketch.

1. The proof was provided already in [35]. However, it also straightforwardly follows from the definition of cc–admissibility and Theorem 8.3.

2. As E is aa–admissible, whatever is mapped to $\mathbf{t}$ by v_E^a is decisively in. Based on Proposition 6.1, whatever is mapped to $\mathbf{f}$ is decisively out. Hence, by Theorem 8.3 the $\mathbf{u}$–completion of v_E^a is admissible. The ac part follows straightforwardly from the definition of ac–admissibility and Theorem 8.3.

3. Recall Example 5.20, where the set $\{b, c, d\}$ was ca–admissible. However, the admissible labelings are: $v_1 = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{u}, d : \mathbf{f}\}, v_2 = \{a : \mathbf{t}, b : \mathbf{u}, c : \mathbf{t}, d : \mathbf{f}\}, v_3 = \{a : \mathbf{u}, b : \mathbf{t}, c : \mathbf{t}, d : \mathbf{u}\}, v_4 = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{u}, d : \mathbf{u}\}, v_5 = \{a : \mathbf{t}, b : \mathbf{u}, c : \mathbf{t}, d : \mathbf{u}\}, v_6 = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{t}, d : \mathbf{f}\}, v_7 = \{a : \mathbf{f}, b : \mathbf{f}, c : \mathbf{f}, d : \mathbf{u}\}, v_8 = \{a : \mathbf{f}, b : \mathbf{f}, c : \mathbf{f}, d : \mathbf{t}\}, v_9 = \{a : \mathbf{u}, b : \mathbf{u}, c : \mathbf{u}, d : \mathbf{u}\}, v_10 = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{t}, d : \mathbf{u}\}$. Thus, there is no labeling corresponding to this extension. The intuition is that it in the ca–approach it can be the case that arguments mapped to $\mathbf{f}$ by the acyclic range interpretation are not decisively out (or not even out), thus the interpretation does not satisfy labeling admissibility (or sometimes even three–valued model) criterion.

4. Let v be an admissible labeling, w its maximal two–valued sub–interpretation and $E = v^{\mathbf{t}}$. If E is not ca–admissible, it means there exists an argument $e \in E$ that is not decisively in w.r.t. the acyclic range interpretation v_E^a of E, even though it is decisively in w.r.t. w (see Theorem 8.3). Thus, there exists a completion v' of v_E^a to $E \cup E^{a+} \cup par(e)$ s.t. $C_e(v'^{\mathbf{t}} \cap par(e)) = out$. Since there is no such completion of w and $w^{\mathbf{t}} = (v_E^a)^{\mathbf{t}}$, it means that there exists at least one argument $s \in S \setminus E$ s.t. $w(s) = \mathbf{f}$ and for which v_E^a is not defined. If v_E^a is not defined for s, it means s has an unblocked pd–acyclic evaluation. This means there exists an acyclic pd–evaluation $((a_0, a_1, ..., s), B)$ for s s.t. no element of the sequence is falsified and no element of the blocking set is true in v_E^a.

   Let us go through the sequence step by step. Since the whole evaluation is not blocked, neither are its "sub–evaluations". Thus, we have that $((a_0), B_0)$ is not blocked by v_E^a. Since $E \cap B_0 = \emptyset$, then a decisively in interpretation for a_0 "can happen". Consequently, w has no power to decisively out a_0 and cannot map it to $\mathbf{f}$, thus the evaluation is not blocked by w. We can now repeat this reasoning for $((a_0, a_1), B_0 \cup B_1)$ and a_1 knowing that a_0 is not mapped to $\mathbf{f}$ and that $E \cap B_1 = \emptyset$. We can continue to go through the evaluations until we reach the original one and the conclusion that w could not have had the power to decisively out a. We reach a contradiction.

8.3 Preferred semantics

Let us now consider the preferred semantics. Unfortunately, due to the differences between two–valued and three–valued approaches and the fact that one follows subset maximality, while the other information precision, we fail to receive an exact correspondence between the results. By this we mean that given a framework there can exist an (arbitrary) preferred extension without a labeling counterpart and a labeling without an appropriate extension, even though certain inclusions relation can be derived.

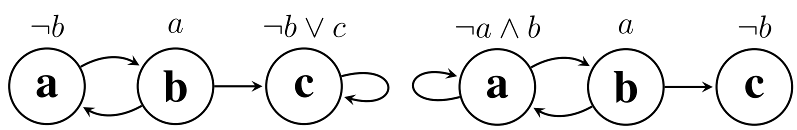
Theorem 8.5. *For any xy–preferred extension there might not exist a corresponding preferred labeling and vice versa.*

As a proof we will now present cases in which the extensions and labelings do not correspond.

Example 8.6. *Recall the framework described in Example 5.20 and visible in Figure 6. The ca–preferred extensions are $\{a, b, c\}$ and $\{b, c, d\}$. The preferred labelings are (please see the proof of Theorem 8.4) are $v_6 = \{a : \mathbf{t}, b : \mathbf{t}, c : \mathbf{t}, d : \mathbf{f}\}$ and $v_8 = \{a : \mathbf{f}, b : \mathbf{f}, c : \mathbf{f}, d : \mathbf{t}\}$, thus corresponding to sets $\{a, b, c\}, \{d\}$. We can see there is no corresponding labeling for $\{b, c, d\}$ and no corresponding extension for $\{d\}$.*

Let us now look at $ADF_1 = (\{a, b, c\}, \{a : \neg a, b : a, c : \neg b \vee c\})$, as depicted in Figure 9. The only ac and aa–preferred extension is $\emptyset$. a and b cannot form a conflict–free extension to start with, so we are only left with c. However, the attack from b on c can be only overpowered by self–support, thus it cannot be part of an ac/aa–admissible extension in the first place. The single preferred labeling solution would be $v = \{a : \mathbf{u}, b : \mathbf{u}, c : \mathbf{t}\}$ and we obtain no correspondence. On the other hand, the result is in compliance with the cc and ca–preferred extension $\{c\}$.

Finally, we have $ADF_2 = (\{a, b, c\}, \{a : \neg a \wedge b, b : a, c : \neg b\})$ depicted in Figure 9. The preferred labeling is $\{a : \mathbf{f}, b : \mathbf{f}, c : \mathbf{t}\}$. The single cc(ac)–preferred extension is $\emptyset$ and again, we receive no correspondence. However, it is compliance with the aa(ca)–preferred extension $\{c\}$.



(a) ADF_1 (b) ADF_2

Figure 9: Sample ADFs

Remark. We will later show that the stable models obtained via extensions and labelings coincide. It also holds that they are aa–preferred and labeling preferred. This means that although perfect correspondence will not be retrieved, in case a stable model exists we have at least one "meeting point" between the two preferred approaches.

8.4 Complete and grounded semantics

Let us first explain complete labeling in terms of decisiveness:

Theorem 8.7. *Let v be a three–valued interpretation and v' its (maximal) two–valued sub–interpretation. v is complete iff all arguments decisively out w.r.t. v' are mapped to $\mathbf{f}$ by v and all arguments decisively in w.r.t. v' are mapped to $\mathbf{t}$ by v.*

Proof. Assume that v is complete, but there exists an argument $s \in S$ that is decisively in/out w.r.t v' that is not mapped to $\mathbf{f}/\mathbf{f}$ by v' (and thus v – from this follows, that $v(s) = \mathbf{u}$). If the s

is decisively in/out w.r.t. v', it naturally means that every extension of v evaluates its condition to in/out. Obviously, if the argument is assigned $\mathbf{u}$ by v, but $\mathbf{t}/\mathbf{f}$ by the characteristic operator, v could not have been a fixpoint (and thus complete) in the first place.

Now assume that every decisive argument is in v', but v is not complete. Consequently, v is not a fixpoint of Γ_D. We have that v is at least admissible by Theorem 8.3, i.e. $v \leq_i \Gamma_D$. Consequently, it has to be the case that there is argument $s \in S$ that is mapped to $\mathbf{u}$ by v but mapped to $\mathbf{t}/\mathbf{f}$ by $\Gamma_D(v)$. Therefore, every extension of v evaluates the arguments condition to $\in /out$. However, then obviously so did every completion of v' and s must have been decisively in/out w.r.t. v'. Hence, it was already assigned $\mathbf{t}/\mathbf{f}$ before and we reach a contradiction. □

With the obvious exception of ca–complete semantics, we have that every cc/ac/aa–complete extensions have a corresponding complete labeling.

Theorem 8.8. *The following holds:*

1. *Let E be a (ac)cc–complete extension. The $\mathbf{u}$–completion of v_E is a complete labeling.*
2. *Let E be an aa–complete extension. The $\mathbf{u}$–completion of v_E^a is a complete labeling.*
3. *Not every ca–complete extension has a corresponding complete labeling and vice versa.*

Proof.

1. By the definition of (ac)cc–completeness, all arguments that are decisively in w.r.t. v_E are already in E (and thus mapped to $\mathbf{t}$ by v_E). By the definition of the discarded set (and standard range), every argument decisively out w.r.t. v_E is mapped to $\mathbf{f}$ by v_E. Thus, by Theorem 8.7 the $\mathbf{u}$–completion of v_E is a complete labeling.

2. By the definition of aa–completeness, all arguments that are decisively in w.r.t. v_E^a are already in E (and thus mapped to $\mathbf{t}$ by v_E). We also know by Proposition 6.1 that all arguments mapped to $\mathbf{f}$ by v_E^a are decisively out. To be able to use Theorem 8.7, we now need to show that all arguments decisively out w.r.t. v_E^a are mapped to $\mathbf{f}$ by v_E^a, i.e. all decisively out arguments are in E^{a+}. Assume there is an argument a that is decisively out w.r.t. v_E^a, but not falsified by it. However, if v_E^a has the power to decisively out a, then of course it conflicts all (minimal) interpretation for which a is decisively in. Consequently, it has the means to block any acyclic pd–evaluation of a and by definition of the acyclic range interpretation, a must have already been mapped to $\mathbf{f}$ by v_E^a. Hence our interpretations satisfies the conditions of Theorem 8.7 and its $\mathbf{u}$–completion is a complete labeling.

3. Recall Example 5.25, where $\{b, c, d\}$ was a ca–complete extension. Since the acceptance condition of a was in fact satisfied, this extension would not give rise to any complete labeling. Now, since we know that every cc–complete extension has a corresponding labeling and that not every cc–complete extension is ca–complete, then obviously we have a labeling corresponding to cc and not to ca approach.

□

8.5 Grounded and stable semantics

As the grounded semantics has a very clear meaning, it is no wonder that both available approaches coincide, as already noted in [11].

Theorem 8.9. *The two–valued grounded extension and the grounded labeling correspond.*

We conclude the report by relating the stability based on the labeling–based grounded model and the one based on the pd–acyclic model.

Theorem 8.10. *A set $M \subseteq S$ of arguments is labeling stable iff it is extension–based stable.*

Proof Sketch. Let us show that if M is labeling stable, it is extension–based stable. Let gv be the grounded labeling of D^M. We know that M is a model of D, we now need to show that every element of M has an unblocked evaluation. By Theorems 8.9 and 6.7, $gv^{\mathbf{t}}$ is an ac–complete extension of D^M and every element $a \in M$ has to have an unblocked pd–acyclic evaluation on $gv^{\mathbf{t}}$. Since $gv^{\mathbf{t}} = M$ (i.e. all arguments of the framework are accepted), every evaluation needs to have an empty blocking set. Therefore, if the blocking set is "brought back" when we revert the reduction, it will consist only of arguments that are not present in the extension. Hence, every member of a given pd–sequence is in M and no member of a blocking set is accepted. Consequently, every $a \in M$ has an unblocked pd–acyclic evaluation in the original framework and M is extension–based stable.

Now we will show that if M is extension–based stable, it is labeling–based stable. Let a be an arbitrary argument in M and $((a_0, .., a_n, a), B)$ its unblocked pd–acyclic evaluation on M. Since $B \cap E = \emptyset$, the evaluation in the reduct will simply be $((a_0, .., a_n, a), \emptyset)$. It is easy to see that the evaluation of a_0 (previously $((a_0), B_0)$) is now $((a_0), \emptyset)$, thus making a_0 decisively in w.r.t. an empty interpretation. By Proposition 5.12, a_0 will be obviously present in the grounded extension (and thus labeling) of D^M. Naturally, with each iteration of the grounded algorithm, $a_1, ..., a_n$ and finally a will be added. Thus, every argument in a will also be presented in the grounded extension (and hence labeling) of D^M. Consequently, M is labeling stable.

9 Concluding Remarks

In this report we have introduced a family of extension–based semantics and their classification w.r.t. positive dependency cycles. Our results also show that they satisfy ADF versions of Dung's Fundamental Lemma and that appropriate sub–semantics preserve the relations between stable, preferred and complete semantics. We have also explained how our formulations relate to the labeling–based approach. We hope that the development of the two–person discussion perspective will allow us to create new notions that will ease the use of abstract dialectical frameworks as tools for dialog and negotiation.

References

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

[2] Katie Atkinson, Trevor J. M. Bench-Capon, and Peter McBurney. PARMENIDES: Facilitating deliberation in democracies. *Artif. Intell. Law*, 14(4):261–275, 2006.

[3] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.

[4] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. Encompassing attacks to attacks in abstract argumentation frameworks. In Claudio Sossai and Gaetano Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 5590 of *LNCS*, pages 83–94. Springer Berlin Heidelberg, 2009.

[5] Pietro Baroni and Massimiliano Giacomin. Semantics of abstract argument systems. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 25–44. Springer, 2009.

[6] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. SCC-Recursiveness: A general schema for argumentation semantics. *Artif. Intell.*, 168(1-2):162–210, 2005.

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

[8] Trevor J. M. Bench-Capon, Henry Prakken, and Giovanni Sartor. Argumentation in legal reasoning. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 363–382. Springer, 2009.

[9] Gustavo A. Bodanza and Fernando A. Tohm. Two approaches to the problems of self-attacking arguments and general odd-length cycles of attack. *Journal of Applied Logic*, 7(4):403 – 420, 2009. Special Issue: Formal Models of Belief Change in Rational Agents.

[10] Guido Boella, Dov Gabbay, Leendert van der Torre, and Serena Villata. Support in abstract argumentation. In *Proc. of COMMA 2010*, pages 111–122, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[11] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In *Proc. IJCAI*, pages 803–809. IJCAI/AAAI, 2013.

[12] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of Dung frameworks and their role in formal argumentation. *Intelligent Systems, IEEE*, PP(99), 2013. In Press.

[13] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proc. KR*, pages 102–111. AAAI Press, 2010.

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

[15] Martin W. A. Caminada, Walter A. Carnielli, and Paul E. Dunne. Semi-stable semantics. *J. Log. and Comput.*, 22(5):1207–1254, October 2012.

[16] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolar abstract argumentation systems. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 65–84. 2009.

[17] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Coalitions of arguments: A tool for handling bipolar argumentation frameworks. *Int. J. Intell. Syst.*, 25(1):83–109, 2010.

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

[19] Andrea Cohen, Sebastian Gottifredi, Alejandro J. García, and Guillermo R. Simari. A survey of different approaches to support in argumentation systems. *The Knowledge Engineering Review*, FirstView:1–38, 2013.

[20] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Inference from controversial arguments. In Geoff Sutcliffe and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3835 of *LNCS*, pages 606–620. Springer Berlin Heidelberg, 2005.

[21] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Prudent semantics for argumentation frameworks. In *Proc. of ICTAI'05*, pages 5 pp.–572, Nov 2005.

[22] Sylvie Coste-Marquis, Sébastien Konieczny, Pierre Marquis, and Mohand Akli Ouali. Weighted attacks in argumentation frameworks. In *Proc. KR*, pages 593–597. AAAI Press, 2012.

[23] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77:321–357, 1995.

[24] Phan Minh Dung and Phan Minh Thang. A unified framework for representation and development of dialectical proof procedures in argumentation. In *Proc. of IJCAI'09*, pages 746–751, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[25] Stefan Ellmauthaler. Abstract dialectical frameworks: properties, complexity, and implementation. Master's thesis, Vienna University of Technology, 2012.

[26] John Fox, David Glasspool, Vivek Patkar, Mark Austin, Liz Black, Matthew South, Dave Robertson, and Charles Vincent. Delivering clinical decision support services: There is nothing as practical as a good theory. *Journal of Biomedical Informatics*, 43(5):831–843, 2010.

[27] Anthony Hunter and Matthew Williams. Aggregating evidence about the positive and negative effects of treatments. *Artificial Intelligence in Medicine*, 56(3):173–190, 2012.

[28] Hadassa Jakobovits and Dirk Vermeir. Dialectic semantics for argumentation frameworks. In *Proc. of ICAIL '99*, pages 53–62, New York, NY, USA, 1999. ACM.

[29] Peter McBurney and Simon Parsons. Dialogue games for agent argumentation. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 261–280. Springer, 2009.

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

[31] Søren Nielsen and Simon Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *Proc. ArgMAS*, volume 4766 of *LNCS*, pages 54–73. Springer, 2007.

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

[33] Farid Nouioua and Vincent Risch. Argumentation frameworks with necessities. In *Proc. SUM*, volume 6929 of *LNCS*, pages 163–176. Springer, 2011.

[34] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In *Proc. COMMA*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 276–284. IOS Press, 2008.

[35] Sylwia Polberg, Johannes Peter Wallner, and Stefan Woltran. Admissibility in the abstract dialectical framework. In *Proc. CLIMA*, volume 8143 of *LNCS*, pages 102–118. Springer, 2013.

[36] Henry Prakken. Models of persuasion dialogue. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 281–301. Springer, 2009.

[37] Iyad Rahwan and Guillermo R. Simari. *Argumentation in Artificial Intelligence*. Springer, 1st edition, 2009.

[38] Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, 205:39 – 70, 2013.

[39] Hannes Strass. Instantiating knowledge bases in abstract dialectical frameworks. In *Proc. CLIMA*, volume 8143 of *LNCS*, pages 86–101. Springer, 2013.

[40] Hannes Strass and Johannes Peter Wallner. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, Vienna, Austria, July 2014. Accepted for publication.