Extension–Based Semantics of Abstract Dialectical Frameworks

Sylwia POLBERG [a,1]
[a] *Vienna University of Technology, Institute of Information Systems, Favoritenstraße 9-11, 1040 Vienna, Austria.*

Abstract. One of the most prominent tools for abstract argumentation is the Dung's framework, AF for short. 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 existing problems in the field. One of the most controversial issues, recognized not only in argumentation, concerns the support or positive dependency cycles. In this paper we introduce a new method to ensure acyclicity of arguments and present a family of extension–based semantics built on it, along with their classification w.r.t. cycles. Finally, we provide ADF versions of the properties known from the Dung setting.

Keywords. abstract argumentation, abstract dialectical frameworks, argumentation semantics

Introduction

Over the last years, argumentation has become an influential subfield of artificial intelligence [2]. One of its subareas is the *abstract argumentation*, which became especially popular thanks to the research of Phan Minh Dung [3]. Although the framework he has developed is quite powerful, it has certain shortcomings, which inspired a search for more general models [4]. Among the most abstract enrichments are the abstract dialectical frameworks, ADFs for short [5]. However, a framework cannot be considered a suitable argumentation tool without properly developed semantics.

The semantics of a framework are meant to capture what we consider rational. Many of the advanced ones, such as grounded or complete, coincide when faced with simple, tree–like frameworks. The differences between them become more visible in complicated cases. On various occasions examples were found for which none of the available semantics returned satisfactory answers. They gave rise to new concepts, such as prudent and careful semantics for handling indirect attacks [6,7], sustainable and tolerant for self–attackers [8] or some of the SCC–recursive semantics for the problem of attack cycles [9]. Introducing a new relation, such as support, creates additional problems.

The most controversial issue in a setting permitting support concerns the support cycles and is handled differently from formalism to formalism. Among the best known

[1]The author is funded by the Vienna PhD School of Informatics. This research is a part of the project I1102 supported by the Austrian Science Fund FWF. An earlier version of this paper can be found in [1].

ones are the Bipolar Argumentation Frameworks (BAFs) [10], Argumentation Frameworks with Necessities (AFNs) [11] and Evidential Argumentation Systems (EASs) [12]. While the latter two discard support cycles, BAFs do not make such restrictions and in general, neither do ADFs. This variety is not an error in any of the structures. First of all, in a more advanced setting, a standard Dung semantics can be extended in several ways. Moreover, since one can find arguments both for and against any of the cycle treatments, lack of consensus as to what approach is the best should not be surprising.

Many properties of the available semantics can be seen as "inside" ones, 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 defends against all possible attacks in the framework. We can then restrict this by saying that self–attackers are not rational, and thus limit the set of arguments we have to defend the extension from. If we now add support, we can again define admissibility in the basic manner. However, one often demands that the extensions are free from support cycles and that we only defend from arguments not taking part in them. 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.

Although various extension–based semantics for ADFs have already been proposed in the original paper [5], many of them were defined only for a particular ADF subclass and were not suitable for all types of situations. Moreover, they did not solve the problem of positive dependency cycles. The aim of this paper is to address these issues. We introduce a family of extension–based semantics and specialize them to handle the problem of support cycles, as their treatment seems to be the greatest difference between the available frameworks. Furthermore, we present a classification of our sub–semantics in the internal–external fashion that we have described before. We also recall our research on admissibility in [13] and show how it fits into the new system. Finally, we show which known properties, such as Fundamental Lemma, carry over from the Dung framework.

1. Dung's Argumentation Frameworks

Let us briefly recall the argumentation framework by Dung [3] and its semantics. For more details we refer the reader to [14].

Definition 1.1. 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.

Definition 1.2. An argument $a \in A$ is **defended** by a set E in AF, if for each $b \in A$ s.t. $(b,a) \in R$, $\exists c \in E$ s.t. $(c,b) \in R$. A set $E \subseteq A$ is:

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

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}.

We would also like to recall the notion of range, as its idea will be used in the ADF semantics. Even in the Dung setting, the concepts of the E^+ and E^- sets can be used to redefine defense. Finally, we also list some of the properties of Dung's semantics [3].

Definition 1.3. Let E^+ be the set of arguments attacked by E and E^- the set of arguments that attack E . $E^+ \cup E$ is the **range** of E.

Lemma 1.4. ***Dung's Fundamental Lemma*** *Let E be an admissible extension, a and b two arguments defended by E. Then $E' = E \cup \{a\}$ is admissible and b is defended by E'.*

Theorem 1.5. *Every stable extension is a preferred extension, but not vice versa. Every preferred extension is a complete extension, but not vice versa. The grounded extension is the least w.r.t. set inclusion complete extension. The complete extensions form a complete semilattice w.r.t. set inclusion.* [2]

2. Abstract Dialectical Frameworks

Abstract dialectical frameworks have been defined in [5] and further studied in [13,15, 16,17,18]. Their main goal is to be able to express arbitrary relations. This is achieved by the use of acceptance conditions, which define what sets of arguments should be present in order to accept or reject a given argument.

Definition 2.1. 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.

An acceptance condition is a total function $C_s : 2^{par(s)} \rightarrow \{in, out\}$, where $par(s) = \{p \in S \mid (p,s) \in L\}$ is the set of **parents** of an argument s.

Please note that one can also represent the acceptance conditions by propositional formulas over arguments instead of "boolean" functions [19]. It is easy to see that links L are somewhat redundant and can be extracted from the conditions. Thus, we will use of shortened notation and assume an ADF $D = (S,C)$ through the rest of this paper. In order to introduce our new semantics, we need to explain some basic notions first.

Decisiveness: At the heart of Dung's semantics is the concept of defense. Admissibility represents a defensible stand, where no matter what our opponent says against us, we can provide a counter argument to it. From this also follows that given accepted arguments E and the ones attacked by them (E^+), whatever argument is left cannot further change the status of the ones in E. This idea that an argument has a "final" assignment is captured with the notion of decisiveness in ADFs. Its basic form, where we check the outcome of an acceptance condition w.r.t. accepted and rejected sets of arguments, can already be found in the original paper [5]. It is further developed in an interpretation–based form in [13], which will be used in this paper. A two–valued interpretation v is a mapping that assigns the truth values $\{\mathbf{t},\mathbf{f}\}$ to arguments. We will use v^x to denote a set of arguments mapped to x by v, where x is some truth–value. Given an argument $s \in S$, its condition C_s and an interpretation v, we define a shorthand $v(C_s)$ as $C_s(v^{\mathbf{t}} \cap par(s))$. Now, in order

[2] 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.

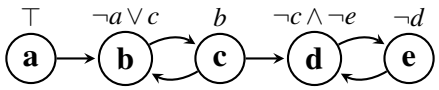

Figure 1. Sample ADF

to check is some (possibly partial) interpretation is decisive for s, we basically need to check if all "bigger" interpretations stemming from it evaluate C_s in the same way:

Definition 2.2. Given a two–valued interpretation v defined on a set A, a **completion** of v to a set Z where $A \subseteq Z$ is an interpretation v' defined on Z s.t. $\forall a \in A\ v(a) = v'(a)$. By a $\mathbf{t}/\mathbf{f}$ completion we understand v' that maps all arguments in $Z \setminus A$ respectively to $\mathbf{t}/\mathbf{f}$.

Definition 2.3. We say that an interpretation v defined on A 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 2.4. Let $(\{a,b,c,d,e\}, \{C_a : \top, C_b : \neg a \vee c, C_c : b, C_d : \neg c \wedge \neg e, C_e : \neg d\})$ be the ADF in Figure 1. Examples of decisively in interpretations for b include $v_1 = \{c : \mathbf{t}\}$. This means that knowing that c is true, we know that the whole disjunction (and thus the acceptance condition) are satisfied. Formally speaking, v_1 is decisive as both of its completions $\{c : \mathbf{t}, a : \mathbf{f}\}$ and $\{c : \mathbf{t}, a : \mathbf{t}\}$ satisfy the condition.

Acyclicity: Let us now focus on the issue of positive dependency cycles. Please note we refrain from calling them support cycles in the ADF setting in order not to confuse them with specific definitions of support available in the literature [10].

Informally speaking, an argument takes part in a cycle if its acceptance depends on itself. An intuitive way of verifying the acyclicity of an argument would be to "track" its evaluation, e.g. in order to accept a we need to accept b, for b we need c and so on. This basic case becomes more complicated when disjunction is introduced. We then receive a number of such "paths", with only some of them proving to be acyclic. Moreover, they might be conflicting one with each other. It can also happen that all acyclic evaluations are blocked and a cycle is forced. Our approach to acyclicity is based on the idea of such "paths" that are accompanied by sets of arguments used to detect possible conflicts.

Let us now introduce the formal definitions. In order to obtain the arguments that are required or should be avoided for the acceptance of a given argument, we will make use of decisive interpretations. Naturally, it suffices to focus on the minimal ones, by which we understand that both $v^{\mathbf{t}}$ and $v^{\mathbf{f}}$ are minimal w.r.t. $\subseteq$. Given an argument $s \in S$ and $x \in \{in, out\}$, by $min\_dec(x,s)$ we will denote the set of minimal two–valued interpretations that are decisively x for s.

Definition 2.5. 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 2.6. An **acyclic positive dependency evaluation** ace^a for $a \in A$ based on a given pd–function pd is a pair $((a_0, ..., a_n), B)$, [3] where $B = \bigcup_{i=0}^{n} pd(a_i)^{\mathbf{f}}$ and $(a_0, ..., a_n)$

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

is a sequence of distinct elements of A s.t.: *1)* $\forall_{i=0}^{n}\, pd(a_i) \neq \mathscr{N}$, *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 as the **pd–sequence** and to 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.

We will write that 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. We can either discard an argument required by the evaluation or accept one that is capable of preventing it. This corresponds to rejecting a member of a pd–sequence or accepting an argument from the blocking set. We can now formulate this "conflict" by the means of an interpretation:

Definition 2.7. 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. $v(a_i) = \mathbf{f}$.

Example 2.4 (Continued)**.** Let us now show why we store the blocking set. For argument b there exist two minimal decisively in interpretations: $v_1 = \{a : \mathbf{f}\}$ and $v_2 = \{c : \mathbf{t}\}$. The interpretations for a and c are respectively $w_1 = \{\}$ and $z_1 = \{b : \mathbf{t}\}$. Therefore, on $\{a, b, c\}$ we have two pd–functions, namely $pd_1 = \{a : w_1, b : v_1, c : z_1\}$ and $pd_2 = \{a : w_1, b : v_2, c : z_1\}$. They result in one acyclic evaluation for a: $((a), \emptyset)$, one for b: $((b), \{a\})$ and one for c: $((b, c), \{a\})$. Let us analyze the set $E = \{a, b, c\}$. We can see that accepting a "forces" a cycle between b and c. The acceptance conditions of all arguments are satisfied, thus this simple check is not enough to verify if a cycle occurs. If we checked only if the members of the pd–sequences are accepted, we would also get the wrong answer. Only looking at the whole evaluations shows us that b and c are both blocked by a. Although b and c are technically pd–acyclic in E, we see that their evaluations are in fact blocked and this type of conflict needs to be taken into account by the semantics.

3. Extension–Based Semantics of ADFs

Although various semantics for ADFs have already been defined in the original paper [5], only three of them – conflict–free, model and grounded – are still used (issues with the other notions can be found in [13,15,16]). 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. Recall that there is no consensus among available bipolar frameworks as to how support cycles should be treated. 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 based on the inside–outside intuition presented in the introduction. Appropriate semantics will receive a two–element prefix $xy-$, where x will denote whether cycles are permitted or not on the "inside" and y on the "outside". We will use $x, y \in \{a, c\}$, where a will stand for *acyclic* and c for *cyclic* constraints. 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 (see Theorem 3.15 and [20]), they have quite unique naming and further annotations are not needed. We are thus left with admissible, preferred and complete. The BAF approach follows the idea that we

can accept arguments that are not acyclic and we allow our opponent to do the same. The ADF semantics created in [13] also shares this view. Therefore, they will receive the $cc-$ prefix. In contrast, AFNs and EASs do not permit cycles both in extensions and in attackers. Thus, the semantics following this line of reasoning will be prefixed with $aa-$[4]. Although we believe that a non–uniform approach can be suitable in certain situations (i.e. $ca-$ and $ac-$), for now we will focus only on the $aa-$ and $cc-$ ones.

Conflict–free and naive semantics: In the Dung setting, conflict–freeness meant that the elements of an extension could not attack one another. In ADF setting, this notion is strengthened by also providing required support. This represents the intuition of arguments that can stand together presented in [14].

Definition 3.1. A set of arguments $E \subseteq S$ is a **conflict–free extension** of D iff for all $s \in E$ we have $C_s(E \cap par(s)) = in$.

In the acyclic version of conflict–freeness we also need to deal with the conflicts arising on the level of evaluations. To meet the formal requirements, we first have to show how the notions of range and the E^+ set are moved to ADFs.

Definition 3.2. 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 $a \in M$ set $v_E(a) = \mathbf{t}$;
2. For every $b \in S \setminus M$ that is decisively out in v_E, set $v_E(b) = \mathbf{f}$ and add b to M;
3. Repeat Step 2 until no new elements are 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 the **range interpretation** of E.

However, this notion of range is quite strict as it requires an explicit "attack" on all possible arguments. This is not always a desirable property, since depending on the approach we might not treat cyclic arguments as valid and hence want them "out of the way".

Definition 3.3. 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 $b \in S \setminus M$ 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 Step 2 until no new elements are added to M.

By E^{a+} we understand the set $(v_E^a)^{\mathbf{f}}$ and call it the **acyclic discarded set**. v_E^a now forms the **acyclic range interpretation** of E.

We can now define an acyclic version of conflict–freeness and the naive semantics:

Definition 3.4. A conflict–free extension E is a **pd–acyclic conflict–free extension** of D iff every argument $a \in E$ has an unblocked acyclic pd–evaluation on E w.r.t. v^E.[5]

[4] More explanations can be found in [20].

[5] Since we are in a conflict–free setting, it suffices to check whether $E \cap B = \emptyset$ to see if an evaluation is not blocked. Consequently, it does not matter which version of range we use.

Definition 3.5. The **naive** and **pd–acyclic naive** extensions are respectively maximal w.r.t. set inclusion conflict–free and pd–acyclic conflict–free extensions.

Example 2.4 (Continued)**.** The conflict–free extensions of our ADF $(\{a,b,c,d,e\},\{C_a : \top, C_b : \neg a \vee c, C_c : b, C_d : \neg c \wedge \neg e, C_e : \neg d\})$ are $\emptyset, \{a\}, \{b\}, \{d\}, \{e\}, \{a,d\}, \{a,e\}, \{b,c\}, \{b,d\}, \{b,e\}, \{a,b,c\}, \{b,c,e\}$ and $\{a,b,c,e\}$. As a blocks evaluations of b and c, $\{a,b,c\}$ and $\{a,b,c,e\}$ are not pd–acyclic conflict–free. Naive and pd–acyclic naive extensions are respectively $\{\{a,d\},\{b,d\},\{a,b,c,e\}\}$ and $\{\{a,d\},\{a,e\},\{b,d\},\{b,c,e\}\}$.

Let us briefly look at the discarded sets of $\{a,d\}$. Since a blocks the evaluations of b and c, they will be included in the acyclic version. However, since none of them is decisively out w.r.t. just $\{a\}$, they will not appear in the standard version. It is easy to see that presence of d permanently discards e and thus it is in both sets.

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

Definition 3.6. A conflict–free extension E is a **model** of D if $\forall\, s \in S,\ C_s(E \cap par(s)) = in$ implies $s \in E$.

Verifying whether a condition of an argument s is met does check the effect of accepting s on E, thus it can happen that including s breaks conflict–freeness of E. Consequently, it is clear to see that model semantics is not universally defined. Moreover, the extensions might not be maximal w.r.t. $\subseteq$, as visible in the continuation of Example 2.4.

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. Although the original reduction–based method was not adequate [15], the initial idea still holds and we use it to define stability. Although the produced extensions are now incomparable w.r.t. set inclusion, the semantics is still not universally defined.

Definition 3.7. A model E is a **stable extension** iff it is pd–acyclic conflict–free.

Example 2.4 (Continued)**.** Out of all conflict–free extensions, only $\{a,d\},\{a,e\}$ and $\{a,b,c,e\}$ are models. $\{a\}$ itself is not a model, since $C_d(\{a\} \cap \{c,e\}) = in$ and $C_e(\{a\} \cap \{d\}) = in$ and we thus we can still accept some arguments. Recall that $\{a,b,c,e\}$ was not pd–acyclic conflict–free. Consequently, $\{a,d\}$ and $\{a,e\}$ are our stable extensions.

Grounded semantics: The grounded semantics introduced in [5] is defined in the terms of a special operator. Although it might look complicated at first, this is nothing more than analyzing decisiveness using the set, not the interpretation form [20].

Definition 3.8. 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.

Example 2.4 (Continued)**.** As noted in [5], the grounded extension can be obtained by applying the operator to $(\emptyset,\emptyset)$. Let us compute $acc(\emptyset,\emptyset)$. Obviously, we can accept a. The condition of c is already *out*, and so are the ones of b, d and e when we consider S' being respectively $\{a\},\{e\}$ and $\{d\}$. It is easy to see that for now, $reb(\emptyset,\emptyset)$ remains empty – S' such as $\{a,c\},\{b\},\emptyset$ and $\emptyset$ evaluate respectively b,c,d and e to *in*. Next

step is $\Gamma_D(\{a\},\emptyset)$. However, by reasoning presented above no further arguments can be accepted or rejected and we reach a fixpoint. Thus, $\{a\}$ is our grounded extension.

Admissible and preferred semantics: In [13] we have presented our first definition of admissibility, before the sub–semantics classification was developed. The new, simplified version of our previous formulation, is now as follows:

Definition 3.9. A conflict–extension $E \subseteq S$ is **cc–admissible** in D iff every $e \in E$ is decisively in w.r.t to the range interpretation v_E.

It should be noted that decisiveness w.r.t. range encapsulates the defense known from the Dung setting. If an argument is decisively in, then any set of arguments that would have the power to out the acceptance condition is "prevented" by the interpretation. Hence, the statements required for the acceptance of a are mapped to **t** and those that would make us reject a are mapped to **f**. The former encapsulates the required "support", while the latter contains the "attackers" known from the Dung setting.

When working with the acyclic semantics, we not only have to defend the members, but also their acyclic evaluations. Example 2.4 shows that although decisiveness encapsulates defense of an argument, it might not be the case for its evaluation.

Definition 3.10. A pd–acyclic conflict–free extension E is **aa–admissible** in D iff every $e \in E$ *1)* is decisively in w.r.t. acyclic range interpretation v^a_E, and *2)* has an unblocked acyclic pd–evaluation on E s.t. all members of its blocking set B are mapped to **f** by v^a_E.

Definition 3.11. A set of arguments is **xy–preferred** in D iff it is maximal w.r.t. set inclusion xy–admissible, where $x,y \in \{a,c\}$.

Example 2.4 (Continued)**.** Recall our ADF $(\{a,b,c,d,e\},\{C_a:\top, C_b:\neg a \vee c, C_c:b, C_d:\neg c \wedge \neg e, C_e:\neg d\})$ and that $\{b,c\}$ was a pd–acyclic conflict–free extension. Its standard and acyclic discarded sets are just $\{d\}$. It is easy to see, that both arguments are decisively in w.r.t. both range interpretations. Although uttering a would not change the values of the conditions, it would still force a cycle between b and c. Thus, the acyclicity is not "defended" and $\{b,c\}$ is cc, but not aa–admissible. Similar follows for $\{b,c,e\}$. $\emptyset$ and $\{a\}$ are trivially both cc and aa–admissible. Since the discarded sets of $\{e\}$ include d, so are $\{e\}$ and $\{a,e\}$. By the reasoning above, it is also easy to see that $\{a,b,c\},\{b,c,e\}$ and $\{a,b,c,e\}$ are cc (though not aa) admissible. Finally, apart from $\emptyset,\{a\}$ and $\{a,e\}$, also $\{a,d\}$ is aa–admissible. Its acyclic discarded set is $\{b,c,e\}$ and thus decisiveness and evaluation defense are preserved. Since the standard set is just $\{e\}$, d is not decisively in and the extension is not cc–admissible. The set $\{a,b,c,e\}$ is the only cc–preferred extension, while $\{a,d\}$ and $\{a,e\}$ are aa–preferred.

Complete semantics: Completeness represents an approach in which we have to accept everything we can safely conclude from our opinions. In the Dung setting "safety" means defense, while in the bipolar setting it is strengthened by providing sufficient support. In a sense, it follows the model intuition that whatever we can accept, we should accept. However, now we not only use an admissible base in place of a conflict–free one, but also defend the arguments in question. Therefore, instead of checking if an argument is in, we want it to be decisively in.

Definition 3.12. A cc–admissible extension E is **cc–complete** in D iff every argument in S that is decisively in w.r.t. to range interpretation v_E is in E. An aa–admissible extension E is **aa–complete** in D iff every argument in S that is decisively in w.r.t. to acyclic range interpretation v_E^a is in E.[6]

Example 2.4 (Continued)**.** It is easy to see that since a can always be accepted, only $\{a,d\},\{a,e\}$ and $\{a\}$ are aa–complete. From this also follows that cc–admissible extensions such as $\emptyset,\{e\},\{b,c\}\{b,c,e\}$ are not cc–complete. Since the discarded set of $\{a,b,c\}$ is $\{d\}$, e can still be included in the extension and thus it is also disqualified. Therefore, we obtain three cc–complete sets: $\{a\}$, $\{a,e\}$ and $\{a,b,c,e\}$.

Properties: Let us close the paper with the properties of our semantics. Although the study provided here will by no means be exhaustive, we would like to show how the lemmas and theorems from the original paper on AFs [3] are shifted into this new setting[7].

Even though every pd–acyclic conflict–free extension is also conflict–free, it does not mean that every aa–admissible is cc–admissible. These approaches differ significantly. The first one makes additional restrictions on the "inside", but due to acyclicity requirements on the "outside" there are less arguments a given extension has to defend from. The latter allows more freedom as to what we can accept, but also gives this freedom to the opponent, thus there are more possible attackers. Moreover, it should not come as a surprise that these differences pass over to the preferred and complete semantics, as visible in Example 2.4. Our results show that admissible sub–semantics satisfy the Fundamental Lemma. Moreover, the relations between the semantics presented in [3] are preserved by some of the specializations.

Lemma 3.13. *CC Fundamental Lemma: Let E be a cc–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–admissible and b is decisively in w.r.t. v'_E.*

Lemma 3.14. *AA Fundamental Lemma: Let E be an aa-admissible extension, v_E^a its acyclic range interpretation and $a,b \in S$ two arguments decisively in w.r.t. v_E^a. Then $E' = E \cup \{a\}$ is aa–admissible and b is decisively in w.r.t. v'_E.*

Theorem 3.15. *Every stable extension is an aa–preferred extension, but not vice versa. Every xy–preferred extension is an xy–complete extension for $x,y \in \{a,c\}$, but not vice versa. The grounded extension might not be an aa–complete extension. The grounded extension is the least w.r.t. set inclusion cc–complete extension.*

4. Conclusions and future work

In this paper we have introduced a method for detecting positive dependency cycles in ADFs and a family of semantics based on it. Our results show that they satisfy ADF versions of Dung's Fundamental Lemma and that appropriate sub–semantics preserve the relations between stable, preferred and complete approaches. Our future work focuses on shifting the mentioned bipolar frameworks into the ADF setting and proving that their

[6] No further assumptions as to the defense of the evaluations are needed, as visible in Lemma 3.14.

[7] The relevant proofs can be found in [20].

semantics are properly generalized by the presented approaches. Moreover, we would like to study the complexity of the new semantics. Final aim is to provide an efficient implementation, as the existing one was created purely for verification purposes and leaves a lot of room for optimization.

References

[1] Sylwia Polberg. Extension-based semantics of abstract dialectical frameworks. In *Proc. of NMR*, 2014.

[2] Iyad Rahwan and Guillermo R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[3] 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.

[4] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of dung frameworks and their role in formal argumentation. *Intelligent Systems, IEEE*, 29(1):30–38, Jan 2014.

[5] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proc. KR '10*, pages 102–111. AAAI Press, 2010.

[6] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Inference from controversial arguments. In Geoff Sutcliffe and Andrei Voronkov, editors, *Proc. LPAR '05*, volume 3835 of *LNCS*, pages 606–620. Springer Berlin Heidelberg, 2005.

[7] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Prudent semantics for argumentation frameworks. In *Proc. of ICTAI'05*, pages 568–572, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Gustavo A. Bodanza and Fernando A. Tohmé. 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.

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

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

[11] Farid Nouioua. AFs with necessities: Further semantics and labelling characterization. In Weiru Liu, V.S. Subrahmanian, and Jef Wijsen, editors, *Proc. SUM '13*, volume 8078 of *LNCS*, pages 120–133. Springer Berlin Heidelberg, 2013.

[12] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In *Proc. COMMA '08*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 276–284. IOS Press, 2008.

[13] Sylwia Polberg, Johannes Peter Wallner, and Stefan Woltran. Admissibility in the abstract dialectical framework. In *Proc. CLIMA'13*, volume 8143 of *LNCS*, pages 102–118. Springer, 2013.

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

[15] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In *Proc. IJCAI'13*, pages 803–809. AAAI Press, 2013.

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

[17] Hannes Strass. Instantiating knowledge bases in abstract dialectical frameworks. In *Proc. CLIMA'13*, volume 8143 of *LNCS*, pages 86–101. Springer, 2013.

[18] Hannes Strass and Johannes Peter Wallner. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In *Proc. KR '14*, Vienna, Austria, July 2014. Forthcoming.

[19] Stefan Ellmauthaler. Abstract dialectical frameworks: properties, complexity, and implementation. Master's thesis, Faculty of Informatics, Institute of Information Systems, Vienna University of Technology, 2012.

[20] Sylwia Polberg. Extension–based semantics of abstract dialectical frameworks. Technical Report DBAI-TR-2014-85, Institute for Information Systems,Vienna University of Technology, 2014.