Stage Semantics and the SCC-recursive Schema for Argumentation Semantics

Wolfgang Dvořák, *University of Vienna, Faculty of Computer Science, Vienna, Austria*
E-mail: wolfgang.dvorak@univie.ac.at

Sarah Alice Gaggl, *Technische Universität Dresden, Institute of Artificial Intelligence, Computational Logic Group, Dresden, Germany*
E-mail: sarah.gaggl@tu-dresden.de

Abstract

Recently, stage and $cf2$ semantics for abstract argumentation attracted specific attention. By distancing from the notion of defense, they are capable to select arguments out of odd-length cycles. In case of $cf2$ semantics, the SCC-recursive schema guarantees that important evaluation criteria for argumentation semantics, like directionality, weak- and $\mathcal{CF}$-reinstatement, are fulfilled. Beside several desirable properties, both stage and $cf2$ semantics still have some drawbacks. The stage semantics does not satisfy the above mentioned evaluation criteria, whereas $cf2$ semantics produces some questionable results on frameworks with cycles of length ≥ 6. Therefore, we suggest to combine stage semantics with the SCC-recursive schema of $cf2$ semantics. The resulting $stage2$ semantics overcomes the problems regarding $cf2$ and stage semantics.

We study properties of $stage2$ semantics and its relations to existing semantics, show that it fulfills the mentioned evaluation criteria, study strong equivalence for $stage2$ semantics, and provide a comprehensive complexity analysis of the associated reasoning problems. Besides the analysis of $stage2$ semantics we also complement existing complexity results for $cf2$ by an analysis of tractable fragments and fixed parameter tractability. Furthermore we provide answer-set programming (ASP) encodings for $stage2$ semantics and labeling-based algorithms for $cf2$ and $stage2$ semantics.

Keywords: abstract argumentation, cf2 semantics, stage2 semantics, odd-length cycles, computational complexity, strong equivalence, answer-set programming, labelings

1 Introduction

The formalism of abstract argumentation frameworks (AFs) as introduced by Dung [22], provides a way of dealing with conflicting knowledge. It allows for a concise representation of statements together with a binary relation between the statements, where the conflicts are then solved on the semantical level. In abstract AFs the internal structure of the statements (also called *arguments*) is not of specific interest. One concentrates on the relation between the arguments, where the actual meaning of the so called *attack relation* depends on the particular application field. Typically this relation denotes some kind of conflict between the statements.

AFs are typically represented as directed graphs, where the nodes are the arguments and the edges are the attack relation. Loops of different kind can appear in these graphs, like even-length as well as odd-length cycles and even self-loops as a special form of odd-length cycles. Recent investigations [5, 7, 9, 10, 44, 58] showed that some of these loops can have a strong and in certain cases undesired influence on the computation of solutions, e.g. in the

case of stable semantics an odd-cycle can prevent the AF from having an extension at all. The meaning of the different loops is not defined explicitly by the AF but by the argumentation semantics one chooses. In particular it is not clear what it means for two arguments to appear in either an even- or an odd-length loop. However, it seems obvious to take this special graph structure also in the evaluation of the framework into account.

Traditional argumentation semantics build on the concept of admissible sets, i.e. conflict-free sets where each argument attacking an argument in the set is also attacked by the set, which may yield undesired behavior in the presence of odd-length cycles. Consider an AF consisting of three arguments a, b and c, where a attacks b, b attacks c, and c attacks a. For the set $S = \{a\}$ the first condition of admissible sets is fulfilled, as a is not self-attacking. Now a is attacked by c, then the set S needs to attack c to defend a. This would require that b belongs to S, as it is the only attacker of c. But a attacks b, thus the first condition is violated. So in an AF consisting of an odd-length cycle the only admissible extension is the empty set. Hence, the traditional well-studied admissibility based semantics are not applicable if we want to select arguments in odd-length cycles.

There are situations, where it is not required that the arguments are defended against all attacks on them. In particular if one sees arguments as different choices and the attack relation between two choices denotes that they can not stand together. Then, one would like to be able to select maximal consistent sets of arguments as solutions. In the following we denote semantics which build on maximal conflict-free, so called naive sets, as *naive-based* semantics. Lately, there have been proposed several approaches how to deal with such situations [5, 7, 9, 10, 44, 58, 61]. Out of them the $cf2$ semantics as introduced in [4] and revisited later in [7], attracted specific attention [44, 45, 47]. This naive-based semantics does not only treat cycles in a more sensitive way than others, the SCC-recursive schema where it is defined guarantees that principal properties for argumentation semantics like directionality [5] are fulfilled. The recursive definition of $cf2$ is based on a decomposition of the framework along its strongly connected components (SCCs), where in the base case (if the AF consists of a single SCC) the naive sets are chosen. In contrast to admissible-based semantics, $cf2$ semantics does not require to defend arguments against all attacks but the SCC-recursive schema somehow simulates defense. In particular arguments do not need to be defended against attacks from members of their own SCC and attacks from other SCCs do only apply if they come from acceptable arguments. However, one weakness of $cf2$ is that in case the AF consists of, resp. contains, an SCC of certain size, even in the absence of odd-length cycles, the evaluation is questionable [44, 47]. These shortcomings already appear in the base-cases for $cf2$ where only naive semantics are applied.

On the other side, the well studied naive-based semantics *stage* [61] can also handle odd-length cycles and does not change the behavior of odd-cycle-free AFs. The disadvantages of stage semantics are that very basic properties are not satisfied, for example the skeptical acceptance of unattacked arguments, i.e. the weak reinstatement property [5] is violated. While naive-based semantics seem to be the right candidates when the above described behavior of admissible-based semantics is unwanted, there are several shortcomings with existing approaches, as mentioned above.

We can see that stage semantics lack a lot of desired properties which are guaranteed by the SCC-recursive schema, while $cf2$ mainly suffers from using naive as base semantics. Thus to overcome the observed problems we propose to combine the concepts of $cf2$ and stage semantics, which results in the sibling semantics $stage2$. We use the SCC-recursive schema of $cf2$ semantics and instantiate the base case with stage semantics. It turns out,

that the novel $stage2$ semantics resolves the shortcomings of both $cf2$ and stage semantics. Besides introducing the novel $stage2$ semantics, we perform a systematic analysis of $stage2$ and compare the results with the other naive-based semantics, $cf2$ and stage. We consider well established methods starting with relating $stage2$ to other naive-based semantics, as well as the classification in terms of the general *evaluation criteria* as proposed by Baroni et al. in [5].

Recently the concept of *strong equivalence* for argumentation semantics attracted specific attention [56]. Two AFs are strongly equivalent w.r.t. a semantics σ, if no matter which new arguments and attacks one adds to both of them, they always produce the same extensions. Surprisingly, it turned out that for $cf2$ semantics, strong equivalence is only given if the two frameworks are identically [47]. So far $cf2$ is the only semantics where this behavior was observed, while for all other semantics, it was possible to identify redundant attacks [56, 47]. This special behavior has been made explicit with the *succinctness property* [47]. We will show that $stage2$ is the second semantics which satisfies the succinctness property.

Strong equivalence not only gives an additional property to investigate the differences between argumentation semantics but also has some interesting applications. First, suppose we model a negotiation between two agents via argumentation frameworks. Here, strong equivalence allows to characterize situations where the two agents have an equivalent view of the world which is moreover robust to additional information. Second, we believe that the identification of *redundant attacks* is important in choosing an appropriate semantics, in particular if an abstract argumentation framework has been built from a given knowledge base. Caminada and Amgoud outlined in [15] that the interplay between how a framework is built and which semantics is used to evaluate the framework is crucial in order to obtain useful results when the (claims of the) arguments selected by the chosen semantics are collected together. Knowledge about redundant attacks (w.r.t. a particular semantics) might help to identify unsuitable such combinations or showing that two ways to build a framework are actual equivalent for a semantics.

Classifying the computational complexity of argumentation semantics always has been an important issue in the field [3, 18, 20, 23, 25, 17, 37, 38, 47] (for an overview see [26, 27]), this is for good reasons. Such an analysis is of high values when it comes to implementations of reasoning systems for argumentation semantics, in particular parametrized complexity analysis [54], guides a way to algorithms performing well on practical instances. From the knowledge representation point of view, the computational complexity of an argumentation semantics is also a measure for the expressiveness of these semantics.

In order to evaluate argumentation frameworks and to compare the different semantics, it is desirable to have efficient systems at hand which are capable of dealing with a large number of argumentation semantics. As argumentation problems are in general intractable, which is also the case for $cf2$ and $stage2$ semantics, developing dedicated algorithms for the different reasoning problems is non-trivial. A promising way to implement such systems is to use a reduction method, where the given problem is translated into another language, for which sophisticated systems already exist. It turned out that the declarative programming paradigm of *Answer-Set Programming* (ASP) is especially well suited for this purpose (see [60] for an overview). Following the ASPARTIX approach [40] we will provide a fixed program for $stage2$ semantics, where the actual AF to process is given as an input database. Then the answer sets are in a one-to-one correspondence to the $stage2$ extensions of the input framework. As the encodings for $stage2$ semantics require a certain maximality check which is performed with the quite involved *saturation technique* [41], we will present an alternative

encoding making use of the novel `metasp` optimization front-end [49] for the ASP-system `gringo/claspD` to simplify the encodings.

A somehow orthogonal approach to the above mentioned reduction methods are dedicated algorithms based on argument labeling functions [21, 53, 62]. Such labelings distinguish different statuses of arguments, e.g. whether they are accepted, attacked or undecided. Now when fixing the label of one argument this has immediate implications for its neighbors. For instance if we mark one argument as accepted we have to mark all arguments attacked by it as attacked. The idea of labeling based algorithms is to use these implications to prune the search space for possible extensions. In this work we will study such propagation rules for labelings of $cf2$ and $stage2$ semantics hand in hand with dedicated labeling-based algorithms.

The contributions of this work are the following:

- In Section 2, we have a closer look at the properties of, and differences between, existing naive-based semantics stage and $cf2$. We highlight the shortcomings of these approaches in the presence of odd- and even-length cycles.
- To overcome the outlined shortcomings we study a combination of the concepts of stage and $cf2$ semantics in Section 3, where we use the SCC-recursive schema of $cf2$ semantics and instantiate the base case with stage semantics. In this way, we obtain the novel $stage2$ semantics.
- We investigate the basic properties of the novel semantics as well as its relation to the existing semantics, and we show that it solves most of the above mentioned problems. In particular, we evaluate $stage2$ semantics with the criteria proposed in [5].
- In Section 4, we analyze strong equivalence w.r.t. $stage2$ semantics, where it turns out that it is the second semantics, beside $cf2$, satisfying the succinctness property [47]. Furthermore, we investigate standard equivalence between $stage2$ and the other naive-based semantics.
- In Section 5, we study computational properties of naive-based SCC-recursive semantics. That is we complement existing results for $cf2$ semantics by:
  - Providing a complexity analysis for the standard argumentation reasoning tasks and $stage2$. We show that complexity from stage semantics carries over to $stage2$ semantics and therefore $stage2$ is located on the second level of the polynomial hierarchy.
  - Moreover, we provide an analysis of possible tractable fragments [18, 32, 33] for $cf2$ and $stage2$ which can help to improve the performance for easy instances of the in general hard problems. In particular we consider acyclic AFs, even-cycle free AFs, bipartite AFs and symmetric AFs. Finally we consider the backdoor approach [32] for augmenting these tractable fragments.
- Finally in Section 6, we concentrate on implementations. We study two orthogonal approaches for implementing abstract argumentation semantics. First we give the ASP encodings of $stage2$, where we make a distinction between the standard saturation encodings [41] and the novel `metasp` encodings [49]. Second, we give labeling-based algorithms to compute the solutions for $cf2$ and $stage2$ semantics.

2 Preliminaries

In this section we introduce the basics of abstract argumentation, the semantics we need for further investigations followed by a comparison of $cf2$ and stage semantics.

2.1 Abstract Argumentation

We first give the formal definition of abstract argumentation frameworks as introduced by Dung [22].

DEFINITION 2.1
An *argumentation framework (AF)* is a pair $F = (A, R)$, where A is a finite[1] set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a attacks b. A set $S \subseteq A$ of arguments attacks b (in F), if there is an $a \in S$, such that $(a, b) \in R$. An argument $a \in A$ is *defended* by $S \subseteq A$ (in F) iff, for each $b \in A$, it holds that, if $(b, a) \in R$, then S attacks b (in F). Moreover, given an AF F, we use A_F to denote the set of its arguments and resp. R_F to denote its attack relation.

The inherent conflicts between the arguments are resolved by selecting subsets of arguments, where a semantics σ assigns a collection of sets of arguments to an AF F. The basic requirement for all semantics is that none of the selected arguments attack each other.

DEFINITION 2.2
Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is said to be *conflict-free* (in F), if there are no $a, b \in S$, such that $(a, b) \in R$. We denote the collection of sets which are conflict-free (in F) by $cf(F)$. A set $S \subseteq A$ is maximal conflict-free or *naive*, if $S \in cf(F)$ and for each $T \in cf(F)$, $S \not\subset T$. We denote the collection of all naive sets of F by $naive(F)$. For the empty AF $F_0 = (\emptyset, \emptyset)$, we set $naive(F_0) = \{\emptyset\}$.

Towards the definition of the semantics we introduce the following formal concepts.

DEFINITION 2.3
Given an AF $F = (A, R)$, let $S \subseteq A$. The characteristic function $\mathcal{F}_F : 2^A \to 2^A$ of F is defined as $\mathcal{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$. We define the *range* of S as $S_R^+ = S \cup \{b \mid \exists a \in S, \text{ s. t. } (a, b) \in R\}$.

In the following we give brief definitions of the standard semantics in abstract argumentation [22] together with the definition of stage semantics [61]. For comprehensive surveys on argumentation semantics the interested reader is referred to [2, 6].

DEFINITION 2.4
Let $F = (A, R)$ be an AF, then $S \in cf(F)$ is

- a *stable* extension (of F), i.e. $S \in stable(F)$, if $S_R^+ = A$;
- an *admissible* extension, i.e. $S \in adm(F)$, if each $a \in S$ is defended by S;
- a *preferred* extension, i.e. $S \in prf(F)$, if $S \in adm(F)$ and for each $T \in adm(F)$, $S \not\subset T$;
- the *grounded* extension (of F), i.e. $S = grd(F)$, if it is the least fixed-point of the characteristic function $\mathcal{F}_F$;
- a *stage* extension (of F), i.e. $S \in stg(F)$, if for each $T \in cf(F)$, $S_R^+ \not\subset T_R^+$.

To illustrate the different behavior of the introduced semantics we have a look at the following example.

[1] While Dung [22] also considers infinite argument sets, we restrict ourselves to finite AFs.

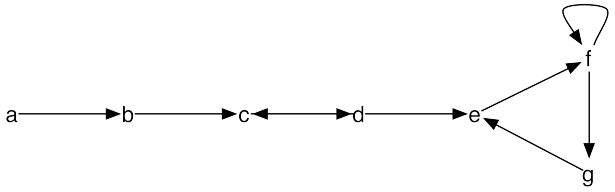


FIG. 1. The argumentation framework F from Example 2.5.

EXAMPLE 2.5
Consider the AF $F = (A, R)$, consisting of the set or arguments $A = \{a, b, c, d, e, f, g\}$ and the attack relation $R = \{(a, b), (c, b), (c, d), (d, c), (d, e), (e, f), (f, f), (f, g), (g, e)\}$ as illustrated in Figure 1. Then, the above defined semantics yield the following extensions.

- $naive(F) = \{\{b, e\}, \{b, d, g\}, \{a, d, g\}, \{a, c, e\}, \{a, c, g\}\}$;
- $stable(F) = \emptyset$, this is the only semantics where it can happen that there does not exist any extension;
- $adm(F) = \{\{\}, \{a\}, \{d\}, \{a, c\}, \{a, d\}\}$, note that the empty set is always an admissible extension;
- $prf(F) = \{\{a, c\}, \{a, d\}\}$;
- $grd(F) = \{\{a\}\}$;
- $stg(F) = \{\{a, d, g\}, \{a, c, e\}, \{a, c, g\}\}$.

$\diamond$

Next we consider *cf2* semantics, which is based on a decomposition along the strongly connected components (SCCs) of an AF depending on a given set S of arguments. The *cf2* semantics has been originally defined by Baroni and Giacomin in 2003 [4] as an approach to solve several problems which arise for frameworks with odd-length cycles. Later in 2005 they defined a general SCC-recursive schema for argumentation semantics [7] where the *cf2* semantics is also involved. The authors in [7] describe a general schema which captures all Dung semantics. The SCC-recursive schema is based on a recursive decomposition of an AF along its strongly connected components. In this work we only concentrate on one special case of this schema, the *cf2* semantics. For a more detailed discussion on the *cf2* semantics we refer to [6, 7, 47].

As mentioned before, all admissible-based semantics, i.e. semantics which build on the concept of admissible sets, cannot accept arguments out of an odd-length cycle. We already introduced stage semantics as the first semantics based on naive sets. On the basis of this requirement one can classify the semantics into admissible-, and naive-based semantics. All Dung semantics fall into the category of admissible-based semantics, whereas naive, stage as well as *cf2* and *stage2* (introduced next and in Section 3) count to the naive-based semantics. Only stable semantics falls into both groups as both $stable(F) \subseteq adm(F)$ and $stable(F) \subseteq naive(F)$ holds for any AF $F = (A, R)$.

Before we introduce the *cf2* semantics, we require some further formal machinery and concepts from graph theory. By $SCCs(F)$, we denote the set of *strongly connected components*

of an AF $F = (A, R)$, i.e. sets of vertices of the maximal strongly connected[2] sub-graphs of F; $SCCs(F)$ is thus a partition of A. Moreover, for an argument $a \in A$, we denote by $C_F(a)$ the component of F where a occurs in, i.e. the (unique) set $C \in SCCs(F)$, such that $a \in C$. AFs $F_1 = (A_1, R_1)$ and $F_2 = (A_2, R_2)$ are called *disjoint* if $A_1 \cap A_2 = \emptyset$. Moreover, the union between (not necessarily disjoint) AFs is defined as $F_1 \cup F_2 = (A_1 \cup A_2, R_1 \cup R_2)$.

It turns out to be convenient to use two different concepts to obtain sub-frameworks of AFs. Let $F = (A, R)$ be an AF and S a set of arguments. Then, $F|_S = ((A \cap S), R \cap (S \times S))$ is the *sub-framework* of F w.r.t. S, and we also use $F - S = F|_{A \setminus S}$. We note the following relation (which we use implicitly later on), for an AF F and sets S, S': $F|_{S \setminus S'} = F|_S - S' = (F - S')|_S$.

We now give the definition of the *cf2* semantics which slightly differs in notation from (but is equivalent to) the original definition in [7].

DEFINITION 2.6
Let $F = (A, R)$ be an AF and $S \subseteq A$. A $b \in A$ is *component-defeated* by S (in F), if there exists an $a \in S$, s.t. $(a, b) \in R$ and $a \notin C_F(b)$. The set of arguments component-defeated by S in F is denoted by $D_F(S)$.

Then, the *cf2* extensions of an AF are recursively defined as follows.

DEFINITION 2.7
Let $F = (A, R)$ be an argumentation framework and S a set of arguments. Then, S is a *cf2* extension of F, i.e. $S \in cf2(F)$, iff

- in case $|SCCs(F)| = 1$, then $S \in naive(F)$,
- else, for each $C \in SCCs(F) : (S \cap C) \in cf2(F|_C - D_F(S))$.

In words, an AF is recursively decomposed along its SCCs depending on a set S, where in the base case S needs to be a naive extensions. We illustrate the behavior of the introduced semantics in the following example.

EXAMPLE 2.8
Consider the following AF $F = (A, R)$ with $A = \{a, b, c\}$ and $R = \{(a, b), (b, c), (c, b), (c, c)\}$, as depicted in Figure 2. Then, the above defined semantics yield the following extensions.

- $stable(F) = \emptyset$;
- $adm(F) = \{\{\}, \{a\}\}$;
- $prf(F) = grd(F) = \{\{a\}\}$; and
- $naive(F) = stg(F) = \{\{a\}, \{b\}\}$.

Regarding stage semantics, note $S = \{b\}$ is a stage extension, as $S_R^+ = \{b, c\}$ and there is not $T \in cf(F)$ s.t. $T_R^+ \supset S_R^+$. Furthermore, $S = \{a\}$ is the only *cf2* extension of F, as F has two SCCs $C_1 = \{a\}$ and $C_2 = \{b, c\}$ and $D_F(S) = \{b\}$. Then,

- $(S \cap C_1) \in cf2(F|_{C_1})$ holds as $\{a\} \in naive(F|_{C_1})$, and
- $(S \cap C_2) \in cf2(F|_{C_2} - \{b\})$ holds as $\emptyset \in naive(F|_{\{c\}})$.

$\diamond$

[2] A directed graph (an AF) is called *strongly connected* if there is a path from each vertex to every other vertex of the graph.

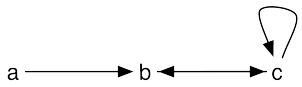


FIG. 2. The argumentation framework F from Example 2.8.

2.2 *Properties of cf2 and Stage Semantics*

To avoid the recursive computation of sub-frameworks, Gaggl and Woltran [45] introduced an alternative characterization of *cf2* semantics which requires the following concepts. The motivation for this was to design a compact Answer-set Programming (ASP) encoding which has also been incorporated in the system ASPARTIX [3] [40]. Furthermore, it facilitated the analysis of strong equivalence w.r.t. *cf2* semantics [46, 47] and the proof of general complexity results for reasoning problems regarding the *cf2* semantics [47].

The first concept describes that an AF is separated if there are no attacks between different SCCs and the separation of an AF deletes all attacks between different SCCs.

DEFINITION 2.9
An AF $F = (A, R)$ is called *separated* if for each $(a, b) \in R$, $C_F(a) = C_F(b)$. We define $[[F]] = \bigcup_{C \in SCCs(F)} F|_C$ and call $[[F]]$ the *separation* of F.

Next we consider a restricted reachability relation identifying whether there is a path from an argument to another only using arguments in a specific set B.

DEFINITION 2.10
Let $F = (A, R)$ be an AF, arguments $a, b \in A$ and $B \subseteq A$. We say that b is *reachable* in F from a *modulo* B, in symbols $a \Rightarrow_F^B b$, if there exists a path from a to b in $F|_B$, i.e. there exists a sequence $c_1, \ldots, c_n$ $(n > 1)$ of arguments such that $c_1 = a$, $c_n = b$, and $(c_i, c_{i+1}) \in R \cap (B \times B)$, for all i with $1 \leq i < n$.

The operator $\Delta_{F,S}(.)$ (applied to $D = \emptyset$) computes recursively all arguments which are attacked by the set S and can not reach their attacker without going over arguments already in $\Delta_{F,S}(.)$.

DEFINITION 2.11
For an AF $F = (A, R)$, $D \subseteq A$ and $S \subseteq A$,

$$\Delta_{F,S}(D) \quad = \quad \{a \in A \mid \exists b \in S : b \neq a, (b, a) \in R, a \not\Rightarrow_F^{A \setminus D} b\}.$$

$\Delta_{F,S}(.)$ is monotonic and thus it has a least fixed-point (lfp). With slightly abuse of notation we will denote the least fixed-point as $\Delta_{F,S}$, i.e. $\Delta_{F,S} = \Delta_{F,S}^n(\emptyset)$ with n such that $\Delta_{F,S}^n(\emptyset) = \Delta_{F,S}^{n+1}(\emptyset)$.

Now the *cf2* extensions can be characterized as follows.

PROPOSITION 2.12 ([45, 47])
For any AF F,

$$cf2(F) = \{S \mid S \in naive(F) \cap naive([[F - \Delta_{F,S}]])\}.$$

[3] See *http://rull.dbai.tuwien.ac.at:8080/ASPARTIX/* for a web front-end.

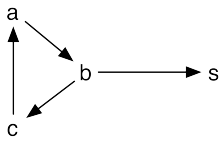


FIG. 3. Framework F.

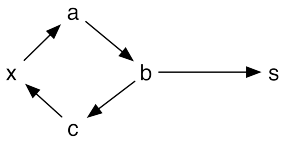


FIG. 4. Modified Framework G.

In the following we illustrate how the characterization of Proposition 2.12 can be used for identifying *cf2* extensions, for a more detailed explanation we refer to [47].

EXAMPLE 2.13
To exemplify the behavior of $\Delta_{F,S}$ and $[[F-\Delta_{F,S}]]$ let us consider the AF F of Example 2.8 (Figure 2). F has two naive sets, namely $S = \{a\}$ and $T = \{b\}$. First, we concentrate on the set S and compute $\Delta_{F,S} = \{b\}$ and $[[F - \Delta_{F,S}]] = (\{a, c\}, \{(c, c)\})$. Thus, $S \in naive([[F - \Delta_{F,S}]])$ and clearly $S \in cf2(F)$.
For T we obtain $\Delta_{F,T} = \emptyset$ and $[[F - \Delta_{F,T}]] = (A, \{(b, c), (c, b), (c, c)\})$. Now, $T \notin naive([[F - \Delta_{F,T}]])$, as there is the set $T' = \{a, b\} \supset T$ and $T' \in cf([[F - \Delta_{F,T}]])$. $\diamond$

Now, we focus on the special behavior of *cf2* and stage semantics. They are both based on naive sets, thus they are, in contrast to admissible-based semantics, capable to select arguments out of odd-length cycles as accepted. Consider the following example [57].

EXAMPLE 2.14
Suppose there are three witnesses A, B and C, where A states that B is unreliable, B states that C is unreliable and C states that A is unreliable. Moreover, C has a statement S. The graph of the framework F is illustrated in Figure 3. Any admissible-based semantics returns the empty set as its only extension. But if we have four rather than three witnesses, let's call the fourth one X, as in the AF G pictured in Figure 4, the situation changes, and the preferred extensions of G are $\{a, c, s\}$ and $\{b, x\}$. On the other hand, the naive-based semantics return $stg(F) = cf2(F) = \{\{b\}, \{a, s\}, \{c, s\}\}$ and $stg(G) = cf2(G) = \{\{a, c, s\}, \{b, x\}\}$. $\diamond$

The motivation behind selecting arguments out of an odd-length cycle is to see the arguments as different choices and to be able to choose between them. There is no need for defense, and the naive sets ensure I-maximality [5]. A special case of odd-length cycles are self-attacking arguments. One might think that it is not necessary to defend against those "broken" arguments. But, admissible-based semantics are not able to distinguish if it is necessary to defend against an attack or not. In this case it might also be desired to abandon defense and take the naive sets as the basic requirement.

So far, we only discussed the positive behavior of the *cf2* and *stg* semantics, but unfortunately there are also some disadvantages.

EXAMPLE 2.15
Consider the AF F in Figure 5. We obtain

- $stg(F) = prf(F) = stable(F) = \{\{a, c, e\}, \{b, d, f\}\}$, but
- $cf2(F) = naive(F) = \{\{a, d\}, \{b, e\}, \{c, f\}, \{a, c, e\}, \{b, d, f\}\}$.

In this example the framework consists of an even-length cycle and the *cf2* semantics produces three more extensions compared to stable semantics. This does not really coincide with

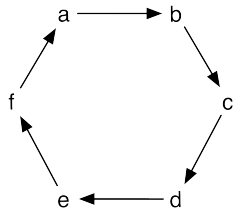


FIG. 5. AF from Example 2.15.

the motivation for a symmetric treatment of odd- and even-length cycles, as now the results differ significantly for an even-length cycle. ◇

Example 2.15 shows, that also the $cf2$ semantics has some drawbacks. Furthermore, for AFs F with odd-length cycles ≥ 9, we can also obtain $cf2(F) \neq stg(F)$. Whereas, stage semantics gives more reasonable results especially on single SCCs and still guarantees a uniform treatment of odd-, and even-length cycles. As stage semantics extends stable semantics in the sense that both semantics coincide if at least one stable extension exists, it holds that for SCCs without odd-length cycles stage semantics proposes stable extensions. Similar observations have also been made in [44]. However, for a stage extension it might be the case that even unattacked arguments are not accepted and more general, the grounded extension is not contained in every stage extension. For instance consider the AF F in Example 2.8 (Figure 2) where $grdF = \{a\}$ but $\{b\}$ is a stage extension not containing the unattacked argument a.

3 Combining Stage and $cf2$ Semantics

In the previous section, we observed that the stage semantics has a more intuitive behavior on single SCCs than $cf2$, because there $cf2$ semantics only selects the naive extensions. Whereas in general the SCC-recursive schema of $cf2$ guarantees that several evaluation criteria are fulfilled.

Our suggestion is to combine the two semantics, where we use the SCC-recursive schema of the $cf2$ semantics and instantiate the base case with stage semantics. To retain the naming introduced in [7] we denote the obtained semantics as $stage2$.

DEFINITION 3.1
Let $F = (A, R)$ be an AF and $S \subseteq A$. Then, S is a $stage2$ extension of F, i.e. $S \in stage2(F)$, iff

- in case $|SCCs(F)| = 1$, then $S \in stg(F)$,
- else, for each $C \in SCCs(F) : (S \cap C) \in stage2(F|_C - D_F(S))$.

According to the alternative characterization of $cf2$ semantics, one can also formulate $stage2$ semantics in the same way.

PROPOSITION 3.2
For any AF F,

$$stage2(F) = \{S \mid S \in naive(F) \cap stg([[F - \Delta_{F,S}]])\}.$$

Towards a proof of Proposition 3.2, we need to define two more formal concepts. First, we define the set of recursively component defeated arguments $\mathcal{RD}_F(S)$ as in [45].

DEFINITION 3.3
Let $F = (A, R)$ be an AF and $S \subseteq A$. We define the set of arguments *recursively component defeated* by S (in F) as follows:

- if $|SCCs(F)| = 1$ then $\mathcal{RD}_F(S) = \emptyset$; else,
- $\mathcal{RD}_F(S) = D_F(S) \cup \bigcup_{C \in SCCs(F)} \mathcal{RD}_{F|_C - D_F(S)}(S \cap C)$.

Next, we define the level of recursiveness a framework shows with respect to a set S of arguments.

DEFINITION 3.4
For an AF $F = (A, R)$ and $S \subseteq A$, we recursively define the *level* $\ell_F(S)$ of F w.r.t. S as follows:

- if $|SCCs(F)| = 1$ then $\ell_F(S) = 1$;
- otherwise, $\ell_F(S) = 1 + max(\{\ell_{F|_C - D_F(S)}(S \cap C) \mid C \in SCCs(F)\})$.

LEMMA 3.5
For any AF $F = (A, R)$, $S \subseteq A$. Let $\mathcal{R}'_{F,C,S} = \mathcal{RD}_{F|_C - D_F(S)}(S \cap C)$, then

$$(F|_C - D_F(S)) - \mathcal{R}'_{F,C,S} = F|_C - \mathcal{RD}_F(S).$$

PROOF. The observation has been proven in more detail in [45]. Here we just sketch the idea. We fix a $C \in SCCs(F)$. Since for each further $C' \in SCCs(F)$ (i.e. $C \neq C'$), no argument from $\mathcal{RD}_{F|_{C'} - D_F(S)}(S \cap C')$ occurs in $F|_C$, the assertion follows. ∎

Lemma 3.6 gives the first alternative characterization of $stage2$.

LEMMA 3.6
Let $F = (A, R)$ be an AF and $S \subseteq A$. Then,

$$S \in stage2(F) \text{ iff } S \in stg([[F - \mathcal{RD}_F(S)]]).$$

PROOF. We show the claim by induction over $\ell_F(S)$.

Induction base. For $\ell_F(S) = 1$, we have $|SCCs(F)| = 1$. By definition $\mathcal{RD}_F(S) = \emptyset$ and we have $[[F - \mathcal{RD}_F(S)]] = [[F]] = F$. Thus, the assertion states that $S \in stage2(F)$ iff $S \in stg(F)$ which matches the original definition for the $stage2$ semantics in case the AF has a single strongly connected component.

Induction step. Let $\ell_F(S) = n$ and assume the assertion holds for all AFs F' and sets S' with $\ell_{F'}(S') < n$. In particular, we have by definition that, for each $C \in SCCs(F)$, $\ell_{F|_C - D_F(S)}(S \cap C) < n$. By the induction hypothesis and Lemma 3.5, we thus obtain that, for each $C \in SCCs(F)$ the following holds:

$$\begin{aligned} &(S \cap C) \in stage2(F|_C - D_F(S)) \text{ iff} \\ &(S \cap C) \in stg\big([[F|_C - \mathcal{RD}_F(S)]]\big). \end{aligned} \tag{3.1}$$

We now prove the assertion. Let $S \in stage2(F)$. By definition, for each $C \in SCCs(F)$, $(S \cap C) \in stage2(F|_C - D_F(S))$. Using (3.1), we get that for each $C \in SCCs(F)$,

$(S \cap C) \in stg([[F|_C - \mathcal{RD}_F(S)]])$. By the definition of components and the semantics of stage, the following relation thus follows:

$$\bigcup_{C \in SCCs(F)} (S \cap C) \in stg\Big(\bigcup_{C \in SCCs(F)} [[F|_C - \mathcal{RD}_F(S)]]\Big).$$

Since $S = \bigcup_{C \in SCCs(F)}(S \cap C)$ and due to [45], $\bigcup_{C \in SCCs(F)}[[F|_C - \mathcal{RD}_F(S)]] = [[F - \mathcal{RD}_F(S)]]$, we arrive at $S \in stg([[F - \mathcal{RD}_F(S)]])$ as desired. The other direction is by essentially the same arguments. ∎

PROOF. [Proof of Proposition 3.2] The result holds by the following observations. Due to Lemma 3.6, $S \in stage2(F)$ iff $S \in stg([[F - \mathcal{RD}_F(S)]])$. Moreover, due to [45], for any $S \in cf(F)$, $\Delta_{F,S} = \mathcal{RD}_F(S)$. Finally, $S \in stage2(F)$ implies $S \in naive(F)$. ∎

We obtain for the framework F of Example 2.8, $stage2(F) = cf2(F) = \{\{a\}\}$, and for the AF of Example 2.15, $stage2(F) = stg(F) = \{\{a, c, e\}, \{b, d, f\}\}$.

3.1 Comparison of stage2 with other Semantics

The novel *stage2* semantics is clearly a naive-based semantics due to the way it is defined. In this section we compare *stage2* with other naive-based semantics w.r.t. the $\subseteq$-relations between the sets of extensions. Furthermore, we consider coherent AFs, as stage semantics also coincides with stable and preferred on these frameworks but *cf2* does not.

We start with stage and *stage2* semantics which are in general incomparable w.r.t. set inclusion. For instance, consider the following example.

EXAMPLE 3.7
Let $F = (A, R)$ as illustrated in Figure 6. Then, the naive sets of F are $\{a, d\}$, $\{a, e\}$, $\{b, d\}$ and $\{b, e\}$. We consider first stage semantics, therefore we compute the range of each naive set.

- $\{b, d\}^+_R = \{a, b, c, d, e\}$,
- $\{b, e\}^+_R = \{a, b, c, e, f\}$,
- $\{a, d\}^+_R = \{a, b, d, e\} \subset \{b, e\}^+_R$,
- $\{a, e\}^+_R = \{a, b, e, f\} \subset \{b, e\}^+_R$.

Thus, $stg(F) = \{\{b, d\}, \{b, e\}\}$.

The *stage2* extensions are $\{a, d\}$ and $\{b, d\}$ which are computed as follows.

- For $S_1 = \{a, d\}$, $\Delta_{F,S_1} = \{e\}$ and $S_1 \in stg([[F - \Delta_{F,S_1}]])$. Thus, $S_1 \in stage2(F)$.
- For $S_2 = \{b, d\}$, $\Delta_{F,S_2} = \{c, e\}$ and $S_2 \in stg([[F - \Delta_{F,S_2}]])$. Thus, $S_2 \in stage2(F)$.
- For $S_3 = \{a, e\}$, $\Delta_{F,S_3} = \{f\}$ but $S_3 \notin stg([[F - \Delta_{F,S_3}]])$ because $S^+_{3_{R'}} = \{a, b, e\}$ and there is the set $T \in naive(F')$ with $T = \{a, d, e\}$ and $T^+_{R'} = \{a, b, d, e\} \supset S^+_{3_{R'}}$ where $F' = [[F - \Delta_{F,S_3}]]$. Hence, $S_3 \notin stage2(F)$.
- For $S_4 = \{b, e\}$, $\Delta_{F,S_4} = \{c, f\}$ but $S_4 \notin stg([[F - \Delta_{F,S_4}]])$ because $S^+_{4_{R''}} = \{a, b, e\}$ and there is the set $T \in naive(F'')$ with $T = \{a, d, e\}$ and $T^+_{R''} = \{a, b, d, e\} \supset S^+_{4_{R''}}$ where $F'' = [[F - \Delta_{F,S_4}]]$. Hence, $S_4 \notin stage2(F)$.

◇

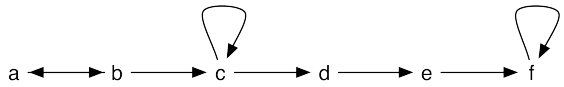


FIG. 6. Framework F from Example 3.7.

Now, we consider the relation between $cf2$ and $stage2$ semantics. By Example 2.15 we know that there are AFs with $cf2(F) \not\subseteq stage2(F)$.

PROPOSITION 3.8
For any AF $F = (A, R)$, $stage2(F) \subseteq cf2(F)$.

PROOF. Consider a set $S \in stage2(F)$. By Proposition 3.2, $S \in naive(F) \cap stg([[F - \Delta_{F,S}]])$. Now using that for every AF G, $stg(G) \subseteq naive(G)$ we obtain $S \in naive(F) \cap naive([[F - \Delta_{F,S}]])$. By Proposition 2.12, $S \in cf2(F)$. ∎

Next, we study the relations between stable and $stage2$ semantics.

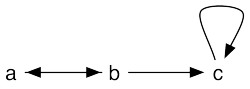
PROPOSITION 3.9
For any AF $F = (A, R)$, $stable(F) \subseteq stage2(F)$.

PROOF. Consider $E \in stable(F)$, then we know that $E \in naive(F)$ and for each $a \in A \setminus E$ there exists $b \in E$ such that $(b, a) \in R$. Hence, $a \in E^+_{R_F}$. It remains to show that $E \in stg([[F - \Delta_{F,E}]])$. We show the stronger statement $E \in stable([[F - \Delta_{F,E}]])$.

To this end, let $F' = F - \Delta_{F,E}$ and $F'' = [[F - \Delta_{F,E}]]$, we have either $a \in \Delta_{F,E}$ or $a \in A_{F'}$. For $a \in A_{F'} = A_{F''}$, we need to show that $a \in E^+_{R_{F''}}$. If $a \in E$ clearly $a \in E^+_{R_{F''}}$, hence we consider $a \in A_{F'} \setminus E$. As E is stable there exists $b \in E$ such that $(b, a) \in R_{F'}$. Now as $a \notin \Delta_{F,E}$, by Definition 2.11 we know that $a \Rightarrow_F^{A \setminus \Delta_{F,E}} b$. In other words a, b are in the same SCC of F' and thus $(b, a) \in R_{F''}$. Hence, for every $a \in A_{F''} \setminus E$ there is an argument $b \in E$ such that $(b, a) \in R_{F''}$, hence $E \in stable(F'')$. As for any AF G $stable(G) \subseteq stg(G)$, it follows that $E \in stg(F'')$. Thus, by Proposition 3.2, $E \in stage2(F)$. ∎

Figure 7 gives an overview of the relations between naive-based semantics. An arrow from semantics σ to semantics τ encodes that each σ-extension is also a τ-extension. Furthermore, if there is no directed path from σ to τ, then one can construct AFs with a σ-extension that is not a τ-extension.

If an AF possesses at least one stable extension, stage coincides with stable semantics. Obviously, this does not hold for $stage2$ semantics, for instance consider the AF $F = (\{a, b, c\}, \{(a, b), (b, a), (b, c), (c, c)\})$.



We obtain $stage2(F) = \{\{a\}, \{b\}\}$ and $stable(F) = \{\{b\}\}$. However, these semantics comply with each other in *coherent* AFs, i.e. AFs where stable and preferred semantics coincide.

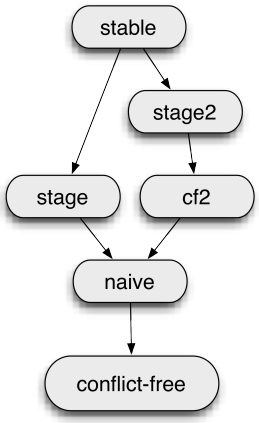


FIG. 7. Relations between naive-based semantics.

PROPOSITION 3.10
For any coherent AF F, $stable(F) = stg(F) = stage2(F)$.

PROOF. By Proposition 3.9, $stable(F) \subseteq stage2(F)$ and thus it only remains to show that also $stable(F) \supseteq stage2(F)$ holds. Let us first consider the case where F consists of a single SCC. Then, $stage2$ semantics coincides with stage semantics and as F is coherent also with stable semantics.

Now, let this be our induction base, and let us assume the claim holds for AFs of size $< n$. Let us consider an AF F of size n with $(C_i)_{1\leq i\leq m}$ being the SCCs of F, such that there is no attack from C_i to C_j for $j < i$. If $m = 1$ we are in the base-case, hence let us assume that $m \geq 2$. Consider $S \in stage2(F)$ and $S_1 = S \cap \bigcup_{1\leq i<m} C_i$, $S_2 = S \cap C_m$. By definition of $stage2$ we know that $S_1 \in stage2(F - C_m)$ and $S_2 \in stage2(F|_{C_m} - S_1^+)$. Note, $S_1 \cap S_2 = \emptyset$. By assumption, F is coherent and it is easy to see that also $F - C_m$ is coherent. This is because prf-semantics satisfies directionality [5] and thus $prf(F - C_m) = prf(F) \cap \bigcup_{1\leq i<m} C_i = stable(F) \cap \bigcup_{1\leq i<m} C_i$. Now as each $E \in prf(F - C_m)$ is contained in an $E' \in stable(F)$ it must be also stable in $F - C_m$, i.e. $F - C_m$ is coherent. Hence, by the induction hypothesis, $stable(F - C_m) = prf(F - C_m) = stage2(F - C_m)$.

Next, we show that also $F|_{C_m} - S_1^+$ is coherent. By definition, $stable(F) \subseteq prf(F)$. Now, consider an extension $E_2 \in prf(F|_{C_m} - S_1^+)$ and the extension $S_1 \cup E_2$ of F. First, by construction $S_1 \cup E_2$ is conflict-free and, as S_1 is not attacked by arguments in C_m, also $S_1 \cup E_2$ defends all arguments in S_1. Second, as $S_1 \in stable(F - C_m)$ we have that S_1 defends E_2 against attacks from $\bigcup_{1\leq i<m} C_i$ and, as $E_2 \in adm(F|_{C_m} - S_1^+)$, E_2 defend itself against the remaining attacks. Finally by the maximality of E_2 in $adm(F|_{C_m} - S_1^+)$ we obtain $(S_1 \cup E_2) \in prf(F)$. Now, as F is coherent also $(S_1 \cup E_2) \in stable(F)$ and thus, $E_2 \in stable(F|_{C_m} - S_1^+)$. Hence, $F|_{C_m} - S_1^+$ is coherent and again we can use the induction hypothesis.

Finally, we obtain $S_1 \in stable(F - C_m)$ and $S_2 \in stable(F|_{C_m} - S_1^+)$, combining these results we get $S \in stable(F)$. ∎

Notice, Proposition 3.10 implies that on coherent AFs $stage2$ semantics coincides with

preferred, stage and semi-stable [17] semantics, because on coherent AFs all these semantics coincide with stable semantics.

3.2 Extension Evaluation Criteria

Several general criteria for the evaluation of argumentation semantics have been proposed in [5]. In this subsection we analyze the criteria relevant for naive-based semantics.[4]

DEFINITION 3.11
A semantics σ satisfies

- the *I-maximality criterion* if for each AF $F = (A, R)$, and for each $S_1, S_2 \in \sigma(F)$, if $S_1 \subseteq S_2$ then $S_1 = S_2$;
- the *reinstatement criterion* if for each AF $F = (A, R)$, and for each $S \in \sigma(F)$, a defended by S implies $a \in S$.
- the *weak reinstatement criterion*, if for each $F = (A, R)$, and for each $S \in \sigma(F)$: $grd(F) \subseteq S$;
- the *$\mathcal{CF}$-reinstatement criterion*, if for each $F = (A, R)$, for each $S \in \sigma(F), \forall b : (b, a) \in R, \exists c \in S : (c, b) \in R$ and $S \cup \{a\} \in cf(F) \Rightarrow a \in S$.
- the *directionality criterion* if for each $F = (A, R)$, and for each unattacked set of arguments $U \subseteq A$ (s. t. $\forall a \in A \setminus U$ there is no $b \in U$ with $(a, b) \in R$), it holds that $\sigma(F|_U) = \{(S \cap U) \mid S \in \sigma(F)\}$.

We start with some general properties of naive-based semantics.

PROPOSITION 3.12
I-maximality and $\mathcal{CF}$-reinstatement are satisfied by each semantics σ with $\sigma(F) \subseteq naive(F)$.

PROOF. Clearly $naive$ semantics satisfies both I-maximality and $\mathcal{CF}$-reinstatement. A set E which is $\subseteq$-maximal in $naive(F)$ is also maximal in each subset of $naive(F)$ and thus, σ satisfies I-maximality. $\mathcal{CF}$-reinstatement is a property defined on single extensions, and as each σ-extension is also a $naive$ extension, $\mathcal{CF}$-reinstatement is satisfied . ∎

Among the naive-based semantics, only stable semantics satisfies the reinstatement property, which is due to the fact that it is also an admissible-based semantics.

PROPOSITION 3.13
The reinstatement property is not satisfied by semantics which can select non-empty conflict-free subsets out of odd-length cycles.

PROOF. Consider an odd length cycle $F = (\{a_0, \ldots, a_{n-1}\}, \{(a_i, a_{i+1 \bmod n}) \mid 0 \leq i \leq n-1\})$ with n being an odd integer. So this is a cycle of length n where a_i attacks a_{i+1} and $a_n = a_0$. We claim that no $E \in cf(F)$ and $E \neq \emptyset$ satisfies the reinstatement property. Now, towards a contradiction let us assume there exists a nonempty $E \in cf(F)$ satisfying the reinstatement property. W.l.o.g. assume that $a_1 \in E$. Then a_3 is defended and by assumption $a_3 \in E$. But then also a_5 is defended, and by induction it follows that $a_i \in E$ if i is odd. Hence also $a_n \in E$, but $\{a_1, a_n\} \subseteq E$ contradicts that E is conflict-free in F. ∎

Hence, when considering naive-based semantics we are usually interested in weaker forms of reinstatement, namely the weak- or $\mathcal{CF}$-reinstatement.

[4] Some of the criteria have been reformulated to fit to the notation of this paper, but can be easily shown to be equivalent to those in [5].

PROPOSITION 3.14
The weak reinstatement and directionality criterion are not satisfied by naive and stage semantics.

PROOF. Consider the AF F from Example 2.8. We obtain $naive(F) = stg(F) = \{\{a\}, \{b\}\}$ and the grounded extension $G = \{a\}$. Then, the weak reinstatement criterion is not satisfied because $G \not\subseteq \{b\}$. Now let us consider directionality and the sub-framework $F|_{\{a\}}$. Then $stg(F|_{\{a\}}) = \{\{a\}\}$ but $\{(\{a\} \cap S) \mid S \in stg(F)\} = \{\emptyset, \{a\}\}$, contradicting the directionality criterion. ∎

PROPOSITION 3.15
The weak reinstatement criterion is satisfied by $stage2$ semantics.

PROOF. Let $F = (A, R)$ and $E \in grd(F)$. Due to [7], for any AF F and any $S \in cf2(F)$, $E \subseteq S$. From Proposition 3.8 we know that for any AF G, $stage2(G) \subseteq cf2(G)$. It follows that for any extension $S \in stage2(F)$, $S \in cf2(F)$ and $E \subseteq S$. ∎

We sum up the results for the novel $stage2$ semantics.

- Directionality is satisfied. Due to [5], any SCC-recursive semantics σ that admits at least one extension for any AF satisfies the directionality criterion. As the $stage2$ semantics has been directly defined in terms of the SCC-recursive schema, the directionality criterion is indeed satisfied.
- I-maximality and $\mathcal{CF}$-reinstatement are satisfied, see Proposition 3.12.
- Reinstatement is not satisfied, see Proposition 3.13.
- Weak reinstatement is satisfied, see Proposition 3.15.

We summarize the evaluation criteria w.r.t. naive-based semantics in Table 1.

Finally, we mention that directionality implies the properties crash-resistance and non-interference (cf. [2]) which both are violated by stable semantics, but satisfied by $stage2$.

	naive	*stable*	*stg*	*cf2*	*stage2*
I-max.	Yes	Yes	Yes	Yes	Yes
Reinst.	No	Yes	No	No	No
Weak reinst.	No	Yes	No	Yes	Yes
$\mathcal{CF}$-reinst.	Yes	Yes	Yes	Yes	Yes
Direct.	No	No	No	Yes	Yes

TABLE 1. Evaluation Criteria w.r.t. Naive-based Semantics.

4 Notions of Equivalence

Argumentation can be understood as a dynamic reasoning process, i.e. it is in particular useful to know the effects additional information causes with respect to a certain semantics. Accordingly, one can identify the information which does not contribute to the results no matter which changes are performed. In other words, we are interested in so-called *kernels*

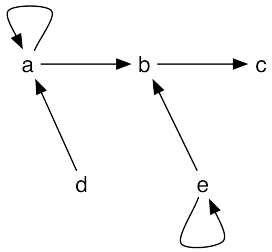


FIG. 8. AF F from Example 4.1.

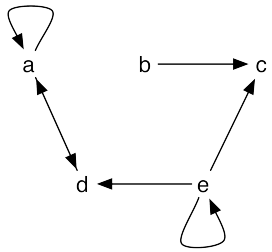


FIG. 9. AF G from Example 4.1.

of frameworks, where two frameworks with the same kernel are equally affected by all kind of newly added information in the sense that they always produce an equal outcome.

The concept of *strong equivalence* for argumentation frameworks captures this intuition and has been analyzed for several semantics mostly based on the concept of admissibility by Oikarinen and Woltran in [56], and for $cf2$ semantics in [47]. It turned out that strong equivalence w.r.t. admissible, preferred, semi-stable and ideal semantics is exactly the same concept, while stable, stage, complete, and grounded semantics require distinct kernels. Interestingly, in the case of $cf2$ semantics, strong equivalence coincides with syntactic equivalence [47], hence there are no redundant attacks at all. This special behavior has been made explicit with the *succinctness property* [47]. If a semantics σ satisfies the succinctness property, then for every framework F, all its attacks contribute to the evaluation of at least one framework F' containing F.

In the following we will introduce the necessary concepts for standard and strong equivalence as well as the succinctness property. Then, we analyze standard equivalence between $stage2$ and the other naive-based semantics, followed by the investigation of strong equivalence w.r.t. $stage2$ semantics.

If two distinct AFs possess the same extensions w.r.t. a semantics σ we speak about *(standard) equivalence*. Consider the following example.

EXAMPLE 4.1
The AFs F and G are illustrated in Figures 8 and 9. The two AFs differ in the attacks $(a, b), (a, d), (e, d), (e, b)$ and (e, c). Both AFs have no stable extension, hence $stable(F) = stable(G) = \emptyset$. Thus, F and G are equivalent with respect to stable semantics. $\diamond$

Strong equivalence for argumentation frameworks not only requires that two AFs have the same extensions under a specific semantics but also, if the frameworks are augmented with additional information, they still possess the same extensions (under the semantics). The following example illustrates this for stable semantics.

EXAMPLE 4.2
Consider the AFs F and G from Example 4.1 (Figures 8 and 9). We add the new AF $H = (\{b, e\}, \{(b, e)\})$ to each of them. Then, they still have the same stable extensions $stable(F \cup H) = stable(G \cup H) = \{\{b, d\}\}$, as highlighted in the graphs of Figures 10 and 11. Furthermore, it can be shown that no matter which framework H one adds to F and G they will always posses the same stable extensions. $\diamond$

The concept of strong equivalence for argumentation frameworks, as introduced by Oikarinen and Woltran in [56], meets exactly the behavior described in Example 4.2. The formal

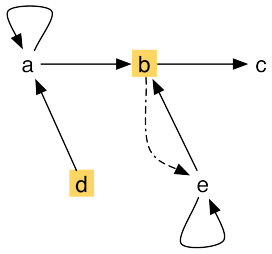


FIG. 10. $F \cup H$ from Example 4.2.

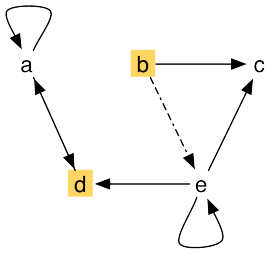


FIG. 11. $G \cup H$ from Example 4.2.

definition is as follows.

DEFINITION 4.3
Two AFs F and G are *strongly equivalent* to each other w.r.t. a semantics σ, in symbols $F \equiv_s^\sigma G$, iff for each AF H, $\sigma(F \cup H) = \sigma(G \cup H)$.

By definition, $F \equiv_s^\sigma G$ implies $\sigma(F) = \sigma(G)$, but the other direction is not true in general.

To identify to which extent attacks contribute in terms of a given semantics, the *succinctness property* has been introduced in [47]. In contrast to strong equivalence which considers particular AFs, the succinctness property denotes a general property for argumentation semantics. Hence, it is independent of the specific instantiation method.

Before we give the definition of the succinctness property, we define what we mean with redundant attacks; for AFs $F = (A, R)$ and $F' = (A', R')$ we write $F \subseteq F'$ to denote that $A \subseteq A'$ and $R \subseteq R'$ jointly hold. Moreover, we use $F \setminus (a, b)$ as a shorthand for the framework $(A, R \setminus \{(a, b)\})$.

DEFINITION 4.4
For an AF $F = (A, R)$ and semantics σ we call an attack $(a, b) \in R$ *redundant in* F *w.r.t.* σ if for all F' with $F \subseteq F'$, $\sigma(F') = \sigma(F' \setminus (a, b))$.

DEFINITION 4.5
An argumentation semantics σ satisfies the *succinctness property* or is *maximal succinct* iff no AF contains a redundant attack w.r.t. σ.

The following proposition gives the link between the succinctness property and strong equivalence.

PROPOSITION 4.6 ([47])
An argumentation semantics σ satisfies the succinctness property iff for any AFs F, G with $A_F = A_G$: $F \equiv_s^\sigma G \Leftrightarrow F = G$.

We point out that for all semantics considered so far, strong equivalence for AFs implies that the AFs have the same arguments. Thus, for the semantics under our consideration, one can drop the condition $A_F = A_G$ in the above proposition.

4.1 Standard Equivalence

We take a closer look at the relations between *stage2* and the other naive-based semantics in terms of equivalence. Especially we are interested if equivalence w.r.t. a semantics implies

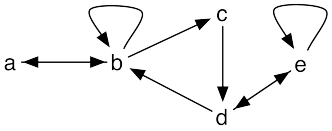


FIG. 12. AF F from Example 4.7.

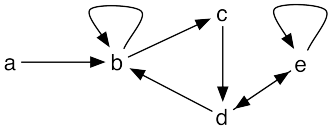


FIG. 13. AF G from Example 4.7.

equivalence w.r.t. another semantics? In the next examples we show that there is no particular relation between naive, stage, stable, *cf2* and *stage2* semantics in terms of standard equivalence.

First, we consider AFs F and G such that $\sigma(F) = \sigma(G) \not\Rightarrow \theta(F) = \theta(G)$, where $\sigma \in \{naive, stg, stable\}$ and $\theta \in \{cf2, stage2\}$.

EXAMPLE 4.7
Let F and G be as illustrated in Figures 12 and 13. The only difference between those two AFs is the attack (b, a) which is contained in F but not in G. This has the effect that the framework F consists of a single SCC; and thus $cf2(F) = naive(F)$ and $stage2(F) = stg(F)$. We have $stable(F) = stable(G) = \emptyset$ and $stg(F) = stg(G) = \{\{a, c\}, \{a, d\}\}$. Furthermore, $naive(F) = naive(G) = \{\{a, c\}, \{a, d\}\}$. However, we have $cf2(F) = stage2(F) = \{\{a, c\}, \{a, d\}\}$ and $cf2(G) = stage2(G) = \{\{a, c\}\}$.

In the following we briefly show why $S = \{a, d\}$ is not a *cf2* extension of G. First, $\Delta_{G,S} = \{b\}$, so we obtain

$$[[G - \Delta_{G,S}]] = (\{a, c, d, e\}, \{(d, e), (e, d), (e, e)\}),$$

and thus $naive([[G - \Delta_{G,S}]]) = \{\{a, c, d\}\}$. For *stage2* and the set S we observe $stg([[G - \Delta_{G,S}]]) = \{\{a, c, d\}\}$, and thus S is no *stage2* extension of G. Hence,

$$\sigma(F) = \sigma(G) \not\Rightarrow \theta(F) = \theta(G)$$

for $\sigma \in \{naive, stg, stable\}$, and $\theta \in \{cf2, stage2\}$ as desired. ◇

The next example shows that $\sigma(F) = \sigma(G) \not\Rightarrow \theta(F) = \theta(G)$, where $\sigma \in \{naive, cf2, stage2\}$ and $\theta \in \{stg, stable\}$.

EXAMPLE 4.8
The AFs F and G are illustrated in Figures 14 and 15. Then, we obtain

- $naive(F) = naive(G) = \{\{a\}, \{b\}\}$,
- $cf2(F) = cf2(G) = \{\{a\}\}$ and
- $stage2(F) = stage2(G) = \{\{a\}\}$.

On the other side

- $stable(F) = \emptyset \neq stable(G) = \{\{a\}\}$ and
- $stg(F) = \{\{a\}, \{b\}\} \neq stg(G) = \{\{a\}\}$.

Thus, we showed that $\sigma(F) = \sigma(G) \not\Rightarrow \theta(F) = \theta(G)$, for $\sigma \in \{naive, cf2, stage2\}$ and $\theta \in \{stg, stable\}$. ◇

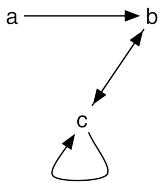


FIG. 14. AF F from Example 4.8.

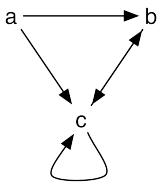


FIG. 15. AF G from Example 4.8.

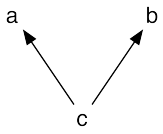


FIG. 16. AF F from Example 4.9.

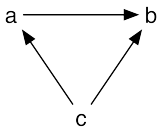


FIG. 17. AF G from Example 4.9.

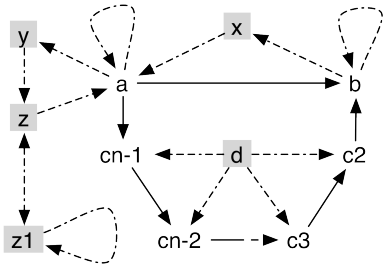


FIG. 18. $F \cup H$.

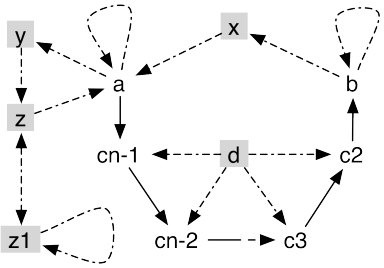


FIG. 19. $G \cup H$.

Now, we provide frameworks F and G such that $\sigma(F) = \sigma(G) \not\Rightarrow naive(F) = naive(G)$, where $\sigma \in \{stable, stg, cf2, stage2\}$.

EXAMPLE 4.9
Let the AFs F and G be as in Figures 16 and 17. Then, we have $\sigma(F) = \sigma(G) = \{\{c\}\}$, where $\sigma \in \{stable, stg, cf2, stage2\}$ but $naive(F) = \{\{a, b\}, \{c\}\}$ and $naive(G) = \{\{a\}, \{b\}, \{c\}\}$. ◇

4.2 Strong Equivalence

In [46, 47], it has been shown that for $cf2$ semantics, strong equivalence coincides with syntactic equivalence. In other words, there are no redundant patterns at all. In the following, we show that the same also holds for $stage2$ semantics.

THEOREM 4.10
For any AFs F and G, $F \equiv_s^{stage2} G$ iff $F = G$.

PROOF. Since for any AFs $F = G$ obviously implies for all AFs H, $stage2(F \cup H) =$

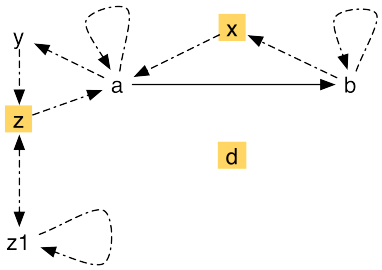


FIG. 20. $[[(F \cup H) - \Delta_{F \cup H,E}]]$.

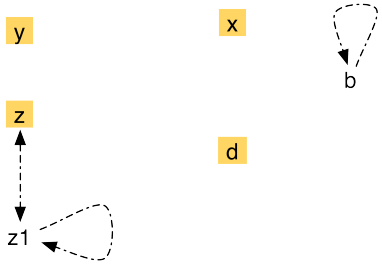


FIG. 21. $[[(G \cup H) - \Delta_{G \cup H,E}]]$.

$stage2(G \cup H)$, we only have to show that if $F \neq G$ there exists an AF H such that $stage2(F \cup H) \neq stage2(G \cup H)$.

For any two AFs F and G, strong equivalence w.r.t. naive-based semantics requires that the AFs coincide with the arguments and the self-attacks [46]. We thus assume that $A = A(F) = A(G)$ and $(a,a) \in R(F)$ iff $(a,a) \in R(G)$, for each $a \in A$. Let us thus suppose w.l.o.g. an attack $(a,b) \in R(F) \setminus R(G)$ and consider the AF

$$\begin{aligned} H \quad = \quad & (A \cup \{d,x,y,z,z1\}, \{(a,a),(b,b),(b,x),(x,a),(a,y),(y,z),(z,a), \\ & (z,z1),(z1,z),(z1,z1),(d,c) \mid c \in A \setminus \{a,b\}\}), \end{aligned}$$

see also Figures 18 and 19 for illustration.

Then, for $E = \{d,x,z\}$, we have $E \in stage2(F \cup H)$ but $E \notin stage2(G \cup H)$. To show that $E \in stage2(F \cup H)$, we first compute $\Delta_{F \cup H,E} = \{c \mid c \in A \setminus \{a,b\}\}$. Thus, we have two SCCs left in the instance $F' = [[(F \cup H) - \Delta_{F \cup H,E}]]$, namely $C_1 = \{d\}$ and $C_2 = \{a,b,x,y,z,z1\}$ as illustrated in Figure 20. Furthermore, all attacks between the arguments of C_2 are preserved, and we obtain that $E \in stg(F')$, and as $E \in naive(F \cup H)$, $E \in stage2(F \cup H)$ follows.

On the other hand, we obtain $\Delta_{G \cup H,E} = \{a\} \cup \{c \mid c \in A \setminus \{a,b\}\}$, and the instance $G' = [[(G \cup H) - \Delta_{G \cup H,E}]]$ consists of five SCCs, namely $C_1 = \{d\}$, $C_2 = \{b\}$, $C_3 = \{x\}$, $C_4 = \{y\}$ and $C_5 = \{z,z1\}$, with b and $z1$ being self-attacking as illustrated in Figure 21. Thus, the set $T = \{d,x,y,z\} \supset E$ is conflict-free in G' and $T^+_{R(G')} \supset E^+_{R(G')}$. Therefore, we obtain $E \notin stg(G')$, and hence, $E \notin stage2(G \cup H)$. $F \not\equiv^{stage2}_s G$ follows. ∎

By Theorem 4.6 and Theorem 4.10 the following result is obvious.

COROLLARY 4.11
The *stage2* semantics satisfies the succinctness property.

No matter which AFs $F \neq G$ are given, we can always construct a framework H such that $stage2(F \cup H) \neq stage2(G \cup H)$. In particular, we can always add new arguments and attacks such that the missing attack in one of the original frameworks leads to different SCCs in the modified ones and therefore to different *stage2* extensions, when suitably augmenting the two AFs under comparison. Till now, *stage2* is the second semantics beside *cf2*, where strong equivalence coincides with syntactic equivalence. This can be seen as another special property of these semantics which is met by the succinctness property.

	Ver_σ	$Cred_\sigma$	$Skept_\sigma$	$Exists_\sigma^{\neg\emptyset}$
naive	in P	in P	in P	in P
grd	P-c	P-c	P-c	in P
stable	in P	NP-c	coNP-c	NP-c
adm	in P	NP-c	trivial	NP-c
comp	in P	NP-c	P-c	NP-c
cf2	in P	NP-c	coNP-c	in P
prf	coNP-c	NP-c	Π_2^P-c	NP-c
stg	coNP-c	Σ_2^P-c	Π_2^P-c	in P
sem	coNP-c	Σ_2^P-c	Π_2^P-c	NP-c

TABLE 2. Complexity of decision problems ($\mathcal{C}$-c denotes completeness for class $\mathcal{C}$).

5 Computational Properties

In this section we study the computational complexity of the typical reasoning tasks in abstract argumentation for the *stage2* semantics and complement existing results for *cf2* semantics. We first consider these tasks on arbitrary AFs, give the general complexity of *stage2* semantics and compare it with results from the literature. Second, we consider graph classes which are typically tractable for abstract argumentation and study the complexity of reasoning with *cf2* and *stage2* side by side. Finally we consider the backdoor approach [32] for extending these tractable fragments.

We assume the reader has knowledge about standard complexity classes, i.e. P, NP and coNP. Nevertheless we briefly recapitulate the concept of oracle machines and some related complexity classes. Thus let $\mathcal{C}$ notate some complexity class. By a $\mathcal{C}$-oracle machine we mean a (polynomial time) Turing machine which can access an oracle that decides a given (sub)-problem in $\mathcal{C}$ within one step. We denote such machines as $\mathrm{P}^{\mathcal{C}}$ if the underlying Turing machine is deterministic; and $\mathrm{NP}^{\mathcal{C}}$ if the underlying Turing machine is nondeterministic. We are now ready to define specific complexity classes using NP-oracles. First the class $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$, denotes the problems which can be decided by a nondeterministic polynomial time algorithm that has (unrestricted) access to an NP-oracle. The class $\Pi_2^P = \mathrm{coNP}^{\mathrm{NP}}$ is defined as the complementary class of Σ_2^P, i.e. $\Pi_2^P = \mathrm{co}\Sigma_2^P$.

The typical problems of interest in abstract argumentation are the following decision problems for given $F = (A, R)$, a semantics σ, $a \in A$ and $S \subseteq A$:

- Verification Ver_σ: is $S \in \sigma(F)$?
- Credulous acceptance $Cred_\sigma$: is a contained in at least one σ extension of F?
- Skeptical acceptance $Skept_\sigma$: is a contained in every σ extension of F?
- Non-emptiness $Exists_\sigma^{\neg\emptyset}$: is there any $S \in \sigma(F)$ for which $S \neq \emptyset$?

In Table 2 known complexity results for various argumentation semantics are summarized [17, 18, 20, 23, 25, 37, 38, 47]. For a detailed discussion of them we refer to [26, 27].

We briefly review the hardness results for *cf2* semantics presented in [47]. The hardness proofs of $Cred_{cf2}$ and $Skept_{cf2}$ are based on the following reduction from propositional formulas in conjunctive normal form (CNF) to AFs as in [20, 23].

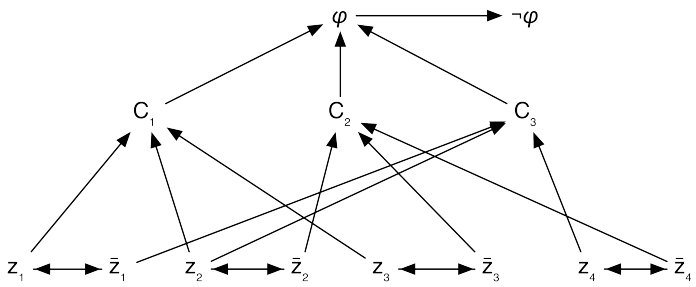


FIG. 22. AF F_φ for the example 3-CNF formula φ.

DEFINITION 5.1
Given a 3-CNF formula $\varphi = \bigwedge_{j=1}^{m} C_j$ over atoms Z with $C_j = l_{j1} \vee l_{j2} \vee l_{j3}$ $(1 \leq j \leq m)$ the corresponding AF $F_\varphi = (A_\varphi, R_\varphi)$ is built as follows.

$$\begin{aligned} A_\varphi &= Z \cup \bar{Z} \cup \{C_1, \ldots, C_m\} \cup \{\varphi, \neg\varphi\} \\ R_\varphi &= \{(z, \bar{z}), (\bar{z}, z) \mid z \in Z\} \cup \{(C_j, \varphi) \mid j \in \{1, \ldots, m\}\} \cup \{(\varphi, \neg\varphi)\} \cup \\ &\quad \{(z, C_j) \mid j \in \{1, \ldots, m\}, z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \cup \\ &\quad \{(\bar{z}, C_j) \mid j \in \{1, \ldots, m\}, \neg z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \end{aligned}$$

In [47] it is shown that a formula φ is satisfiable iff φ is credulously accepted in F_φ (w.r.t. *cf2*) iff $\neg\varphi$ is not skeptically accepted in F_φ (w.r.t. *cf2*), which proves NP (resp. coNP) hardness of the corresponding reasoning problems.

5.1 General Complexity of stage2 Semantics

We now give an exact complexity case of *stage2* semantics when considering arbitrary AFs.

THEOREM 5.2
For *stage2* semantics the following holds

- Ver_{stage2} is coNP-complete;
- $Cred_{stage2}$ is Σ_2^{P}-complete;
- $Skept_{stage2}$ is Π_2^{P}-complete;
- $Exists^{\neg\emptyset}$ *stage2* is in P.

PROOF. We first consider the membership part starting with Ver_{stage2}. Given an AF $F = (A, R)$ a set E of arguments, by Proposition 3.2 we have to check whether $E \in naive(F)$ (which can be done in P), and whether $E \in stg([[F - \Delta_{F,S}]])$. As $[[F - \Delta_{F,S}]]$ can be constructed in polynomial time and $Ver_{stg} \in$ coNP [37] also $Ver_{stage2} \in$ coNP.

The problems $Cred_{stage2}$ and $Skept_{stage2}$ can be solved by a standard guess and check algorithm, i.e. guessing an extension containing the argument (resp. not containing the argument) and using an NP-oracle to verify the extension. Thus $Cred_{stage2} \in \Sigma_2^{\mathrm{P}}$ and $Skept_{stage2} \in \Pi_2^{\mathrm{P}}$ follows.

(a) AF F (b) AF $\mathcal{R}(F)$

FIG. 23: An illustration of the reduction in proof of Theorem 5.2, with $stg(F) = \{\{b\}\} = stage2(F)$.

For the hardness part we give a reduction $\mathcal{R}$ mapping argumentation frameworks to argumentation frameworks, such that for each AF F it holds that $stg(F) = stage2(\mathcal{R}(F))$.[5] The hardness results then follow from the corresponding hardness results for stage semantics [37]. Given an AF $F = (A, R)$ we define $\mathcal{R}(F) = (A^*, R^*)$ with $A^* = A \cup \{t\}$ and $R^* = R \cup \{(t,t)\} \cup \{(t,a),(a,t) \mid a \in A\})$, where t is a fresh argument. We illustrate the construction in Figure 23. Then, $\mathcal{R}(F)$ consists of a single SCC and hence $stg(\mathcal{R}(F)) = stage2(\mathcal{R}(F))$. It remains to show that $stg(F) = stg(\mathcal{R}(F))$. First, as $(t,t) \in R^*$, the argument t can not be contained in a stage extension. Furthermore, the reduction $\mathcal{R}$ does not modify attacks between arguments in A and we obtain $cf(F) = cf(\mathcal{R}(F))$. By the construction of $\mathcal{R}(F)$, for each non-empty $E \subseteq A$ we have $E_R^+ \cup \{t\} = E_{R^*}^+$ thus, $stg(F) = stg(\mathcal{R}(F))$. It is easy to see that $\emptyset \in stg(F)$ iff $cf(F) = \{\emptyset\}$ iff $\emptyset \in stg(\mathcal{R}(F))$.

Finally consider $Exists_{stage2}^{\neg\emptyset} \in \mathrm{P}$. Recall, for every AF F it holds that each $stage2$ extension of F is a naive extension of F. Thus, in case we have an F which possesses only the empty set as its $stage2$ extension, we know, the empty set is also the only naive extension of F. However, this is only the case if all arguments of F are self-attacking. Thus, to decide whether there exists a non-empty $cf2$ extension of an AF $F = (A, R)$, it is sufficient to check if there exists any argument $a \in A$ such that $(a,a) \notin R$. This can be done in polynomial time. ∎

We summarize the complexity results for naive-based semantics in Table 3. The results for naive semantics are due to [18], the ones for stable semantics are from [20] and the results for stage semantics have been shown in [37]. Regarding $cf2$, the complexity of $Cred_{cf2}, Skept_{cf2}$ and Ver_{cf2} is the same as for stable semantics, only non-emptiness is in P for $cf2$ where it is NP-complete for stable semantics. Considering the plethora of argumentation semantics, beside $stage2$, only for stage and semi-stable semantics the complexity of both skeptical and credulous reasoning is located on the second level of the polynomial hierarchy. Remember, for preferred semantics only skeptical acceptance is located on the second level of the polynomial hierarchy while credulous acceptance is NP-complete [25]. This indicates that $stage2$ is among the computationally hardest semantics but in the same breath also among the most expressive ones.

As mentioned before, the complexity results discussed so far are worst-case scenarios, for specific classes of problem instances one can achieve better results. In the next section

[5] Such a reduction $\mathcal{R}$ from stage to $stage2$, is called an exact translation for stage $\Rightarrow$ $stage2$ in [38]. In terms of [38] we show that stage semantics can be exactly translated to $stage2$ semantics.

	Ver_σ	$Cred_\sigma$	$Skept_\sigma$	$Exists_\sigma^{\neg\emptyset}$
naive	in P	in P	in P	in P
stable	in P	NP-c	coNP-c	NP-c
cf2	in P	NP-c	coNP-c	in P
stg	coNP-c	Σ_2^{P}-c	Π_2^{P}-c	in P
stage2	coNP-c	Σ_2^{P}-c	Π_2^{P}-c	in P

TABLE 3. Computational Complexity of naive-based semantics.

we investigate instances where possibly better results for *cf2* and *stage2* semantics can be obtained.

5.2 Tractable Fragments for *cf2* and *stage2*

As both *cf2* and *stage2* semantics are in general computationally intractable, i.e. the former is on the NP-layer while the latter is even on the second level of the polynomial hierarchy, naturally the issue of identifying tractable instances arises. Such studies of special instances of AFs where efficient algorithms can solve the reasoning problems on polynomial time have been done for several admissibility-based semantics [18, 23, 25]. In the following we study tractable fragments, i.e. classes of problem instances that can be decided in (deterministic) polynomial time, proposed for admissibility-based semantics and whether tractability also applies to *cf2* and *stage2* semantics.

First, we identify a relation between credulous and skeptical acceptance. By the following result, whenever credulous acceptance is tractable we immediately get tractability for skeptical acceptance.

PROPOSITION 5.3
Given an AF $F = (A, R)$ and $a \in A$ such that $(a, a) \notin R$. Then, a is skeptically accepted with *cf2* (resp. *stage2*) iff no $\{b \mid (b, a) \in R \text{ or } (a, b) \in R\}$ is credulously accepted with *cf2* (resp. *stage2*).

PROOF. For the proof we abstract from the concrete semantics *cf2*, *stage2* and consider an arbitrary semantics σ with $\sigma(F) \subseteq naive(F)$.
$\Rightarrow$: Consider $E \in \sigma(F)$ with $a \in E$. As $E \in cf(F)$, clearly $\{b \mid (b, a) \in R \text{ or } (a, b) \in R\} \cap E = \emptyset$. As, by assumption, for each $E \in \sigma(F)$ we have $a \in E$ no $\{b \mid (b, a) \in R \text{ or } (a, b) \in R\}$ is credulously accepted.
$\Leftarrow$: Consider $E \in \sigma(F)$ with $\{b \mid (b, a) \in R \text{ or } (a, b) \in R\} \cap E = \emptyset$. As $E \in naive(F)$ and $(a, a) \notin R$ we have $a \in E$. By assumption each $E \in \sigma(F)$ satisfies $\{b \mid (b, a) \in R \text{ or } (a, b) \in R\} \cap E = \emptyset$ and thus a is skeptically accepted. ∎

In the following we consider different graph classes which have been proposed as tractable fragments for abstract argumentation in the literature and study the complexity of *stage2* and *cf2* semantics on these graph classes.

5.2.1 Acyclic Argumentation Frameworks

One tractable fragment for argumentation is the class of acyclic AFs. Tractability is due to the fact that on acyclic AFs most semantics coincide with the grounded semantics [22]. This result extends to $cf2$ and $stage2$.

THEOREM 5.4
For acyclic AFs and $\sigma \in \{cf2, stage2\}$ the problems $Cred_\sigma$ and $Skept_\sigma$ are in P.

PROOF. We first show that, on acyclic AFs, grounded, $cf2$ and $stage2$ semantics coincide. Having a look at the SCC-recursive schema applied to acyclic AFs, then the base semantics is only applied to AFs consisting of a single argument and no attack. Thus semantics coincide if they coincide on these AFs. We have $grd(\{a\}, \emptyset) = naive(\{a\}, \emptyset) = stg(\{a\}, \emptyset) = \{\{a\}\}$ and thus the assertion follows. Now the complexity results are immediate by the fact that these problems are in P for grounded semantics. ∎

5.2.2 Even-Cycle Free Argumentation Frameworks

By a result in [24], reasoning with admissible-based semantics in AFs without even-length cycles is tractable. Unsurprisingly this result does not extend to $cf2$ and $stage2$ semantics. As odd and even length cycles are treated in the same manner we can simple replace the even-cycles by odd ones.

THEOREM 5.5
For AFs without even-length cycles:

- $Cred_{cf2}$ is NP-complete,
- $Skept_{cf2}$ is coNP-complete,
- $Cred_{stage2}$ is NP-hard, and
- $Skept_{stage2}$ is coNP-hard.

PROOF. The membership part for $cf2$ follows immediately from the complexity results for arbitrary AFs. For the hardness part we reduce the NP-hard SAT (resp. coNP-hard UNSAT) problem to $Cred$ (resp. $Skept$).

Given a 3-CNF formula $\varphi = \bigwedge_{j=1}^{m} C_j$ over atoms Z with $C_j = l_{j1} \vee l_{j2} \vee l_{j3}$ $(1 \leq j \leq m)$, the corresponding AF $F_\varphi = (A_\varphi, R_\varphi)$ is built as follows:

$$\begin{aligned} A_\varphi &= Z \cup \bar{Z} \cup \hat{Z} \cup \{C_1, \ldots, C_m\} \cup \{\varphi, \neg\varphi\} \\ R_\varphi &= \{(z, \bar{z}), (\bar{z}, \hat{z}), (\hat{z}, z) \mid z \in Z\} \cup \{(C_j, \varphi) \mid 1 \leq j \leq m\} \cup \{(\varphi, \neg\varphi)\} \cup \\ &\quad \{(z, C_j) \mid j \in \{1, \ldots, m\}, z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \cup \\ &\quad \{(\bar{z}, C_j) \mid j \in \{1, \ldots, m\}, \neg z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \end{aligned}$$

Figure 24 illustrates the AF F_φ of the formula $\varphi = (z_1 \vee z_2 \vee z_3) \wedge (\neg z_2 \vee \neg z_3 \vee \neg z_4) \wedge (\neg z_1 \vee z_2 \vee z_4)$.

An SCC of F_φ either consists of a single argument or is a cycle of length three which is not attacked by another SCC. As stage and naive semantics coincide on both we have $cf2(F_\varphi) = stage2(F_\varphi)$. Thus, in the remainder of the proof we only consider $cf2$ semantics. We now claim

(1) φ is satisfiable iff

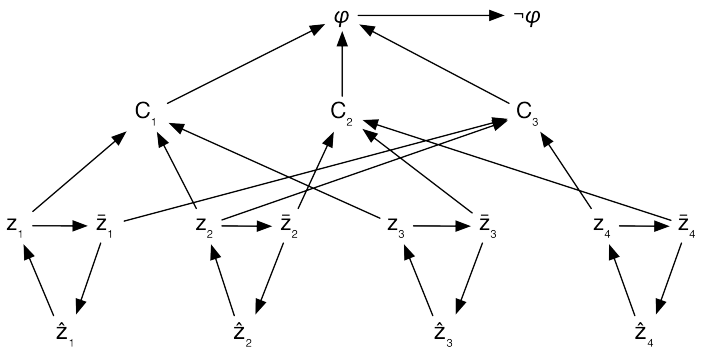


FIG. 24. AF F_φ for the 3-CNF φ.

(2) φ is credulously accepted in F_φ iff
(3) $\neg\varphi$ is not skeptically accepted in F_φ.

(1) $\Rightarrow$ (2): φ is satisfiable and thus it has a model $M \subseteq Z$. Consider the set $E = M \cup \{\bar{z} \mid z \in Z \setminus M\} \cup \{\varphi\}$. We next show, E is a *cf2* extension of F_φ. It is easy to check that $E \in naive(F_\varphi)$. So we consider $\Delta_{F_\varphi,E}$. As M is a model of φ each C_j is either attacked by a $z_i \in E$ or $\bar{z}_i \in E$, and as there are no attacks from C_j to $Z \cup \bar{Z}$ we obtain $C_j \in \Delta_{F_\varphi,E}$ for $1 \leq i \leq m$. Similarly, $\neg\varphi$ is attacked by φ, and as $\neg\varphi$ has no outgoing attacks also $\neg\varphi \in \Delta_{F_\varphi,E}$.

Now consider $Z \cup \bar{Z} \cup \hat{Z}$. Those arguments are not attacked from outside their SCCs, hence none of the arguments is contained in $\Delta_{F_\varphi,E}$. Now consider

$$F' = [[F_\varphi - \Delta_{F_\varphi,E}]] = (Z \cup \bar{Z} \cup \hat{Z} \cup \{\varphi\}, \{(z,\bar{z}), (\bar{z},\hat{z}), (\hat{z},z) \mid z \in Z\}).$$

It is easy to see that $E \in naive(F')$ and we finally obtain, $E \in cf2(F_\varphi)$. Hence, φ is credulously accepted.

(1) $\Leftarrow$ (2): Let $E \in cf2(F_\varphi)$ such that $\varphi \in E$. As E is conflict-free and $\varphi \in E$ we have $C_j \notin E$ for $1 \leq i \leq m$. Moreover $C_j \in \Delta_{F_\varphi,E}$. Assume the contrary, then there exists a $C_j \in [[F_\varphi - \Delta_{F_\varphi,E}]]$, and as C_j is not strongly connected to any argument, it is an isolated argument in the separation and thus in any naive set of $[[F_\varphi - \Delta_{F_\varphi,E}]]$, a contradiction. Now as $C_j \in \Delta_{F_\varphi,E}$, for each C_j there exists $l \in Z \cup \bar{Z}$ and $l \in E$ such that l attacks C_j (which is equivalent to $l \in C_j$). Notice, as E is conflict-free it can not happen that $\{z, \bar{z}\} \subseteq E$. Finally, we obtain $M = E \cap Z$ is a model of φ.

(2) $\Leftrightarrow$ (3): This is by the fact that in F_φ the argument $\neg\varphi$ is only connected to φ and thus each naive (resp. *cf2*) extension of F_φ either contains φ or $\neg\varphi$. ∎

While even-cycle free AFs are tractable for admissible-based semantics, in particular for stable semantics, they are still hard for *cf2*, *stage2* and also for stage semantics [31, 32].

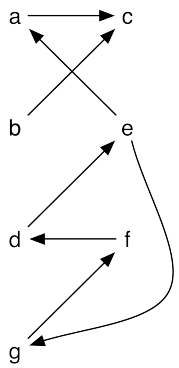


FIG. 25. The bipartite AF F from Example 5.6.

5.2.3 Bipartite Argumentation Frameworks

Bipartite AFs are a special class of frameworks where there exists a partition of the set of arguments A into two sets A_1 and A_2 such that attacks only exist between A_1 and A_2 but not within the sets.

EXAMPLE 5.6
Consider the AF $F = (A, R)$ as illustrated in Figure 25. We can partition A in $A_1 = \{a, b, d, g\}$ and $A_2 = \{c, e, f\}$, and it is easy to see that there are only attacks between those two sets. Thus, F is a bipartite argumentation framework. $\diamond$

Bipartite AFs have been shown to be tractable for admissible based semantics [23]. In the following we show that they are also tractable for $cf2$ and $stage2$ semantics.

THEOREM 5.7
For bipartite AFs the problems $Cred_{cf2}$, $Skept_{cf2}$ are in P.

PROOF. Given a bipartite AF $F = (A_1, A_2, R)$ with $A = A_1 \cup A_2$. In the following we use the notation $S \rightarrowtail a$ if a set S attacks an argument a. We consider the following procedure. Start with $E_1 = A_1$ and $E_2 = \emptyset$, iterate (until E_1, E_2 reach a fixed-point)

(1) $E_2 := E_2 \cup \{b \in A_2 \mid E_1 \not\rightarrowtail b\}$ and
(2) $E_1 := E_1 \setminus \{a \in E_1 \mid E_2 \rightarrowtail a\}$.

By results in [23] the above algorithm works in polynomial time and computes the stable extension $S = E_1 \cup E_2$ of F, with E_1 being the set of credulously accepted arguments of F from A_1 (w.r.t. stable semantics). We next show that this algorithm also applies to $cf2$. Due to [63], in coherent systems an argument is skeptically accepted iff none of its attackers is credulously accepted. Bipartite AFs are indeed coherent, this property explains intuitively the functioning of our procedure. To this end let C_1 be the set of credulously accepted arguments of F from A_1 and S_2 the set of skeptically accepted arguments of F from A_2 (w.r.t $cf2$ semantics). We claim that after each iteration step it holds that

(i) $E_1 \supseteq C_1$,
(ii) $E_2 \subseteq S_2$ and

(iii) $A_1 \setminus E_1 \subseteq \Delta_{F,S_2}$.

As an induction base observe that $E_1 = A_1$ and $E_2 = \emptyset$ trivially satisfies (i)-(iii). Now for the induction step assume (i)-(iii) holds before applying the iteration step, we have to show that it also holds afterwards.

First consider (ii): E_2 is only changed if there is a $b \in A_2$ and $E_1 \not\mapsto b$. But by (iii) this means that for all $E \in \mathit{cf2}(F)$ all attackers of b are contained in $\Delta_{F,E}$. Hence, for each $E \in \mathit{cf2}(F)$, the argument b is isolated in the AF $[[F - \Delta_{F,E}]]$ and thus clearly $b \in E$. Hence, $b \in S_2$ and (ii) is satisfied.

Now consider (i): By (2) an argument a is only removed from E_1 if it is attacked by a skeptically accepted argument. But then a can not be credulously accepted, i.e. $a \notin C_1$, and thus still $E_1 \supseteq C_1$.

Finally consider (iii): If an argument a is removed from E_1 it is attacked by an argument b such that for $E \in \mathit{cf2}(F)$ all attackers of b are contained in $\Delta_{F,E}$. Then clearly $a \not\mapsto_F^{A \setminus \Delta_{F,E}} b$ and thus $a \in \Delta_{F,E}$. Now using that $S = E_1 \cup E_2$ is a stable extension, the fixed-point of the above algorithm is also a *cf2* extension. Thus, $E_1 = C_1$ and $E_2 = S_2$. By symmetry we finally obtain that in bipartite AFs, the credulously (resp. skeptically) accepted arguments w.r.t. *cf2* coincide with the credulously (resp. skeptically) accepted arguments w.r.t. stable[6]. Hence, the P results for stable semantics in [23] carry over to *cf2* semantics. ∎

In the following we illustrate the procedure of the proof of Theorem 5.7 on the AF of Figure 25.

EXAMPLE 5.8
Let F be the bipartite AF of Example 5.6 with $A_1 = \{a, b, d, g\}$ and $A_2 = \{c, e, f\}$. We start the algorithm for computing credulous and skeptical accepted arguments as in the proof above. First, for $E_1 = A_1$ and $E_2 = \emptyset$ the sets remain unchanged. Thus, we obtain $S_1 = \{a, b, d, g\}$ as a stable extension of F which is also the set of credulously accepted arguments of F from A_1, and none of the arguments from A_2 is skeptically accepted in F. Due to symmetry we consider now $E_1 = A_2$ and $E_2 = \emptyset$. Then, we obtain

- $E_2 = \{b\}$ and
- $E_1 = A_2 \setminus \{c\} = \{e, f\}$.

The set $S_2 = \{b, e, f\}$ is a stable extension of F, the arguments e and f from A_2 are credulously accepted in F and $\{b\} \subset A_1$ is skeptically accepted in F (w.r.t. *cf2* and stable semantics). Finally, the arguments a, b, d, g, e and f are credulously accepted in F (w.r.t. *cf2* and stable semantics). ◇

Even though credulous and skeptical acceptance of *cf2* and stable semantics coincide on bipartite AFs, they propose different extensions. For instance consider the AF F from Example 2.15 (illustrated on page 9). F consists of a cycle of length 6 and is a bipartite, with $A_1 = \{a, c, e\}$ and $A_2 = \{b, d, f\}$. The set $\{a, d\}$ is a *cf2* extension of F which is not stable. Furthermore, no argument is skeptically accepted w.r.t. *cf2* and stable semantics but all arguments are credulously accepted in F. However, for *stage2* and stable semantics, also the extensions coincide.

[6] By $\mathit{stable}(F) \subseteq \mathit{stage2}(F) \subseteq \mathit{cf2}(F)$ and Proposition 5.3 this also extends to *stage2* semantics. However, this does not cover the complexity of the Ver_{stage2} problem.

	cf2	*stage2*	*stable*	*stg*
$Cred_\sigma{}^{acycl}$	in P	in P	P-c	P-c
$Skept_\sigma{}^{acycl}$	in P	in P	P-c	P-c
$Cred_\sigma{}^{even-free}$	NP-c	coNP-h	P-c	Σ_2^{P}-c
$Skept_\sigma{}^{even-free}$	coNP-c	coNP-h	P-c	Π_2^{P}-c
$Cred_\sigma{}^{bipart}$	in P	in P	P-c	P-c
$Skept_\sigma{}^{bipart}$	in P	in P	P-c	P-c
$Cred_\sigma{}^{sym}$	in P	in $\mathrm{P}/\Sigma_2^{\mathrm{P}}$-c*	in P	in $\mathrm{P}/\Sigma_2^{\mathrm{P}}$-c*
$Skept_\sigma{}^{sym}$	in P	in $\mathrm{P}/\Pi_2^{\mathrm{P}}$-c*	in P	in $\mathrm{P}/\Pi_2^{\mathrm{P}}$-c*

TABLE 4: Complexity results for special AFs * with self-attacking arguments). An entry $\mathcal{C}$-c denotes completeness for the class $\mathcal{C}$.

THEOREM 5.9
For bipartite AFs $Cred_{stage2}$, $Skept_{stage2}$, Ver_{stage2} are in P.

PROOF. Bipartite AFs are odd-cycle free and therefore coherent [22]. Hence stable and stage semantics coincide. By Proposition 3.10 we know that also $stable(F) = stage2(F)$. Then, the theorem follows from the results for stable semantics in [23]. ∎

5.2.4 Symmetric AFs

Finally we consider symmetric AFs, which where studied in [18]. In symmetric AFs all attacks go into both directions, hence all SCCs are isolated in the sense that there is no attack from one SCC to another (otherwise by symmetry, there would be an attack back and thus, those SCCs would merge to just one). Thus, in symmetric AFs *cf2* coincides with naive semantics while *stage2* coincides with stage semantics. We immediately obtain the complexity result for *cf2* and *stage2* by the corresponding results for naive and stage. In the first case this clearly leads to tractability. In the latter one we have to be more careful. If we follow [18] and assume that symmetric AFs are also irreflexive then, we have tractability by the fact that such AFs are coherent and stable semantics are tractable. However, without the assumption of irreflexivity, the tractability results for stable and stage semantics do not hold. Thus, they do not hold for *stage2* as well. In fact one can show that stage (and thus also *stage2*) semantics maintain their full complexity in symmetric AFs allowing self-attacks [27].

We summarize the results for the discussed tractable fragments in Table 4. For comparison we also included the results for stable and stage semantics from [27].

5.3 Backdoor Approach

A generalization of these fragments is the so called called backdoor approach [32], using the distance to a tractable fragment. This approach comes from parametrized complexity theory (see [42, 54]). For so called fixed-parameter tractability (fpt), one identifies problem parameters, for instance the above mentioned distance, such that computational costs heavily depend on the parameter but are only polynomial in the size of the instance. Now, if only considering

problem instances with bounded parameter, one obtains a polynomial time algorithm.

Next we formally define the distance of an AF to a graph class in terms of arguments that one has to delete such that the AF falls into the graph class.

DEFINITION 5.10
Let $\mathcal{G}$ be a graph class and $F = (A, R)$ an AF. We define $\text{dist}_{\mathcal{G}}(F)$ as the minimal number k such that there exists a set $S \subseteq A$ with $|S| = k$ and $(A \setminus S, R \cap (A \setminus S \times A \setminus S)) \in \mathcal{G}$. If there is no such set S we define $\text{dist}_{\mathcal{G}}(F) = \infty$.

In [32] it was shown that the backdoor approach does not help in the case of stage semantics, i.e. that even instances with a small distance are still NP or coNP hard, while it can be applied to admissibility based semantics. Here we show that the backdoor approach is also not applicable for $cf2$ and $stage2$ semantics.

THEOREM 5.11
$Cred_{cf2}$ is NP-hard, $Skept_{cf2}$ is coNP-hard even for AFs F with $\text{dist}_{acycl}(F) = 1$.

PROOF. We consider a variation of the standard reduction from Definition 5.1. Given a 3-CNF the AF $F'_\varphi = (A'_\varphi, R'_\varphi)$ is built as follows.

$$\begin{aligned} A'_\varphi &= Z \cup \bar{Z} \cup \{C_1, \ldots, C_m\} \cup \{\varphi, \neg\varphi, g\} \\ R'_\varphi &= \{(z, \bar{z}), (\bar{z}, g), (g, z) \mid z \in Z\} \cup \{(C_j, \varphi) \mid j \in \{1, \ldots, m\}\} \cup \{(\varphi, \neg\varphi), (g, g)\} \cup \\ &\quad \{(z, C_j) \mid j \in \{1, \ldots, m\}, z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \cup \\ &\quad \{(\bar{z}, C_j) \mid j \in \{1, \ldots, m\}, \neg z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \end{aligned}$$

For an illustration see Figure 26. The differences between F_φ and F'_φ being that F'_φ does not contain attacks $(\bar{z}, z)$ and the new argument g with its attacks. When deleting the argument g from the AF F'_φ the AF becomes acyclic and hence $\text{dist}_{acycl}(F'_\varphi) = 1$. Moreover one can show that $cf2(F_\varphi) = cf2(F'_\varphi)$ from which the claim follows.

We define $A = Z \cup \bar{Z} \cup \{g\}$, $F_{|A} = (A \setminus \{g\}, R_\varphi \cap (A \times A))$, and $F'_{|A} = (A, R'_\varphi \cap (A \times A))$. By the SCC-recursiveness of $cf2$ it suffices to show that $cf2(F_{|A}) = cf2(F'_{|A})$. As the g is self-attacking in $F'_{|A}$ and the remaining conflicts are the same as in $F_{|A}$ (although with different orientation) we obtain that $naive(F_{|A}) = naive(F'_{|A})$. In $F_{|A}$ there are only symmetric attacks and thus naive and stable semantics coincide. Using that each stable extension is a $cf2$ extension at each $cf2$-extension is a naive set we obtain $naive(F_{|A}) = cf2(F_{|A})$. Next consider $F'_{|A}$. As $F'_{|A}$ has just one strongly connected component (each argument is strongly connected to g) we obtain that $naive(F'_{|A}) = cf2(F'_{|A})$. Thus, $cf2(F_{|A}) = cf2(F'_{|A})$ and hence $cf2(F_\varphi) = cf2(F'_\varphi)$ as desired.

∎

THEOREM 5.12
$Cred_{cf2}$ is NP-hard, $Skept_{cf2}$ is coNP-hard even for AFs F with $\text{dist}_{bipart}(F) = 1$.

PROOF. Here we again consider the standard reduction from Definition 5.1 but apply it to a monotone 3-CNF (SAT is still NP-hard for such instances). The characteristic of a monotone 3-CNF is that each clause either consists solely of positive literals or solely of negative literals. We denote the set of the former by C and the set of the latter by $\bar{C}$.

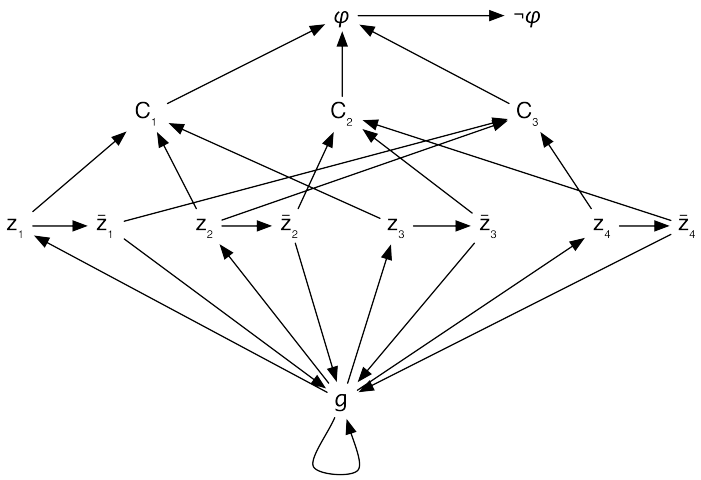


FIG. 26. AF F'_φ for the example 3-CNF φ.

Then we can partition A_φ in $A_1 = Z \cup \bar{C}$ $A_2 = \bar{Z} \cup C \cup \{\neg\varphi\}$ and $B = \{\varphi\}$. It is easy to verify that each of these sets is conflict free and thus when deleting B from F_φ we get an bipartite AF. Hence $\text{dist}_{bipart}(F_\varphi) = 1$ and the claim follows. ∎

THEOREM 5.13
$Cred_{cf2}$ is NP-hard, $Skept_{cf2}$ is coNP-hard even for AFs F with $\text{dist}_{sym}(F) = 1$.

PROOF. Once more we consider a variation of the standard reduction from Definition 5.1. Given a 3-CNF the AF $F^*_\varphi = (A^*_\varphi, R^*_\varphi)$ is built as follows.

$$\begin{aligned} A^*_\varphi &= Z \cup \bar{Z} \cup \{C_1, \ldots, C_m\} \cup \{\varphi, \neg\varphi\} \\ R^*_\varphi &= \{(z, \bar{z}), (\bar{z}, z) \mid z \in Z\} \cup \{(C_j, \varphi) \mid j \in \{1, \ldots, m\}\} \cup \{(\varphi, \neg\varphi)\} \cup \\ &\quad \{(z, C_j), (C_j, z) \mid j \in \{1, \ldots, m\}, z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \cup \\ &\quad \{(\bar{z}, C_j), (C_j, \bar{z}) \mid j \in \{1, \ldots, m\}, \neg z \in \{l_{j1}, l_{j2}, l_{j3}\}\} \end{aligned}$$

This reduction is illustrated in Figure 27. The differences between F_φ and F^*_φ being that in F^*_φ the attacks between literals and clauses are symmetric.

We show that φ is satisfiable iff φ is credulously accepted. The claim then follows.

$\Rightarrow$: Assume φ has a model M. We show that $E = M \cup \overline{X \setminus M} \cup \{\varphi\}$ is a stable extension and thus also a *cf2*-extension. First it is easy to check that E is conflict-free. For each $x \in X$ either $x \in E$ or $\bar{x} \in E$ and thus $x \in E^+$ for each $x \in X \cup \bar{X}$. Further as M is a model each C_i is attacked by an $x \in E$ or an $\bar{x} \in E$. Finally $\neg\varphi$ is attacked by φ and thus $E^+ = A_\varphi$. Hence E is a stable extension.

$\Leftarrow$: Now consider a *cf2* extension E with $\varphi \in E$. We define $A = \{C_1, \ldots, C_m\} \cup X \cup \bar{X}$, $E' = E \cap A$ and $F^*_{|A} = (A, R_\varphi \cap (A \times A))$. As mentioned earlier *cf2* satisfies directionality and thus we have that $E' \in cf2(F^*_{|A})$ and further as $F^*_{|A}$ is symmetric $E' \in naive(F^*_{|A})$. Using that $\varphi \in E$ we obtain that no C_i can be contained in E'. But as E' is a naive set we

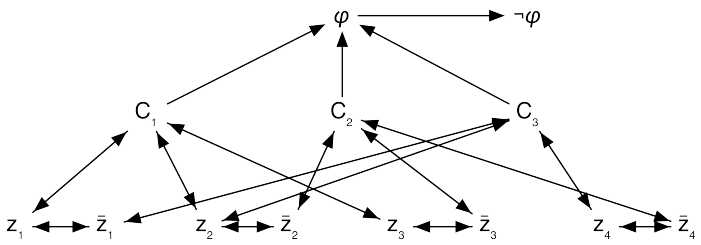


FIG. 27. AF F^*_φ for the example 3-CNF φ.

have that for each C_i there is either an $x \in E'$ or an $\bar{x} \in E'$ that attacks C_i, i.e. $x \in C_i$, resp. $\bar{x} \in C_i$. Thus $E \cap X$ is a model of φ. ∎

THEOREM 5.14
$Cred_{stage2}$ is NP-hard, $Skept_{stage2}$ is coNP-hard even for

- AFs F with $\text{dist}_{acycl}(F) \leq 2$,
- AFs F with $\text{dist}_{bipart}(F) \leq 2$, and
- AFs F with $\text{dist}_{sym}(F) \leq 3$.

PROOF. In [32] hardness proofs for the case of stage semantics are given for AFs F with $\text{dist}_{acycl}(F) = 1$, AFs F with $\text{dist}_{bipart}(F) \leq 1$, and AFs F with $\text{dist}_{sym}(F) \leq 2$. The reduction presented in the proof of Theorem 5.2 just introduces one argument and ensures that stage semantics coincides with $stage2$ semantics. Indeed this reduction just increases the distance to a tractable fragment by one. By combining this with the above mentioned results from [32] we obtain the claim. ∎

5.4 Summary and Further Considerations

To sum up, we completed the complexity analysis for $cf2$ and $stage2$ semantics for the standard reasoning problems verification, credulous and skeptical acceptance. It turned out that both semantics are intractable, where $stage2$ is even on the second level of the polynomial hierarchy.

Furthermore, we considered special instances of AFs and showed that acyclic, bipartite and symmetric self-attack free frameworks are tractable for both $cf2$ and $stage2$ semantics. Whereas, if self-attacking arguments are contained in a symmetric frameworks, then we do not have tractability for $stage2$. Even-cycle free AFs are not tractable for $cf2$ and $stage2$ semantics, which reflects the special behavior of these semantics on odd-length cycles.

Beside the backdoor approach there are several other ways to parametrize argumentation have been studies in the literature. First investigations for fixed-parameter tractability regarding abstract argumentation were undertaken for the graph parameters tree-width [23, 33] and clique-width [35]. The work in [36] shows that also reasoning with $cf2$ semantics is fpt w.r.t. tree-width and clique-width. Moreover, using the building blocks provided there, one can easily construct a monadic second order logic encoding for $stage2$ semantics, and by the

results presented in [36] this implies fixed-parameter tractability w.r.t. tree-width and clique-width. Moreover, [27, 33] shows that several parameters tailored for directed graphs are not applicable to abstract argumentation with the common semantics. These results immediately turn over to $cf2$ and $stage2$ semantics.

6 Implementation

In this section we concentrate on the more practical part of our investigation. In order to evaluate and compare abstract argumentation frameworks with respect to the numerous semantics it is indispensable to have efficient systems. Here we consider two approaches to such systems: First encoding argumentation semantics in answer-set programming and second using labelings as basis for dedicated algorithms.

Many argumentation semantics have been already implemented in ASP, see [60] for an overview. In this work we follow the ASPARTIX approach [40], where a single program is used to encode a particular semantics, while the instance of the framework is given as an input database. In particular we will present ASP encodings for $stage2$ semantics.

The challenging part in the design of the encodings for $stage2$ semantics is that the definition following the SCC-recursive schema involves a recursive computation of different sub-frameworks which is rather cumbersome to represent directly in ASP. This was the main reason why the alternative characterization for $cf2$ (resp. for $stage2$) semantics has been invented [45, 47]. With the alternative characterization we are able to directly (i) guess a set S of the given AF F and then (ii) check whether S is a stage extension of the instance $[[F - \Delta_{F,S}]]$. While the encodings for $cf2$ (as presented in [45]) are quite short and comprehensible this is not the case for the standard encodings for $stage2$ semantics. This semantics is located on the second level of the polynomial hierarchy and is based on stage semantics which requires a test for subset-maximality. To perform this test we need to apply a certain *saturation technique* [41] which is hardly accessible for non-experts in ASP.

However, recent advances in ASP solvers, in particular, the `metasp` optimization front-end for the ASP-system `gringo/claspD` allows for much simpler encodings for such tests. More precisely, `metasp` allows to use the traditional $\#minimize$ statement (which in its standard variant minimizes w.r.t. cardinality or weights, but not w.r.t. subset inclusion) also for selection among answer sets which are minimal w.r.t. subset inclusion in certain predicates. Details about `metasp` can be found in [49]. We will use this optimization to simplify the encodings for $stage2$ semantics.

Besides the ASP approach we will consider the *labeling-based approach* as a direct implementation method. Lately algorithms based on labelings attracted specific attention [13, 14, 53, 55, 62]. In contrast to the traditional extension-based approach, so called labelings (see e.g. [2]) distinguish different kinds of unaccepted arguments, e.g. those which are rejected by the extension and those which are neither rejected nor accepted. Such distinctions are interesting from a logic perspective [16] but also have proven to be highly useful for algorithmic issues. Hence, we follow this approach and present two algorithms which compute all valid labelings / extensions for $cf2$ and $stage2$ semantics.

6.1 Answer-Set Programming Encodings

We consider ASP as a reduction-based approach for the implementation of AFs. First we introduce the necessary background on ASP, then we formalize how argumentation frame-

works are represented in ASP and we give the encodings for $stage2$ semantics. In particular we distinguish between the standard saturation encodings and the optimized `metasp` encodings for $stage2$ semantics. All our encodings are fixed where the instance of an AF is given as input, and they are incorporated in the system ASPARTIX [7](see [39, 40] for more details). The encodings from the system ASPARTIX are written in the general datalog syntax. It may be the case that one needs to adapt the encodings for some ASP solvers. The `metasp` encodings can only be performed with `gringo/claspD`. Furthermore we point out that we give an informal description of the ASP encodings. For a more formal investigation of the system ASPARTIX we refer to [40].

6.1.1 Background Answer-Set Programming

We give a brief overview of the syntax and semantics of disjunctive logic programs under the answer-sets semantics [50]; for further background, see [52].

We fix a countable set $\mathcal{U}$ of *(domain) elements*, also called *constants*; and suppose a total order $<$ over the domain elements. An *atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity $n \geq 0$ and each t_i is either a variable or an element from $\mathcal{U}$. An atom is *ground* if it is free of variables. $B_\mathcal{U}$ denotes the set of all ground atoms over $\mathcal{U}$.

A *(disjunctive) rule* r with $n \geq 0, m \geq k \geq 0, n + m > 0$ is of the form

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_k, \; not\, b_{k+1}, \ldots, \; not\, b_m$$

where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms, and "not" stands for *default negation*. An atom a is a positive literal, while $not\, a$ is a default negated literal. The *head* of r is the set $H(r) = \{a_1, \ldots, a_n\}$ and the *body* of r is $B(r) = B^+(r) \cup B^-(r)$ with $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. A rule r is *normal* if $n \leq 1$ and a *constraint* if $n = 0$. A rule r is *safe* if each variable in r occurs in $B^+(r)$. A rule r is *ground* if no variable occurs in r. A *fact* is a ground rule without disjunction and with an empty body. An *(input) database* is a set of facts. A program is a finite set of disjunctive rules. For a program π and an input database D, we often write $\pi(D)$ instead of $D \cup \pi$. If each rule in a program is normal (resp. ground), we call the program normal (resp. ground).

Besides disjunctive and normal program, we consider here the class of optimization programs, i.e. normal programs which additionally contain $\#minimize$ statements

$$\#minimize[l_1 = w_1@J_1, \ldots, l_k = w_k@J_k] \tag{6.1}$$

where l_i is a literal, w_i an integer weight and J_i an integer priority level.

For any program π, let U_π be the set of all constants appearing in π. $Gr(\pi)$ is the set of rules $r\tau$ obtained by applying, to each rule $r \in \pi$, all possible substitutions τ from the variables in r to elements of U_π. An *interpretation* $I \subseteq B_\mathcal{U}$ *satisfies* a ground rule r iff $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. I satisfies a ground program π, if each $r \in \pi$ is satisfied by I. A non-ground rule r (resp., a program π) is satisfied by an interpretation I iff I satisfies all groundings of r (resp., $Gr(\pi)$). $I \subseteq B_\mathcal{U}$ is an *answer set* of π iff it is a subset-minimal set satisfying the *Gelfond-Lifschitz reduct*

$$\pi^I = \{H(r) \leftarrow B^+(r) \mid I \cap B^-(r) = \emptyset, r \in Gr(\pi)\}.$$

For a program π, we denote the set of its answer sets by $\mathcal{AS}(\pi)$. For semantics of optimization programs, we interpret the $\#minimize$ statement w.r.t. subset-inclusion: For any sets X

[7] The encodings are available at `http://www.dbai.tuwien.ac.at/research/project/argumentation/systempage/`.

e	normal programs	disjunctive program	optimization programs
$\models_c$	NP	Σ_2^{P}	Σ_2^{P}
$\models_s$	coNP	Π_2^{P}	Π_2^{P}

TABLE 5. Data Complexity for logic programs (all results are completeness results).

and Y of atoms, we have $Y \subseteq_J^w X$, if for any weighted literal $l = w@J$ occurring in (6.1), $Y \models l$ implies $X \models l$. Then, M is a collection of relations of the form $\subseteq_J^w$ for priority levels J and weights w. A standard answer set (i.e. not taking the minimize statements into account) Y of π *dominates* a standard answer set X of π w.r.t. M if there are a priority level J and a weight w such that $X \subseteq_J^w Y$ does not hold for $\subseteq_J^w \in M$, while $Y \subseteq_{J'}^{w'} X$ holds for all $\subseteq_{J'}^{w'} \in M$ where $J' \geq J$. Finally a standard answer set X is an answer set of an optimization program π w.r.t. M if there is no standard answer set Y of π that dominates X w.r.t. M.

Credulous and skeptical reasoning in terms of programs is defined as follows. Given a program π and a set of ground atoms A. Then, we write $\pi \models_c A$ (credulous reasoning), if A is contained in some answer set of π; we write $\pi \models_s A$ (skeptical reasoning), if A is contained in each answer set of π.

We briefly recall some complexity results for disjunctive logic programs. In fact, since we will deal with fixed programs we focus on results for data complexity. Depending on the concrete definition of $\models$, we give the complexity results in Table 6.1.1 (cf. [19] and the references therein). We note here, that even normal programs together with the optimization technique have a worst case complexity of Σ_2^{P} (resp. Π_2^{P}). Inspecting Table 2 one can see which kind of encoding is appropriate for an argumentation semantics.

6.1.2 Representing AFs in ASP

Here we first show how to represent AFs in ASP, and we give two programs which we need later on in this section. The first one π_{cf} opens the search space for our solutions via two guessing rules and eliminates all guesses which are not conflict-free. The second program $\pi_<$ defines an order over the domain elements.

All our programs are fixed which means that the only translation required, is to give an AF F as input database $\hat{F}$ to the program π_σ for a semantics σ. In fact, for an AF $F = (A, R)$, we define $\hat{F}$ as

$$\hat{F} = \{\ \mathrm{arg}(a) \mid a \in A\} \cup \{\mathrm{att}(a,b) \mid (a,b) \in R\ \}.$$

In what follows, we use unary predicates $\mathrm{in}/1$ and $\mathrm{out}/1$ to perform a guess for a set $S \subseteq A$, where $\mathrm{in}(a)$ represents that $a \in S$ (resp. $\mathrm{out}(a)$ for $a \notin S$). The following notion of correspondence is relevant for our purposes.

DEFINITION 6.1
Let $\mathcal{S} \subseteq 2^{\mathcal{U}}$ be a collection of sets of domain elements and let $\mathcal{I} \subseteq 2^{B_\mathcal{U}}$ be a collection of sets of ground atoms. We say that $\mathcal{S}$ and $\mathcal{I}$ correspond to each other, in symbols $\mathcal{S} \cong \mathcal{I}$, iff

(i) for each $S \in \mathcal{S}$, there exists an $I \in \mathcal{I}$, such that $\{a \mid \mathrm{in}(a) \in I\} = S$;
(ii) for each $I \in \mathcal{I}$, it holds that $\{a \mid \mathrm{in}(a) \in I\} \in \mathcal{S}$; and
(iii) $|\mathcal{S}| = |\mathcal{I}|$.

Let $F = (A, R)$ be an AF. The following program fragment guesses, when augmented by $\hat{F}$, any subset $S \subseteq A$ and then checks whether the guess is conflict-free in F:

$$\begin{aligned}\pi_{cf} = \{\ & \mathrm{in}(X) \leftarrow \mathit{not}\ \mathrm{out}(X), \mathrm{arg}(X);\\ & \mathrm{out}(X) \leftarrow \mathit{not}\ \mathrm{in}(X), \mathrm{arg}(X);\\ & \leftarrow \mathrm{in}(X), \mathrm{in}(Y), \mathrm{att}(X,Y)\ \}.\end{aligned}$$

PROPOSITION 6.2 ([40])
For any AF F, $cf(F) \cong \mathcal{AS}(\pi_{cf}(\hat{F}))$.

For ASP encodings, it is sometimes required or desired to avoid the use of negation. This might either be the case for the saturation technique or if a simple program can be solved without a Guess&Check approach. Then, encodings typically rely on a form of loops where all domain elements are visited and it is checked whether a desired property holds for all elements visited so far. We will use this technique for saturation-based encoding but also for the computation of the instance $[[F - \Delta_{F,S}]]$ for *stage2* semantics.

For this purpose, an order $<$ over the domain elements (usually provided by common ASP solvers) is used together with helper predicates defined in program $\pi_<$ below; in fact, predicates $\mathrm{inf}/1, \mathrm{succ}/2$ and $\mathrm{sup}/1$ denote infimum, successor and supremum of the order $<$.

$$\begin{aligned}\pi_< = \{\ & \mathrm{lt}(X,Y) \leftarrow \mathrm{arg}(X), \mathrm{arg}(Y), X < Y;\\ & \mathrm{nsucc}(X,Z) \leftarrow \mathrm{lt}(X,Y), \mathrm{lt}(Y,Z);\\ & \mathrm{succ}(X,Y) \leftarrow \mathrm{lt}(X,Y), \mathit{not}\ \mathrm{nsucc}(X,Y);\\ & \mathrm{ninf}(Y) \leftarrow \mathrm{lt}(X,Y);\\ & \mathrm{inf}(X) \leftarrow \mathrm{arg}(X), \mathit{not}\ \mathrm{ninf}(X);\\ & \mathrm{nsup}(X) \leftarrow \mathrm{lt}(X,Y);\\ & \mathrm{sup}(X) \leftarrow \mathrm{arg}(X), \mathit{not}\ \mathrm{nsup}(X)\ \}.\end{aligned}$$

6.2 ASP-Encodings for stage2 Semantics

Here we concentrate on implementing the *stage2* semantics in ASP. We provide a fixed program which, augmented with an input database representing a given AF F, has its answer sets in a one-to-one correspondence to the *stage2* extensions of F. In particular, the program computes *stage2* extensions along the lines of Proposition 3.2. The modularity of ASP allows us to split the program into several modules, where we also make use of the two program moduls π_{cf} and $\pi_<$ introduced above. Then, the program implements the following steps, given an AF $F = (A, R)$:

1. *Guess* the conflict-free sets $S \subseteq A$ of F.
2. For each S, compute the set $\Delta_{F,S}$.
3. For each S, derive the *instance* $[[F - \Delta_{F,S}]]$.
4. *Check* whether $S \in \mathit{stg}([[F - \Delta_{F,S}]])$.

Steps $1-3$ are the same as for the computation of *cf2* extensions introduced in [45]. For the sake of completeness we recall these steps here. For Step 4 we will use two different ways of computing stage extensions, namely the saturation encodings and the `metasp` encodings which have both been introduced for stage semantics in [30].

Step 1 is computed by π_{cf}, thus we go directly to Step 2. In the module π_{reach} we use the predicates $\inf(\cdot)$, $\mathrm{succ}(\cdot,\cdot)$ and $\sup(\cdot)$ from the module $\pi_<$ to iterate over the operator $\Delta_{F,S}(\cdot)$. Given $F = (A, R)$, by definition of $\Delta_{F,S}$ it is sufficient to compute at most $|A|$ such iterations to reach the fixed-point. Let us now present the module and then explain its behavior in more detail.

$$\begin{aligned}
\pi_{reach} = \{\ & \mathrm{arg_set}(N, X) \leftarrow \mathrm{arg}(X), \inf(N); && (6.2)\\
& \mathrm{reach}(N, X, Y) \leftarrow \mathrm{arg_set}(N, X), \mathrm{arg_set}(N, Y), \mathrm{att}(X, Y); && (6.3)\\
& \mathrm{reach}(N, X, Y) \leftarrow \mathrm{arg_set}(N, X), \mathrm{att}(X, Z), \mathrm{reach}(N, Z, Y); && (6.4)\\
& \mathrm{d}(N, X) \leftarrow \mathrm{arg_set}(N, Y), \mathrm{arg_set}(N, X), \mathrm{in}(Y), \mathrm{att}(Y, X),\\
& \qquad\qquad \mathit{not}\ \mathrm{reach}(N, X, Y); && (6.5)\\
& \mathrm{arg_set}(M, X) \leftarrow \mathrm{arg_set}(N, X), \mathit{not}\ \mathrm{d}(N, X), \mathrm{succ}(N, M)\ \}. && (6.6)
\end{aligned}$$

Rule (6.2) first copies all arguments into a set indexed by the infimum which initiates the computation. The remaining rules are applicable to arbitrary indices, whereby rule (6.6) copies (a subset of the) arguments from the currently computed set into the "next" set using the successor function $\mathrm{succ}(\cdot,\cdot)$. This guarantees a step-by-step computation of $\mathrm{arg\_set}(i, \cdot)$ by incrementing the index i. The functioning of rules (6.3)–(6.6) is as follows. Rules (6.3) and (6.4) compute a predicate $\mathrm{reach}(n, x, y)$ indicating that there is a path from argument x to argument y in the given framework *restricted* to the arguments of the current set n. In rule (6.5), $\mathrm{d}(n, x)$ is obtained for all arguments x which are component-defeated by S in this restricted framework. In other words, if n is the i-th argument in the order $<$, $\mathrm{d}(n, x)$ carries exactly those arguments x which are contained in $\Delta^i_{F,S}$. Finally, rule (6.6) copies arguments from the current set which are *not* component-defeated to the successor set.

Next, we derive the instance $[[F - \Delta_{F,S}]]$ with the module π_{inst}. As already outlined above, if the supremum m is reached in π_{reach}, we are guaranteed that the derived atoms $\mathrm{arg\_set}(m, x)$ characterize exactly those arguments x from the given AF F which are not contained in $\Delta_{F,S}$. It is thus now relatively easy to obtain the instance $[[F - \Delta_{F,S}]]$ which is done below via predicates $\mathrm{arg\_new}(\cdot)$ and $\mathrm{att\_new}(\cdot,\cdot)$.

$$\begin{aligned}
\pi_{inst} = \{\ & \mathrm{arg_new}(X) \leftarrow \mathrm{arg_set}(M, X), \sup(M);\\
& \mathrm{att_new}(X, Y) \leftarrow \mathrm{arg_new}(X), \mathrm{arg_new}(Y), \mathrm{att}(X, Y),\\
& \qquad\qquad \mathrm{reach}(M, Y, X), \sup(M)\ \}.
\end{aligned}$$

In the following we give the two different ways of how to encode the checking part. We start with the saturation encodings for *stage2*.

6.2.1 Saturation Encodings for *stage2* Semantics

The saturation technique as introduced by Eiter and Gottlob in [41] allows for encodings which solve associated problems located on the second level of the polynomial hierarchy. This technique was already used to encode the preferred and semi-stable semantics in [40] and the stage semantics in [30]. While with default negation, one is capable to formulate an exclusive guess, disjunction can be employed for the saturation technique, which allows for representing even more complex problems. The term "saturation" indicates that all atoms which are subject to a guess can also be jointly contained in an interpretation. To saturate a guess, it is however necessary that the checking part of a program interacts with the guessing part.

To implement Step 4, for any $F = (A, R)$ and $S \in cf(F)$, check whether $S \in stg(F')$ with $F' = [[F - \Delta_{F,S}]]$, we need to check whether no $T \in cf(F')$ with $S^+_{R'} \subset T^+_{R'}$ exists. Therefore we have to guess an arbitrary set T and saturate if either

(i) T is not conflict-free in F', or
(ii) $S^+_{R'} \not\subset T^+_{R'}$.

Let $F = (A, R)$, the following module computes for a guessed set $S \subseteq A$ the range $S^+_{R'}$ of S, in $F' = [[F - \Delta_{F,S}]]$.

$$\begin{aligned}\pi_{range} = \{\ & \mathrm{in\_range}(X) \leftarrow \mathrm{in}(X);\\ & \mathrm{in\_range}(X) \leftarrow \mathrm{in}(Y), \mathrm{att\_new}(Y, X);\\ & \mathrm{not\_in\_range}(X) \leftarrow \mathrm{arg\_new}(X), \mathit{not}\, \mathrm{in\_range}(X)\ \}.\end{aligned}$$

In the next module we make a second guess for the set T. Then, $\mathrm{in}/1$ holds the current guess for S and $\mathrm{inN}/1$ holds the current guess for T.

$$\begin{aligned}\pi_{satstage} = \{\ & \mathrm{inN}(X) \vee \mathrm{outN}(X) \leftarrow \mathrm{arg\_new}(X); && (6.7)\\ & \mathrm{fail} \leftarrow \mathrm{inN}(X), \mathrm{inN}(Y), \mathrm{att\_new}(X, Y); && (6.8)\\ & \mathrm{fail} \leftarrow \mathrm{eqplus}; && (6.9)\\ & \mathrm{fail} \leftarrow \mathrm{in\_range}(X), \mathrm{not\_in\_rangeN}(X); && (6.10)\\ & \mathrm{inN}(X) \leftarrow \mathrm{fail}, \mathrm{arg\_new}(X); && (6.11)\\ & \mathrm{outN}(X) \leftarrow \mathrm{fail}, \mathrm{arg\_new}(X); && (6.12)\\ & \leftarrow \mathit{not}\, \mathrm{fail}\ \}. && (6.13)\end{aligned}$$

More specifically:

- In rule (6.7) we use disjunction for the guess. This is essential for the saturation technique because it allows for an argument a to have both $\mathrm{inN}(a)$ and $\mathrm{outN}(a)$ in the same answer set which is not possible for the predicates $\mathrm{in}/1$ and $\mathrm{out}/1$ from module π_{cf}.
- Rule (6.8) checks requirement (i), so if the set T is not conflict-free in F' we derive fail.
- Rule (6.9) fires in case $S^+_{R'} = T^+_{R'}$ (indicated by predicate $\mathrm{eqplus}/0$ described below).
- Rule (6.10) fires if there exists an $a \in S^+_{R'}$ such that $a \notin T^+_{R'}$ (here we use predicate $\mathrm{in\_range}/1$ from above and predicate $\mathrm{not\_in\_rangeN}/1$ which we also present below). As is easily checked one of the last two conditions holds exactly if (ii) holds.
- Next, the rules (6.11) and (6.12) saturate if fail was derived. This means that we derive for each $a \in A$ both $\mathrm{inN}(a)$ and $\mathrm{outN}(a)$ and therefore blow up the answer sets.
- Finally, the constraint (6.13) rules out all guesses which do not contain fail.

To sum up, exactly those sets S survive where there is no T which is both conflict-free (in F') and has a bigger range than S (in F').

Next, in the module π_{rangeN} we compute the predicate $\mathrm{not\_in\_rangeN}/1$ via $\mathrm{undef\_upto}/2$. To compute the predicate $\mathrm{undef\_upto}(i, a)$ which states that the argument a is undefeated (in F') up to the i-th argument, we use new predicates for the order over the arguments in the instance F' as given below in the module $\pi_{<'}$. This is important as there might be less arguments in F' than in F, so that the infimum, supremum and successor may have changed. Then, if an argument a is undefeated up to the supremum (in F'), we derive

$\mathrm{not\_in\_rangeN}(a)$. Furthermore we compute the predicate $\mathrm{in\_rangeN}/1$ which gives us the range $T^+_{R'}$ for the arguments in the second guess.

$$\begin{aligned}\pi_{rangeN} = \{\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{infN}(N), \mathrm{outN}(X), \mathrm{outN}(N);\\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{infN}(N), \mathrm{outN}(X), \mathit{not}\ \mathrm{att\_new}(N,X);\\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{succN}(Z,N), \mathrm{undef\_upto}(Z,X), \mathrm{outN}(N);\\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{succN}(Z,N), \mathrm{undef\_upto}(Z,X), \mathit{not}\ \mathrm{att\_new}(N,X);\\ & \mathrm{not\_in\_rangeN}(X) \leftarrow \mathrm{supN}(M), \mathrm{outN}(X), \mathrm{undef\_upto}(M,X);\\ & \mathrm{in\_rangeN}(X) \leftarrow \mathrm{inN}(X);\\ & \mathrm{in\_rangeN}(X) \leftarrow \mathrm{outN}(X), \mathrm{inN}(Y), \mathrm{att\_new}(Y,X)\ \}.\end{aligned}$$

$\pi_{<'}$ is defined as the order over the arguments contained in the instance $[[F - \Delta_{F,S}]]$.

$$\begin{aligned}\pi_{<'} = \{\ & \mathrm{ltN}(X,Y) \leftarrow \mathrm{arg\_new}(X), \mathrm{arg\_new}(Y), X < Y;\\ & \mathrm{nsuccN}(X,Z) \leftarrow \mathrm{ltN}(X,Y), \mathrm{ltN}(Y,Z);\\ & \mathrm{succN}(X,Y) \leftarrow \mathrm{ltN}(X,Y), \mathit{not}\ \mathrm{nsuccN}(X,Y);\\ & \mathrm{ninfN}(Y) \leftarrow \mathrm{ltN}(X,Y);\\ & \mathrm{infN}(X) \leftarrow \mathrm{arg}(X), \mathit{not}\ \mathrm{ninfN}(X);\\ & \mathrm{nsupN}(X) \leftarrow \mathrm{ltN}(X,Y);\\ & \mathrm{supN}(X) \leftarrow \mathrm{arg\_new}(X), \mathit{not}\ \mathrm{nsupN}(X)\ \}.\end{aligned}$$

In the module π^+_{eq} we obtain eqplus, if the range from the first guess S and the second guess T is equal (in F'), i.e. if $S^+_{R'} = T^+_{R'}$. This is done via the predicate $\mathrm{eqplus\_upto}/1$.

$$\begin{aligned}\pi^+_{eq} = \{\ & \mathrm{eqplus\_upto}(X) \leftarrow \mathrm{infN}(X), \mathrm{in\_range}(X), \mathrm{in\_rangeN}(X);\\ & \mathrm{eqplus\_upto}(X) \leftarrow \mathrm{infN}(X), \mathrm{not\_in\_range}(X), \mathrm{not\_in\_rangeN}(X);\\ & \mathrm{eqplus\_upto}(X) \leftarrow \mathrm{succN}(Z,X), \mathrm{in\_range}(X), \mathrm{in\_rangeN}(X), \mathrm{eqplus\_upto}(Z);\\ & \mathrm{eqplus\_upto}(X) \leftarrow \mathrm{succN}(Y,X), \mathrm{not\_in\_range}(X), \mathrm{not\_in\_rangeN}(X),\\ & \qquad\qquad \mathrm{eqplus\_upto}(Y);\\ & \mathrm{eqplus} \leftarrow \mathrm{supN}(X), \mathrm{eqplus\_upto}(X)\ \}.\end{aligned}$$

Finally, we put everything together and obtain the encodings for *stage2* semantics:

$$\pi_{stage2} = \pi_{cf} \cup \pi_{<} \cup \pi_{reach} \cup \pi_{inst} \cup \pi_{range} \cup \pi_{<'} \cup \pi_{rangeN} \cup \pi^+_{eq} \cup \pi_{satstage}.$$

The following result gives the link between the *stage2* extensions of an AF F and the answer sets of the program π_{stage2} with the input $\hat{F}$[8].

PROPOSITION 6.3
For any AF F, $stage2(F) \cong \mathcal{AS}(\pi_{stage2}(\hat{F}))$.

As one can see saturation encodings are quite complicated and normally one needs an ASP expert to design them. As many interesting problems require some kind of meta-reasoning, Gebser and Schaub designed the `metasp` optimization front end for `gringo/claspD` [49], which also allows for ASP beginners to encode problems which are on the second level of the polynomial hierarchy. In the next subsection we will explain how one can simplify the encodings of *stage2* semantics using `metasp`.

[8] We omit the proof of Proposition 6.3 as it would require to introduce several further concepts which are not relevant for the content of this article. For the interested reader we refer to [40] which contains proofs of similar ASP encodings for other argumentation semantics.

6.2.2 `metasp` Encodings for *stage2* Semantics

In [30] the `metasp` approach has been used to simplify the encodings for preferred, semi-stable, stage and resolution-based grounded semantics. Here we picture this novel method by means of *stage2* semantics. In particular we present the simplified encodings for *stage2* semantics with the aid of the $\#minimize$ statement which are then evaluated with the subset-minimization semantics provided by `metasp`. For our encodings we do not need prioritization and weights, therefore these are omitted (i.e. set to default) in the minimization statements. The minimization technique is realized through meta programming techniques, which themselves are answer set programs. This works as follows.

- The ASP encodings to solve are given to the grounder `gringo` which reifies the programs, i.e. outputs ground programs consisting of facts, which represent the rules and facts of the original input encodings.
- The grounder is then again executed on this output together with the meta programs which encode the optimization.
- Finally, `claspD` computes the answer sets.

Note that here we use the version of `clasp` which supports disjunctive rules. Therefore for a program π and an AF F we have the following execution.

```
gringo --reify π(F̂) | \
    gringo - {meta.lp,metaO.lp,metaD.lp} \
    <(echo ``optimize(1,1,incl).'')  | claspD 0
```

Here, `meta.lp, metaO.lp` and `metaD.lp` are the encodings for the minimization statement [9]. The statement `optimize(incl,1,1)` indicates that we use subset inclusion for the optimization technique using priority and weight 1.

Now the program for *stage2* semantics uses the modules π_{cf}, $\pi_{<}$, π_{reach} and π_{inst} to compute for each guess the instance $[[F-\Delta_{F,S}]]$. For the check if the guessed set S is a stage extension of the instance, we first compute the predicate $\mathrm{not\_in\_rangeN}/1$ via $\mathrm{undef\_upto}/2$ in the slightly modified module $\pi_{rangeN'}$.

$$\begin{aligned}\pi_{rangeN'} = \{\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{inf}(N), \mathrm{out}(X), \mathrm{out}(N);\\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{inf}(N), \mathrm{out}(X), \mathit{not}\ \mathrm{att\_new}(N,X);\\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{succ}(Z,N), \mathrm{undef\_upto}(Z,X), \mathrm{out}(N);\\ & \mathrm{undef\_upto}(N,X) \leftarrow \mathrm{succ}(Z,N), \mathrm{undef\_upto}(Z,X), \mathit{not}\ \mathrm{att\_new}(N,X);\\ & \mathrm{not\_in\_rangeN}(X) \leftarrow \mathrm{sup}(M), \mathrm{out}(X), \mathrm{undef\_upto}(M,X)\ \}.\end{aligned}$$

Then, we check if S is a naive extension of the instance in the module π_{check_naive}.

$$\begin{aligned}\pi_{check\_naive} = \{\ & \mathrm{conflicting}(X) \leftarrow \mathrm{att\_new}(Y,X), \mathrm{out}(X), \mathrm{in}(Y);\\ & \mathrm{conflicting}(X) \leftarrow \mathrm{att\_new}(X,Y), \mathrm{out}(X), \mathrm{in}(Y);\\ & \mathrm{conflicting}(X) \leftarrow \mathrm{att\_new}(X,X);\\ & \leftarrow \mathit{not}\ \mathrm{conflicting}(X), \mathrm{not\_in\_rangeN}(X), \mathrm{arg\_new}(X)\ \}.\end{aligned}$$

[9] available at: `http://www.cs.uni-potsdam.de/wv/metasp/`

We put everything together including the minimize statement for $\mathrm{not\_in\_range}\mathrm{N}/1$.

$$\pi_{stage2\_metasp} = \pi_{cf} \cup \pi_{<} \cup \pi_{reach} \cup \pi_{inst} \cup \pi_{check\_naive} \cup \pi_{rangeN'} \cup \{\#minimize[\mathrm{not\_in\_range}]\}.$$

Finally we obtain the following result.

PROPOSITION 6.4
For any AF F, $stage2(F) \cong \mathcal{AS}(\pi_{stage2\_metasp}(\hat{F}))$.

Performance tests comparing the saturation encodings with the `metasp` encodings on different random instances showed that the use of this optimization front-end not only makes the encodings simpler but also faster. Especially in the case of stage semantics the runtime differences are in evidence. A detailed representation of the experimental evaluation can be found in [30].

6.3 Labeling-based Algorithms

Labeling-based algorithms build on top of alternative characterizations for argumentation semantics using certain labeling functions, assigning labels to arguments. While extension-based semantics just distinguish whether an argument is in the extensions or not, the labeling-based approach allows for a more fine-grained classification of the justification status of an argument, probably the most prominent variant being the 3-valued labelings due to Caminada and Gabbay [16]. These labelings label each argument either with *in*, *out* or *undec*, with the intended meaning that they are either accepted, rejected or one has not decided whether to accept or to reject the argument. There are characterizations in terms of labelings for nearly all prominent semantics, for an overview we refer to [2], where also a labeling for $cf2$ semantics is included. However, we present here a slightly different notion of $cf2$ labelings, which better suites our purposes, as well as labeling-based characterization of $stage2$ semantics. On top of these we provide labeling based algorithms for $cf2$ and $stage2$ semantics, which are complexity sensitive in the sense that they reflect some results from Section 5.

6.3.1 Labelings

Here we present labeling-based characterizations of argumentation semantics which we will exploit in the subsequent algorithms. As we use labelings solely for algorithmic issues we use a 4-valued labelings extending the 3-valued labelings of [16] by the label out_{Δ}.

DEFINITION 6.5
Let $F = (A, R)$ be an AF. A *labeling* is a total function $\mathcal{L} : A \to \{in, out, out_{\Delta}, undec\}$.

Then, a labeling can be denoted as a tuple $\mathcal{L} = (\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_{\Delta}}, \mathcal{L}_{undec})$, where $\mathcal{L}_l = \{a \in A \mid \mathcal{L}(a) = l\}$. Following [2] conflict-free and stage labelings are given as follows.

DEFINITION 6.6
Let $F = (A, R)$ be an AF. Then, $\mathcal{L}$ is a *conflict-free labeling* of F, i. e. $\mathcal{L} \in cf_{\mathcal{L}}(F)$, iff

- for all $a \in \mathcal{L}_{in}$ there is no $b \in \mathcal{L}_{in}$ such that $(a, b) \in R$,
- for all $a \in \mathcal{L}_{out}$ there exists a $b \in \mathcal{L}_{in}$ such that $(b, a) \in R$, and
- $\mathcal{L}_{out_{\Delta}} = \emptyset$.

Then, $\mathcal{L}$ is a *naive labeling* of F, i. e. $\mathcal{L} \in naive_{\mathcal{L}}(F)$, iff $\mathcal{L} \in cf_{\mathcal{L}}(F)$ and there is no $\mathcal{L}' \in cf_{\mathcal{L}}(F)$ with either $\mathcal{L}_{in} \subset \mathcal{L}'_{in}$ or $\mathcal{L}_{out} \subset \mathcal{L}'_{out}$. Finally, $\mathcal{L}$ is a *stage labeling* of F, i. e. $\mathcal{L} \in stg_{\mathcal{L}}(F)$, iff $\mathcal{L} \in cf_{\mathcal{L}}(F)$ and there is no $\mathcal{L}' \in cf_{\mathcal{L}}(F)$ with $\mathcal{L}'_{undec} \subset \mathcal{L}_{undec}$.

We are now prepared to define $cf2$ and $stage2$ labelings, where an argument is labeled out_Δ iff it is attacked by an argument labeled in which does not belong to the same SCC.

DEFINITION 6.7
Let $F = (A, R)$ be an AF. Then, $\mathcal{L}$ is a *cf2 labeling* of F, i.e. $\mathcal{L} \in cf2_{\mathcal{L}}(F)$, iff

- $\mathcal{L} \in naive_{\mathcal{L}}(F)$, in case $|SCCs(F) = 1|$.
- otherwise, $\forall C \in SCCs(F), \mathcal{L}|_{C \setminus D_F(\mathcal{L}_{in})} \in cf2_{\mathcal{L}}(F|_C - D_F(\mathcal{L}_{in}))$, and $D_F(\mathcal{L}_{in}) = \mathcal{L}_{out_\Delta}$.

It is easy to see that there is a one-to-one mapping between $cf2$ extensions and labelings, s.t. each extension S corresponds to a labeling $\mathcal{L}$ with $\mathcal{L}_{in} = S$ and $\mathcal{L}_{out_\Delta} = \Delta_{F,S}$.

From the alternative characterization for $cf2$-extensions we obtain the following characterization for $cf2$-labelings $\mathcal{L} \in cf2_{\mathcal{L}}(F)$, iff $(\mathcal{L}_{in}, \mathcal{L}_{out} \cup \mathcal{L}_{out_\Delta}, \emptyset, \mathcal{L}_{undec}) \in naive_{\mathcal{L}}(F)$, $(\mathcal{L}_{in}, \mathcal{L}_{out}, \emptyset, \mathcal{L}_{undec}) \in naive_{\mathcal{L}}([[F - \Delta_{F,\mathcal{L}_{in}}]])$, and $\Delta_{F,\mathcal{L}_{in}} = \mathcal{L}_{out_\Delta}$.

The labeling based definition of $stage2$ semantics is in a similar fashion.

DEFINITION 6.8
Let $F = (A, R)$ be an AF. Then, $\mathcal{L}$ is a *stage2 labeling* of F, i.e. $\mathcal{L} \in stage2_{\mathcal{L}}(F)$, iff

- $\mathcal{L} \in stg_{\mathcal{L}}(F)$, in case $|SCCs(F) = 1|$.
- otherwise, $\forall C \in SCCs(F), \mathcal{L}|_{C \setminus D_F(\mathcal{L}_{in})} \in stage2_{\mathcal{L}}(F|_C - D_F(\mathcal{L}_{in}))$, and $D_F(\mathcal{L}_{in}) = \mathcal{L}_{out_\Delta}$

Again there is a one-to-one mapping between $stage2$ extensions and labelings, and each extension S corresponds to a labeling $\mathcal{L}$ with $\mathcal{L}_{in} = S$ and $\mathcal{L}_{out_\Delta} = \Delta_{F,S}$.

Using the characterization from Proposition 3.2 we also obtain that $\mathcal{L} \in stage2_{\mathcal{L}}(F)$, iff $(\mathcal{L}_{in}, \mathcal{L}_{out} \cup \mathcal{L}_{out_\Delta}, \emptyset, \mathcal{L}_{undec}) \in naive_{\mathcal{L}}(F)$, $(\mathcal{L}_{in}, \mathcal{L}_{out}, \emptyset, \mathcal{L}_{undec}) \in stg_{\mathcal{L}}([[F - \Delta_{F,\mathcal{L}_{in}}]])$, and $\Delta_{F,\mathcal{L}_{in}} = \mathcal{L}_{out_\Delta}$.

Finally, we have $stage2_{\mathcal{L}}(F) \subseteq cf2_{\mathcal{L}}(F)$ for every AF F.

6.3.2 Labeling Algorithm for $cf2$

The basic idea of labeling-based algorithms is to iterate over the possible labeling functions and test them for being, in our case, $cf2$-labelings. Of courses, one does not want to consider all labelings, hence one tries to reduce the search space. In labeling-based algorithms this is done by using the fact as soon a label for one argument is fixed, this has immediate implications for the possible labels of the arguments in the neighborhood. The following proposition identifies two such rules to propagate already computed labels.

PROPOSITION 6.9
For AF $F = (A, R)$ and labeling $\mathcal{L} = (\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_\Delta}, \mathcal{L}_{undec}) \in cf2_{\mathcal{L}}(F)$. Let $a \in A$, then $att(a) = \{b \in A \mid (b, a) \in R\}$ denotes all attackers of a.

1. For every $a \in A$: if $att(a) \subseteq \mathcal{L}_{out_\Delta} \wedge (a, a) \notin R$ then $a \in \mathcal{L}_{in}$.
2. For every $a \in A$: if $\exists b \in \mathcal{L}_{in}, O \subseteq \mathcal{L}_{out_\Delta} : (b, a) \in R \wedge a \not\Rightarrow_F^{A \setminus O} b$ then $a \in \mathcal{L}_{out_\Delta}$.

PROOF. (1) By definition $a \in \mathcal{L}_{out_\Delta}$ implies $a \in \Delta_{F,\mathcal{L}_{in}}$. If all attackers of a are in $\Delta_{F,\mathcal{L}_{in}}$ we get that $\{a\}$ is an isolated argument in $[[F - \Delta_{F,S}]]$. Now, as $\mathcal{L} \in naive([[F - \Delta_{F,S}]])$ and $(a,a) \notin R$ we finally get $a \in \mathcal{L}_{in}$.

(2) Using $\exists b \in \mathcal{L}_{in}, O \subseteq \mathcal{L}_{out_\Delta} : (b,a) \in R \wedge a \not\Rightarrow_F^{A \setminus O} b$ and $O \subseteq \mathcal{L}_{out_\Delta} = \Delta_{F,\mathcal{L}_{in}}$, we obtain that $\exists b \in \mathcal{L}_{in} : (b,a) \in R \wedge a \not\Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} b$. As $\Delta_{F,\mathcal{L}_{in}}$ is a fixed-point we obtain that $a \in \Delta_{F,\mathcal{L}_{in}}$ and thus also $a \in \mathcal{L}_{out_\Delta}$. ∎

We exploit these rules in Algorithm 1 for computing *cf2* extensions.

Algorithm 1 $cf2_{\mathcal{L}}(F, \mathcal{L})$

Require: AF $F = (A, R)$, labeling $\mathcal{L} = (\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_\Delta}, \mathcal{L}_{undec})$;
Ensure: Return all *cf2* labelings of F.

1: $X = \{a \in \mathcal{L}_{undec} \mid att(a) \subseteq \mathcal{L}_{out_\Delta}\}$;
2: $Y = \{a \in \mathcal{L}_{undec} \mid \exists b \in \mathcal{L}_{in}, (b,a) \in R, a \not\Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} b\}$;
3: **while** $(X \cup Y) \neq \emptyset$ **do**
4: $\quad \mathcal{L}_{in} = \mathcal{L}_{in} \cup X, \mathcal{L}_{out_\Delta} = \mathcal{L}_{out_\Delta} \cup Y, \mathcal{L}_{undec} = \mathcal{L}_{undec} \setminus (X \cup Y)$;
5: $\quad$ update X and Y;
6: **end while**
7: $B = \{a \in \mathcal{L}_{undec} \mid \mathcal{L}_{in} \cup \{a\} \in cf(F)\}$;
8: **if** $B \neq \emptyset$ **then**
9: $\quad C = \{a \in B \mid \nexists b \in B : b \Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} a, a \not\Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} b\}$;
10: $\quad \mathcal{E} = \emptyset$;
11: $\quad$ **for all** $\mathcal{L}' \in naive_{\mathcal{L}}(F|_C)$ **do**
12: $\quad\quad \mathcal{L}_{in} = \mathcal{L}_{in} \cup \mathcal{L}'_{in}, \mathcal{L}_{out} = \mathcal{L}_{out} \cup \mathcal{L}'_{out}, \mathcal{L}_{undec} = \mathcal{L}_{undec} \setminus (\mathcal{L}'_{in} \cup \mathcal{L}'_{out})$;
13: $\quad\quad \mathcal{E} = \mathcal{E} \cup cf2_{\mathcal{L}}(F, \mathcal{L})$;
14: $\quad$ **end for**
15: $\quad$ **return** $\mathcal{E}$;
16: **else**
17: $\quad$ **return** $\{(\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_\Delta}, \mathcal{L}_{undec})\}$;
18: **end if**

To compute all *cf2* labelings of an AF F the function $stage2_{\mathcal{L}}(F, \mathcal{L})$ is called with the labeling $\mathcal{L} = (\emptyset, \emptyset, \emptyset, A)$.

Algorithm 1 requires as input an AF $F = (A, R)$ and a labeling $\mathcal{L} = (\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_\Delta}, \mathcal{L}_{undec})$. To compute all *cf2* labelings of an AF F we start $cf2_{\mathcal{L}}(F, \mathcal{L})$ with the initial labeling $\mathcal{L} = (\emptyset, \emptyset, \emptyset, A)$. So the algorithm starts with all arguments labeled $undec$ and will eventually change such labels to either in, out or out_Δ, but other labels are never changed by the algorithm.

The procedure first computes the labels which can be directly propagated according to Proposition 6.9. At the beginning, the two sets X and Y are computed (Lines 1-2). Where X identifies those arguments in $\mathcal{L}_{undec}$ which can directly be labeled with in, and Y identifies those arguments in $\mathcal{L}_{undec}$ which can directly be labeled with out_Δ according to Proposition 6.9. Then in the while loop (Lines 3-6) we first update the labeling according to the sets X, Y and then compute anew the sets X, Y for the updated labeling. We repeat this till a fixed-point is reached, i.e. there are no further labels that can be computed using the rules from Proposition 6.9.

Second, we test whether there are still arguments which can be added to $\mathcal{L}_{in}$ (Line 7-8) and identify them in the set B, these are precisely the arguments in $\mathcal{L}_{undec}$ which are not in conflict with $\mathcal{L}_{in}$. If there are still such arguments, the set C identifies the next SCCs to be labeled, following the definition of *cf2* these are the minimal SCCs, i.e. those SCCs which are not attacked by any argument from an other SCC in B. Note here, C does not contain all arguments of an SCC, but all arguments which can be labeled in. To be more precise, self-attacking arguments are omitted in C. By definition each *cf2*-labeling must be a naive labeling of the sub-framework $F|_C$. Thus in Line 11 a separated procedure identifies all naive labelings of the sub-framework $F|_C$. For each naive labeling $\mathcal{L}'$ we update the actual labeling $\mathcal{L}$ with $\mathcal{L}'$ and call $cf2_{\mathcal{L}}(F, \mathcal{L})$ recursively. Notice, as all this naive labellings differ in at least one argument, this step is a branch between different *cf2* extensions, and thus we never produce the same labeling twice.

EXAMPLE 6.10
Consider the AF illustrated in Figure 28. We call $cf2_{\mathcal{L}}(F, \mathcal{L})$ with the initial labeling $\mathcal{L} = (\emptyset, \emptyset, \emptyset, A)$.

At the beginning we have $X = \emptyset$, $Y = \emptyset$, $B = A$ and $C = \{a, b, c\}$. We invoke the external procedure for computing the naive extensions of $F|_C$ which return three naive labelings $\mathcal{L}_1 = (\{a\}, \{b\}, \emptyset, \{c\})$, $\mathcal{L}_2 = (\{b\}, \{c\}, \emptyset, \{a\})$ and $\mathcal{L}_3 = (\{c\}, \{a\}, \emptyset, \{b\})$. For each of them the actual labeling is updated with $\mathcal{L}' \in naive_{\mathcal{L}}(F|_C)$ and $cf2_{\mathcal{L}}(F, \mathcal{L})$ is called.

- For $\mathcal{L}_1$ this looks as follows. We call $cf2_{\mathcal{L}}(F, \mathcal{L})$ with $\mathcal{L} = (\{a\}, \{b\}, \emptyset, A \setminus \{a, b\})$. Then, $X = \emptyset$, $Y = \emptyset$, $B = \{d, e, f, g, h, i\}$ and $C = \{d\}$. As $F|_C$ consists of the single argument d, we can update the actual labeling to $(\{a, d\}, \{b\}, \emptyset, A \setminus \{a, d\})$ and call $cf2_{\mathcal{L}}$ again.
  - Now, $X = \emptyset$, $Y = \{f\}$ and $\mathcal{L}_{out_\Delta} = \{f\}$. Then, $X = \{g\}$, $Y = \emptyset$ and $\mathcal{L}_{in} = \{a, d, g\}$. Next, $X = \emptyset$, $Y = \{h\}$ and $\mathcal{L}_{out_\Delta} = \{f, h\}$. Then, $X = \{i\}$, $Y = \emptyset$ and $\mathcal{L}_{in} = \{a, d, g, i\}$. Thus we obtain $B = C = \{e\}$ and we can update the labeling and return $(\{a, d, e, g, i\}, \{b\}, \{f, h\}, \{c\})$.
- For $\mathcal{L}_2$ we call $cf2_{\mathcal{L}}(F, \mathcal{L})$ with $\mathcal{L} = (\{b\}, \{c\}, \emptyset, A \setminus \{b, c\})$. Then, $X = \emptyset$, $Y = \{d, e\}$ and $\mathcal{L}_{out_\Delta} = \{d, e\}$. Next, $B = C = \{f, g, h, i\}$ and as $F|_C$ has two naive extensions we can return the two *cf2* labelings $(\{b, f, h\}, \{c, g, i\}, \{d, e\}, \{a\})$ and $(\{b, g, i\}, \{c, f, h\}, \{d, e\}, \{a, c\})$.
- Finally for $\mathcal{L}_3$ we call $cf2_{\mathcal{L}}(F, \mathcal{L})$ with $\mathcal{L} = (\{c\}, \{a\}, \emptyset, A \setminus \{a, c\})$. Then, $X = \emptyset$, $Y = \emptyset$, $B = \{d, e, f, g, h, i\}$ and $C = \{d\}$. Here we have the same set B as in the step above for $\mathcal{L}_1$, which leads us to the *cf2* labeling $(\{c, d, e, g, i\}, \{a\}, \{f, h\}, \{b\})$.

$\diamond$

Finally, notice that for acyclic AFs, the above algorithm will compute the *cf2* labeling solely by applying the rules from Proposition 6.9. Hence, there are no recursive calls and thus the algorithm terminates in polynomial time. We cannot expect a similar behavior for the other tractable fragments from Section 5, i.e. symmetric and bipartite AFs, because these may propose an exponential number of extensions (the tractability for reasoning tasks was via some shortcut preventing us from computing all extensions).

6.3.3 Labeling Algorithm for *stage2*

In this section we adopt the algorithm presented before for *stage2*. We first discuss the necessary modifications resulting Algorithm 2 and then present an illustrative example.

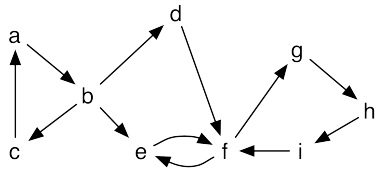


FIG. 28. The argumentation framework F from Example 6.10.

First, as $stage2_{\mathcal{L}}(F) \subseteq cf2_{\mathcal{L}}(F)$ we can use the propagation rules of Proposition 6.9, also for computing $stage2$ labelings. Hence, we only make changes in the second part of the algorithm where we branch between the different extensions. For $stage2$ semantics we have to consider self-attacking arguments in the AF $F|_D$ (Line 10 & 12), as they have an impact on the stage extension of a SCC (while they do not effect the naive extensions). Hence we compute the minimal SCCs including the self-attacking arguments and store them in the set D. Second, instead of computing the naive labelings of $F|_C$ we have to compute the stage labelings of $F|_D$, a labeling-based procedure for this is presented in [14].

Algorithm 2 $stage2_{\mathcal{L}}(F, \mathcal{L})$

Require: AF $F = (A, R)$, labeling $\mathcal{L} = (\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_\Delta}, \mathcal{L}_{undec})$;
Ensure: Return all $stage2$ labelings of F.
1: $X = \{a \in \mathcal{L}_{undec} \mid att(a) \subseteq \mathcal{L}_{out_\Delta}\}$;
2: $Y = \{a \in \mathcal{L}_{undec} \mid \exists b \in \mathcal{L}_{in}, (b, a) \in R, a \not\Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} b\}$;
3: **while** $(X \cup Y) \neq \emptyset$ **do**
4: $\quad \mathcal{L}_{in} = \mathcal{L}_{in} \cup X, \mathcal{L}_{out} = \mathcal{L}_{out} \cup Y, \mathcal{L}_{undec} = \mathcal{L}_{undec} \setminus (X \cup Y)$;
5: $\quad$ update X and Y;
6: **end while**
7: $B = \{a \in \mathcal{L}_{undec} \mid \mathcal{L}_{in} \cup \{a\} \in cf(F)\}$;
8: **if** $B \neq \emptyset$ **then**
9: $\quad C = \{a \in B \mid \nexists b \in B : b \Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} a, a \not\Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} b\}$;
10: $\quad D = C \cup \{a \in \mathcal{L}_{undec} \mid \exists b \in C, a \Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} b, b \Rightarrow_F^{A \setminus \mathcal{L}_{out_\Delta}} a\}$
11: $\quad \mathcal{E} = \emptyset$;
12: $\quad$ **for all** $\mathcal{L}' \in stg_{\mathcal{L}}(F|_D)$ **do**
13: $\quad\quad \mathcal{L}_{in} = \mathcal{L}_{in} \cup \mathcal{L}'_{in}, \mathcal{L}_{out} = \mathcal{L}_{out} \cup \mathcal{L}'_{out}, \mathcal{L}_{undec} = \mathcal{L}_{undec} \setminus (\mathcal{L}'_{in} \cup \mathcal{L}'_{out})$;
14: $\quad\quad \mathcal{E} = \mathcal{E} \cup stage2_{\mathcal{L}}(F, \mathcal{L})$;
15: $\quad$ **end for**
16: $\quad$ **return** $\mathcal{E}$;
17: **else**
18: $\quad$ **return** $\{(\mathcal{L}_{in}, \mathcal{L}_{out}, \mathcal{L}_{out_\Delta}, \mathcal{L}_{undec})\}$;
19: **end if**

To compute all $stage2$ labelings of an AF F the function $stage2_{\mathcal{L}}(F, \mathcal{L})$ is called with the labeling $\mathcal{L} = (\emptyset, \emptyset, \emptyset, A)$.

EXAMPLE 6.11
We illustrate the behavior of Algorithm 2 on the AF F pictured in Figure 29. We start

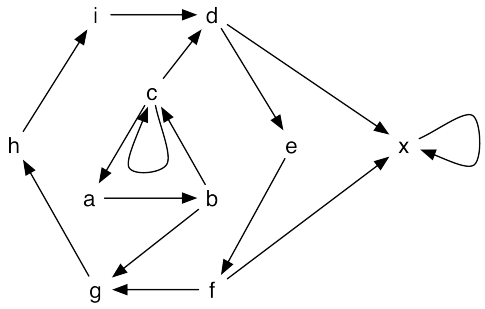


FIG. 29. The argumentation framework F from Example 6.11.

$stage2_{\mathcal{L}}(F, \mathcal{L})$ with the initial labeling $\mathcal{L} = (\emptyset, \emptyset, \emptyset, A)$.

In this first call we have $X = \emptyset$, $Y = \emptyset$, $B = \{a, b, d, e, f, g, h, i\}$ and $C = \{a, b\}$. To complete the inner loop we compute $D = \{a, b, c\}$ which also takes the self-attacking argument c into account. Next we call the external procedure to obtain all stage labelings of the restricted AF $F|_D$ which gives us $\mathcal{L}_1 = (\{a\}, \{b\}, \emptyset, \{c\})$ and $\mathcal{L}_2 = (\{b\}, \{c\}, \emptyset, \{a\})$. Here we have the first branch where we update the actual labeling to the ones obtained from $stg_{\mathcal{L}}(F|_D)$.

- For $\mathcal{L}_1$ we call $stage2_{\mathcal{L}}(F, \mathcal{L})$ with the updated labeling $\mathcal{L} = (\{a\}, \{b\}, \emptyset, A \setminus \{a, b\})$. This leads us to $X = \emptyset$, $Y = \emptyset$ and $B = C = D = \{d, e, f, g, h, i\}$. We call $stg_{\mathcal{L}}(F|_D)$ which returns $\mathcal{L}_{1,1} = (\{e, g, i\}, \{d, f, h\}, \emptyset, \emptyset)$ and $\mathcal{L}_{1,2} = (\{d, f, h\}, \{e, g, i\}, \emptyset, \emptyset)$ as the two stage labelings of $F|_D$. We update the actual labeling with them and branch another time.
  - For $\mathcal{L}_{1,1}$ we call $stage2_{\mathcal{L}}(F, \mathcal{L})$ with $\mathcal{L} = (\{a, e, g, i\}, \{b, d, f, h\}, \emptyset, \{c, x\})$, where we have $X = \emptyset$, $Y = \emptyset$ and $B = \emptyset$. Thus, Algorithm 2 returns the $stage2$ labeling $(\{a, e, g, i\}, \{b, d, f, h\}, \emptyset, \{c, x\})$.
  - For $\mathcal{L}_{1,2}$ we call $stage2_{\mathcal{L}}(F, \mathcal{L})$ with $\mathcal{L} = (\{a, d, f, h\}, \{b, e, g, i\}, \emptyset, \{c, x\})$. Then, $X = \emptyset$, $Y = \{x\}$ and we obtain $\mathcal{L}_{out_\Delta} = \{x\}$. As $B = \emptyset$ we return $(\{a, d, f, h\}, \{b, e, g, i\}, \{x\}, \{c\})$.
- For $\mathcal{L}_2$ we call $stage2_{\mathcal{L}}(F, \mathcal{L})$ with $\mathcal{L} = (\{b\}, \{c\}, \emptyset, A \setminus \{b, c\})$. Then $X = \emptyset, Y = \{g\}$ and $\mathcal{L}_{out_\Delta} = \{g\}$. Next, $X = \{h\}$, $Y = \emptyset$ and $\mathcal{L}_{in} = \{b, h\}$. In the next iteration we have $X = \emptyset, Y = \{i\}$ and $\mathcal{L}_{out} = \{g, i\}$ and then $X = \{d\}, Y = \emptyset$ and $\mathcal{L}_{in} = \{b, d, h\}$. We continue with $X = \emptyset, Y = \{e, x\}$ and $\mathcal{L}_{out} = \{e, g, i, x\}$ and $X = \{f\}, Y = \emptyset$ and $\mathcal{L}_{in} = \{b, d, f, h\}$. Finally $X = \emptyset$, $Y = \emptyset$ and $B = \emptyset$ and the algorithm returns the last $stage2$ labeling of F, namely $(\{b, d, f, h\}, \{c\}, \{e, g, i, x\}, \{a\})$.

$\diamond$

7 Discussion

In this paper we studied abstract argumentation semantics which treat odd- and even-length cycles in a similar fashion, i.e. semantics which are able to select arguments from odd-length cycles. We first highlighted shortcomings of the existing semantics $cf2$ and stage, and then,

	stg	*cf2*	*stage2*
I-max.	Yes	Yes	Yes
Reinst.	No	No	No
Weak reinst.	No	Yes	Yes
$\mathcal{CF}$-reinst.	Yes	Yes	Yes
Direct.	No	Yes	Yes
Succinct.	No	Yes	Yes

TABLE 6. Properties of Naive-based Semantics.

to overcome these shortcomings, we proposed to use the SCC-recursive schema of *cf2* and instantiate the base case with stage semantics, instead of only naive semantics. Thus, we obtained a new sibling semantics of *cf2* which we called *stage2*. We showed that this novel semantics solves the problematic behavior of *cf2* on longer cycles, in particular on cycles of length ≥ 6. Moreover, on coherent AFs (and in particular on odd-cycle free AFs) *stage2* semantics coincide with the standard admissibility based semantics. This guarantees that we do not get an unintended behavior on this class of AFs, and in general on parts of AFs that are coherent. Furthermore, *stage2* satisfies the directionality property as well as the weak reinstatement property which was not the case for stage semantics. A comparison of the properties of stage, *cf2* and *stage2* semantics is provided in Table 6.

The analysis of equivalence showed that *stage2* is the second semantics considered so far where strong equivalence coincides with syntactic equivalence, i.e. there are no redundant attacks at all. Thus, *stage2* semantics also satisfies the succinctness property, which allows to relate the semantics according to how much meaning every attack has for the computation of the extensions.

We provided a comprehensive complexity analysis for *cf2* and *stage2* semantics. A summary of the obtained results for the standard reasoning problems for argumentation semantics and for the investigation of tractable fragments is pictured in Table 7. It turned out that both semantics are computationally hard and *stage2* semantics is even located on the second level of the polynomial hierarchy. Thus *stage2* it is among the hardest but also most expressiveness argumentation semantics. Faced with this in general intractable complexity, we were able to identify tractable fragments for both semantics, namely acyclic, bipartite and symmetric self-attack free frameworks. However, we showed that recent techniques [32] to augment such fragments are not applicable here.

In the implementation part we gave the ASP encodings of *stage2* semantics, where the alternative characterization facilitated this step. As *stage2* is located at the second level of the polynomial hierarchy, we needed more involved programming techniques like the saturation encodings. To simplify those encodings we applied the novel `metasp` optimization front-end from the ASP system `gringo/claspD`. All these encodings are incorporated in the system ASPARTIX and available on the web[10]. We also provided labeling based algorithms for *cf2* and *stage2* to directly compute the respective extensions.

[10] See *http://rull.dbai.tuwien.ac.at:8080/ASPARTIX/* for a web front-end.

	cf2	*stage2*	*stg*
$Ver\sigma$	in P	coNP-c	coNP-c
$Cred\sigma$	NP-c	Σ_2^{P}-c	Σ_2^{P}-c
$Skept\sigma$	coNP-c	Π_2^{P}-c	Π_2^{P}-c
$Exists^{\neg\emptyset}\sigma$	in P	in P	in P
${Cred_\sigma}^{acycl}$	in P	in P	P-c
${Skept_\sigma}^{acycl}$	in P	in P	P-c
${Cred_\sigma}^{even-free}$	NP-c	coNP-h	Σ_2^{P}-c
${Skept_\sigma}^{even-free}$	coNP-c	coNP-h	Π_2^{P}-c
${Cred_\sigma}^{bipart}$	in P	in P	P-c
${Skept_\sigma}^{bipart}$	in P	in P	P-c
${Cred_\sigma}^{sym}$	in P	in $\mathrm{P}/\Sigma_2^{\mathrm{P}}$-c*	in $\mathrm{P}/\Sigma_2^{\mathrm{P}}$-c*
${Skept_\sigma}^{sym}$	in P	in $\mathrm{P}/\Pi_2^{\mathrm{P}}$-c*	in $\mathrm{P}/\Pi_2^{\mathrm{P}}$-c*

TABLE 7: Summary of complexity results (* with self-attacking arguments). An entry $\mathcal{C}$-c denotes completeness for the class $\mathcal{C}$.

7.1 Related Work

Here, we consider different approaches which deal with the problems related to *cycles* in AFs, and in particular with odd-length cycles. Of course the *cf2* semantics is not the only attempt to solve this problem. Bodanza and Tohmé introduced two semantics the *sustainable* and the *tolerant* semantics [10]. While the former basically ignores attacks from self-attacking arguments the latter is in a similar spirit than *cf2* and *stage2*. The intuition there is that the defense of a set of arguments should not be defined in absolute terms but relative to other possible challenging sets of arguments. They propose an application of this semantics in the field of strategic argumentation games, where each player has to choose a set of arguments to confront with and defend against the possible choices of the other agent. As tolerant, *cf2* and stage semantics are of similar fashion one might ask whether some of them coincide. In [10] (Example 27) there is an example for an AF with a tolerant extension which is not a *cf2* extensions. As each *stage2* extension is also a *cf2* extension this also shows that tolerant semantics differ from *stage2*. If we compare tolerant with *cf2* semantics one main difference is that on odd-cycle free AFs tolerant semantics coincides with preferred semantics [11] and thus does not have the problematic behavior on even-length cycles. However, it has the same (problematic) behavior in odd-length cycles of certain length. To best of the authors knowledge there is no systematic analysis of the properties studied in this paper for sustainable and tolerant semantics.

Gabbay introduced several *loop-busting* semantics in [44]. One of them, the LB2 semantics has shown to be equivalent to *cf2*. All these semantics are involved in the *equational approach* to argumentation networks [43]. The author also defines an equational approach

[11] This does not hold for coherent AFs in general, e.g. consider the coherent AF $(\{a, b, c, \}, \{(a, b), (a, c), (c, b), (b, a)\})$ where $\{a\}$ is the only preferred extension but both $\{a\}$ and $\{b\}$ being tolerant extensions.

to $stage2$ semantics in [44], namely LB$2 - stage$. Furthermore in [1] the authors propose the *Shkop* semantics which has been shown to be equivalent to LB4 from the loop-busting semantics in [44].

Roos proposed in [58] the *preferential model semantics* which also handles odd loops in a special way. The motivation for this semantics comes from the preferential model semantics for non-monotonic reasoning systems [51]. There, the attack relation is used to define preferences over states. So, not one argument is preferred over another one, but one prefers a state where the attacking argument is valid, over a state where the attacked argument is valid. This semantics results in different extensions than $cf2$, for example consider the AF $F = (A, R)$ with $A = \{a, b, c\}$ and $R = \{(a, b), (b, c), (c, a), (c, c)\}$. Then, $\{a\}$ and $\{b\}$ are $cf2$ extensions, but only $\{a\}$ is a *pm* extension, because the state a is preferred over the state b, as the only attacker of a is the argument c which is self-attacking.

7.2 Future Work

While the computational complexity of a semantics gives a first impression of its expressiveness the concept of translations between semantics [38, 34] gives a more fine-grained hierarchy. To get an even better feeling how $cf2$ and $stage2$ relates to the other semantics in terms of expressiveness we plan to investigate possible intertranslations between $cf2$, $stage2$ and standard argumentation semantics. Notice that the construction in proof of Theorem 5.2 already provides a translation from stage to $stage2$ semantics. So one particular question of interest, would be whether also the reverse is possible, i.e. whether there is a translation from $stage2$ to stage semantics.

As it turned out that strong equivalence is indeed a very strong condition for many semantics, in case of $cf2$ and $stage2$ it even breaks down to syntactic equivalence, it can be beneficial to relax the notion of equivalence and for example consider a relativized notion, where source and target of attacks are restricted. This can be interesting in the course of two agents, where one can only point attacks from and to a specific set of arguments. Recently, one step towards this direction has already been made in [8] with the normal and strong expansion equivalence. However, the studies of [8] are restricted to admissibility-based semantics.

As the ASP encodings provided for $cf2$ and $stage2$ semantics are rather comprehensive there might be several ways to improve the performance. One direction for future investigations are possible optimizations of the ASP encodings, by using advanced ASP constructs like aggregates and techniques like symmetry breaking, and to investigate how they can improve the performance [48].

One main criticism concerning naive-based semantics comes from the non-abstract setting where one instantiates argumentation frameworks from a knowledge base. One of the most prominent framework for such an instantiation is the ASPIC+ system and is has been shown that both stage and $cf2$ semantics can produce inconsistent solutions when instantiated by ASPIC+ (see [2]). However, the examples given in [2] can not be generalized to $stage2$ semantics and hence it is open issue whether $stage2$ proposes consistent conclusion in the ASPIC+ setting. As on coherent AFs $stage2$ semantics coincides with stable semantics constructing inconsistent solutions seems to me more challenging, if even possible. Finally, as the ASPIC+ framework seems to be tailored to admissibility based semantics, it seems worth to consider $cf2$ and $stage2$ semantics in other instantiation schemes. However, recently Strass [59] proposed an instantiation scheme for abstract dialectical frameworks (ADFs) [11, 12], a generalization of Dung's AFs, which overcomes the problems of ASPIC+

regarding the rationality postulates proposed in [15]. Hence, generalizing $cf2$ and $stage2$ semantics to the ADFs setting is promising direction for future research.

For both $cf2$ and $stage2$ semantics we presented a characterization of the form $\sigma(F) = \{S \mid \tau(F) \cap \theta([[F - \Delta_{F,S}]])\}$. One can see this as a general schema for argumentation semantics, where one can exchange the semantics τ and θ. For naive-based semantics one might choose $\tau = naive$ while for admissible-based semantics one can choose $\tau = adm$. One special instantiation is $stable2(F) = \{S \mid S \in naive(F) \cap stable([[F - \Delta_{F,S}]])\} = \{S \mid S \in adm(F) \cap stable([[F - \Delta_{F,S}]])\}$ and it clearly holds that $stable2(F) = stable(F)$. The investigation of other such combinations might reveal new options.

Acknowledgments

Earlier versions of parts of this paper have been presented at the 14th International Workshop on Non-Monotonic Reasoning (NMR'12) [29] and the Fourth International Conference on Computational Models of Argument (COMMA'12) [28]. The authors want to thank the reviewers of the preceding papers and the current paper for their valuable comments which helped to improve the quality of the paper. Furthermore the authors are grateful to Stefan Woltran for helpful discussions and comments.

References

[1] Michael Abraham, Dov M. Gabbay, and Uri J. Schild. The handling of loops in talmudic logic, with application to odd and even loops in argumentation. In David Rydeheard, Andrei Voronkov, and Margarita Korovina, editors, *Proceedings of Higher-Order Workshop on Automated Runtime Verification and Debugging* (*HOWARD 60*), 2011.

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

[3] Pietro Baroni, Paul E. Dunne, and Massimiliano Giacomin. On the resolution-based family of abstract argumentation semantics and its grounded instance. *Artif. Intell.*, 175(3–4):791–813, 2011.

[4] Pietro Baroni and Massimiliano Giacomin. Solving semantic problems with odd-length cycles in argumentation. In Thomas D. Nielsen and Nevin L. Zhang, editors, *Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2003)*, volume 2711 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2003.

[5] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artif. Intell.*, 171(10-15):675–700, 2007.

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

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

[8] Ringo Baumann. Normal and strong expansion equivalence for argumentation frameworks. *Artif. Intell.*, 193:18–44, 2012.

[9] Trevor J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[10] 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.

[11] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes P. Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 803–809. AAAI Press, 2013.

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

[13] Martin Caminada. An algorithm for computing semi-stable semantics. In Khaled Mellouli, editor, *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, volume 4724 of *Lecture Notes in Computer Science*, pages 222–234. Springer, 2007.

[14] Martin Caminada. An algorithm for stage semantics. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo R. Simari, editors, *Proceedings of the 3rd Conference on Computational Models of Argument (COMMA 2010)*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 147–158. IOS Press, 2010.

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

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

[17] Martin Caminada, Walter A. Carnielli, and Paul E. Dunne. Semi-stable semantics. *Journal of Logic and Computation*, 22(5):1207–1254, 2012.

[18] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric argumentation frameworks. In Lluis Godo, editor, *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, volume 3571 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2005.

[19] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[20] Yannis Dimopoulos and Alberto Torres. Graph theoretical structures in logic programs and Default Theories. *Theor. Comput. Sci.*, 170(1–2):209–244, 1996.

[21] Sylvie Doutre and Jérôme Mengin. Preferred extensions of argumentation frameworks: Query answering and computation. In *IJCAR*, pages 272–288, 2001.

[22] Phan M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[23] Paul E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10–15):701–729, 2007.

[24] Paul E. Dunne and Trevor J. M. Bench-Capon. Complexity and combinatorial properties of argument systems. Technical report, Dept. of Computer Science, University of Liverpool, 2001.

[25] Paul E. Dunne and Trevor J. M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, 141(1/2):187–203, 2002.

[26] Paul E. Dunne and Michael Wooldridge. Complexity of abstract argumentation. In Iyad Rahwan and Guillermo R. Simari, editors, *Argumentation in Artificial Intelligence*, pages 85–104. Springer, 2009.

[27] Wolfgang Dvořák. *Computational Aspects of Abstract Argumentation*. PhD thesis, Vienna University of Technology, 2012.

[28] Wolfgang Dvořák and Sarah A. Gaggl. Computational aspects of cf2 and stage2 argumentation semantics. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 273–284. IOS Press, 2012.

[29] Wolfgang Dvořák and Sarah A. Gaggl. Incorporating stage semantics in the scc-recursive schema for argumentation semantics. In *In Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR 2012)*, 2012.

[30] Wolfgang Dvořák, Sarah A. Gaggl, Johannes P. Wallner, and Stefan Woltran. Making use of advances in answer-set programming for abstract argumentation systems. In *Proceedings of the 19th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2011)*, pages 117–130, 2011.

[31] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. Complexity-sensitive decision procedures for abstract argumentation. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 13th International Conference, KR 2012, Rome, Italy, June 10-14, 2012*, pages 54–64. AAAI Press, 2012.

[32] Wolfgang Dvořák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186(0):157–173, 2012.

[33] Wolfgang Dvořák, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artificial Intelligence*, 186(0):1 – 37, 2012.

[34] Wolfgang Dvořák and Christof Spanring. Comparing the expressiveness of argumentation semantics. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 261–272. IOS Press, 2012.

[35] Wolfgang Dvořák, Stefan Szeider, and Stefan Woltran. Reasoning in argumentation frameworks of bounded clique-width. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo R. Simari, editors, *Proceedings of the 3rd Conference on Computational Models of Argument (COMMA 2010)*, Frontiers in Artificial Intelligence and Applications, pages 219–230. IOS Press, 2010.

[36] Wolfgang Dvořák, Stefan Szeider, and Stefan Woltran. Abstract argumentation via monadic second order logic. In Eyke Hüllermeier, Sebastian Link, Thomas Fober, and Bernhard Seeger, editors, *Proceedings of the 6th International Conference on Scalable Uncertainty Management (SUM 2012)*, volume 7520 of *Lecture Notes in Computer Science*, pages 85–98. Springer, 2012.

[37] Wolfgang Dvořák and Stefan Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.*, 110(11):425–430, 2010.

[38] Wolfgang Dvořák and Stefan Woltran. On the intertranslatability of argumentation semantics. *J. Artif. Intell. Res. (JAIR)*, 41:445–475, 2011.

[39] Uwe Egly, Sarah A. Gaggl, and Stefan Woltran. Aspartix: Implementing argumentation frameworks using answer-set programming. In Maria Garcia de la Banda and Enrico Pontelli, editors, *Proceedings of the 24th International Conference on Logic Programming (ICLP 2008)*, volume 5366 of *Lecture Notes in Computer Science*, pages 734–738. Springer, 2008.

[40] Uwe Egly, Sarah A. Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, 1(2):144–177, 2010.

[41] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.*, 15(3–4):289–323, 1995.

[42] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006.

[43] Dov M. Gabbay. Equational approach to argumentation networks. *Argument and Computation*, 3(2–3):87–142, 2012.

[44] Dov M. Gabbay. The equational approach to cf2 semantics. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 141–152. IOS Press, 2012.

[45] Sarah A. Gaggl and Stefan Woltran. cf2 semantics revisited. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo R. Simari, editors, *Proceedings of the 3rd Conference on Computational Models of Argument (COMMA 2010)*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 243–254. IOS Press, 2010.

[46] Sarah A. Gaggl and Stefan Woltran. Strong equivalence for argumentation semantics based on conflict-free sets. In Weiru Liu, editor, *Proceedings of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2011)*, volume 6717 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2011.

[47] Sarah A. Gaggl and Stefan Woltran. The cf2 argumentation semantics revisited. *Journal of Logic and Computation*, 23(5):925–949, 2013.

[48] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[49] Martin Gebser, Roland Kaminski, and Torsten Schaub. Complex optimization in answer set programming. *Theory and Practice of Logic Programming*, 11(4–5):821–839, 2011.

[50] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3–4):365–386, 1991.

[51] Sarit Kraus, Daniel J. Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artif. Intell.*, 44(1–2):167–207, 1990.

[52] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[53] Sanjay Modgil and Martin Caminada. Proof theories and algorithms for abstract argumentation frameworks. In Iyad Rahwan and Guillermo Simari, editors, *Argumentation in Artificial Intelligence*, pages 105–129. Springer, 2009.

[54] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics And Its Applications. OUP Oxford, 2006.

[55] Samer Nofal, Paul E. Dunne, and Katie Atkinson. On preferred extension enumeration in abstract argumentation. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 205–216. IOS Press, 2012.

[56] Emilia Oikarinen and Stefan Woltran. Characterizing strong equivalence for argumentation frameworks. *Artif. Intell.*, 175(14–15):1985–2009, 2011.

[57] John L. Pollock. Justification and defeat. *Artificial Intelligence*, 67:377–407, 1994.

[58] Nico Roos. The relation between preferential model and argumentation semantics. In *In Proceedings of the 13th International Workshop on Non-Monotonic Reasoning (NMR 2010)*, 2010.

[59] Hannes Strass. Instantiating knowledge bases in abstract dialectical frameworks. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *Proceedings of the Fourteenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XIV)*, volume 8143 of *Lecture Notes in Computer Science*, pages 86–101. Springer, September 2013.

[60] Francesca Toni and Marek Sergot. Argumentation and answer set programming. In Marcello Balduccini and Tran C. Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *Lecture Notes in Computer Science*, pages 164–180. Springer Berlin Heidelberg, 2011.

[61] Bart Verheij. Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In John-Jules Ch. Meyer and Linda C. van der Gaag, editors, *Proceedings of the 8th Dutch Conference on Artificial Intelligence (NAIC 1996)*, pages 357–368. University of Utrecht, 1996.

[62] Bart Verheij. A labeling approach to the computation of credulous acceptance in argumentation. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 623–628, 2007.

[63] Gerard Vreeswijk and Henry Prakken. Credulous and sceptical argument games for preferred semantics. In Manuel Ojeda-Aciego, Inman P. de Guzmán, Gerhard Brewka, and Luís M. Pereira, editors, *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA 2000)*, volume 1919 of *Lecture Notes in Computer Science*, pages 239–253. Springer, 2000.