Artificial Intelligence 307 (2022) 103697

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Advanced algorithms for abstract dialectical frameworks based on complexity analysis of subclasses and SAT solving ☆

Thomas Linsbichler [a], Marco Maratea [b], Andreas Niskanen [c,*], Johannes P. Wallner [d], Stefan Woltran [a]

[a] Institute of Logic and Computation, TU Wien, Austria
[b] DIBRIS, University of Genova, Italy
[c] Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland
[d] Institute of Software Technology, Graz University of Technology, Austria

ARTICLE INFO

Article history:
Received 4 March 2021
Received in revised form 6 January 2022
Accepted 8 March 2022
Available online 15 March 2022

Keywords:
Abstract dialectical frameworks
SAT-based procedures
Complexity analysis

ABSTRACT

Abstract dialectical frameworks (ADFs) constitute one of the most powerful formalisms in abstract argumentation. Their high computational complexity poses, however, certain challenges when designing efficient systems. In this paper, we tackle this issue by (i) analyzing the complexity of ADFs under structural restrictions, (ii) presenting novel algorithms which make use of these insights, and (iii) implementing these algorithms via (multiple) calls to SAT solvers. An empirical evaluation of the resulting implementation on ADF benchmarks generated from ICCMA competitions shows that our solver is able to outperform state-of-the-art ADF systems.

1. Introduction

Over the last two decades, argumentation has become a major field in AI research [9] and an increasing number of applications (see [6] for a survey) encourages the development of computational models of argumentation. Within the field, Dung's abstract argumentation frameworks (AFs for short, [32]) have received notable attention. This is due to two central observations: first, AFs constitute an integral building block in many advanced argumentation formalisms; second, several formalisms from the AI domain, e.g., logic programming or nonmonotonic reasoning principles, can be instantiated via AFs. For both aspects, the variety of available semantics is crucial.

The last years have witnessed a multitude of methods for solving reasoning problems in such frameworks (see [24,22] for surveys). Since 2015, a dedicated competition – the International Competition on Computational Models of Argumentation [70], known as ICCMA – has further fostered development of systems that implement these methods. Hereby, in particular for reasoning problems with beyond-NP complexity, procedures implementing counterexample-guided abstraction refinement (CEGAR) [27,28] approaches, which rely on iterative calls to SAT solvers, have been proven successful: examples include the first CEGAR approach of Cegartix [34] and the system ArgSemSAT [23]; recently this method has been further

☆ This paper is an extended and revised version of a paper that appeared in the Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018) [50].

* Corresponding author.
E-mail addresses: linsbich@dbai.tuwien.ac.at (T. Linsbichler), marco@dibris.unige.it (M. Maratea), andreas.niskanen@helsinki.fi (A. Niskanen), wallner@ist.tugraz.at (J.P. Wallner), woltran@dbai.tuwien.ac.at (S. Woltran).

https://doi.org/10.1016/j.artint.2022.103697

refined resulting in systems like μ-toksia [57,58], which extends Cegartix to tackle the ICCMA "dynamic track" [12], and Taeydennae [8,59], a solver for incomplete and control AFs.

Despite the popularity of AFs, their conceptual simplicity – an AF is just a directed graph with nodes as abstract arguments and edges representing individual attacks – has also often turned out to be overly limiting. Therefore, generalizations of AFs introducing further link types such as support or collective attacks [55,21] or imposing quantitative aspects have been presented. Abstract dialectical frameworks (ADFs for short [18,16]) constitute one of the most comprehensive generalizations of AFs available. By equipping every argument with an acceptance condition, ADFs are capable of modeling arbitrary relations between arguments. Potential applications in legal reasoning [1,2], online dialog systems [53,54], text exploration [20], and the instantiation of defeasible theories [65] have already been proposed; also the relationship to nonmonotonic formalisms has been investigated [3,42]. Indeed, the growing number of potential applications for ADFs calls for efficient systems that can handle ADFs of sizes required in the respective domains.

However, the power of ADFs comes with a price: compared to AFs, the computational complexity increases by one step in the polynomial hierarchy [64] for nearly all reasoning tasks [69,33,38], with the hardest problems being complete for the third level. Therefore, the identification of fragments which yield decreased complexity, or even tractability, is of high interest. So far, only the class of bipolar ADFs [18] has been shown to be unaffected by the increase in complexity compared to AFs. Recent work by [30] has investigated further subclasses but without an explicit complexity analysis. Existing implementations of ADFs deal with the complexity by employing reductions either to QBF [31] or to ASP [68,14], but barely exploit potential shortcuts for ADFs of lower complexity.

In this work, we aim towards more efficient systems for ADFs by generalizing the CEGAR approach due to [34], which so far has not been applied in the context of ADFs. We shall focus on those ADF semantics that are based on 3-valued interpretations; these kind of interpretations are key to generalize AF semantics via the characteristic operator for ADFs (see Section 2 for details). We also make use of conflict-free semantics which provides useful basic properties and will investigate the related naive semantics. Our findings will be presented in terms of the four standard decision problems which are typically studied for abstract argumentation in general [33], and for ADFs in particular [69]. These include the problems of credulous and skeptical acceptance, existence of non-trivial solutions, and the verification problem which asks whether a given interpretation constitutes a solution of interest. Understanding the complexity of the latter problem is crucial in implementations that rely on a guess-and-check strategy, a concept we shall also follow in our CEGAR approach.

More specifically, we provide in this paper (1) a careful complexity analysis of ADF subclasses together with investigations about whether ADFs being "close" to a subclass remain easier; and (2) implementations which make use of these insights. Indeed, the natural choice for implementing a system that exploits a certain form of distance to an easier fragment is an incremental approach. That is, some (easier) core procedure is called several times, where the number of calls increases with the distance to the fragment. Thus, if the instance to solve is already from the fragment, the core procedure is employed only once, but also if the distance is bound to a constant, the number of calls remains bounded as well. This may lead to a form of "complexity-sensitive" solving techniques that exploit certain features of the instance. Examples of such approaches are the already mentioned Cegartix system for AFs [36,34] in the field of argumentation, or loop-formula based approaches in the area of Answer-Set Programming [49,41].

To sum up, our main contributions are as follows:

- We investigate the complexity of ADF subclasses for most of the standard semantics, namely conflict-free, admissible, naive, grounded, complete, and preferred. The subclasses we consider can be divided between syntactic and semantic subclasses. For syntactic subclasses, one takes the link structure of the ADF into account. We shall consider here the classes of acyclic, symmetric, and bipartite ADFs. Subclasses defined on the semantic level are the already mentioned bipolar ADFs but also the class of concise ADFs (this class is defined with respect to a semantics for which the ADFs are required to yield only a single solution). As we will see, in addition to the known result for bipolar ADFs, it is only the classes of acyclic and concise ADFs among the ones mentioned that yield lower complexity compared to the general problem.
- As these fragments impose rather strict requirements, we further show that many of the computational advantages also hold for ADFs which have constant distance to these subclasses. Depending on the actual class, we will define suitable notions of distance, and we provide a complexity landscape for all the semantics considered in this work. It is interesting to mention that depending on the actual class, we have the case that distance does not matter in terms of complexity (the case of bipolar ADFs) or yields complexity between the considered fragment and the general case (this occurs for acyclic and concise ADFs).
- We design novel CEGAR algorithms for ADF reasoning, in particular for (but not limited to) the tasks of skeptical and credulous acceptance under preferred semantics. We provide an implementation based on incremental SAT solving [37], which is not only novel in employing SAT solving for ADF reasoning problems, but is also complexity sensitive as it takes advantage of the input ADFs being in (low distance to) advantageous fragments (namely, bipolar ADFs). Our system is available in open source at: https://bitbucket.org/andreasniskanen/k-adf.
- We provide an experimental evaluation of our new approach on ADF benchmarks generated from ICCMA'17 and ICCMA'19 competitions, which shows that it is capable of outperforming state-of-the-art systems for ADFs.

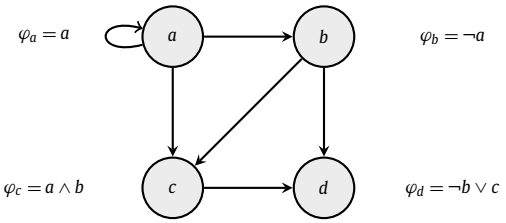

Fig. 1. Example ADF.

The paper is structured as follows. Section 2 contains needed preliminaries about ADFs, together with the considered semantics and reasoning tasks. Then, Section 3 presents complexity results for some subclasses of ADFs, namely acyclic, symmetric, bipartite, concise, and bipolar ADFs, while Section 4 extends such results to ADFs having a constant distance to such subclasses. Section 5 develops encodings and algorithms for solving the hard reasoning tasks of the resulting picture, whose implementation and experiments in comparison to other approaches are presented in Section 6. The paper ends by providing a discussion about related approaches and papers in Section 7, and by drawing conclusions and possible topics for future research in Section 8.

This paper is an extended and revised version of [50], the main improvements over it being: (i) the investigation of two more ADF subclasses; in fact, we provide in Section 3.2 and 3.3 novel transformations from general ADFs to bipartite and symmetric ADFs which show that an extension of the proposed method to these subclasses is not feasible; (ii) the inclusion of proofs for all propositions and theorems; (iii) a new section describing our system implementation, including the improvements over [50]; (iv) experiments on ICCMA'19 benchmarks; and (v) a devoted related work section. Moreover, all parts of the paper were revised in order to improve readability, in particular Section 5 with more explanation and details for encodings and algorithms.

2. Abstract dialectical frameworks

Abstract Dialectical Frameworks (ADFs) have been first introduced by [18], and subsequently been extended and revised [15,16]. We recall the syntax and semantics of ADFs from [16].

We assume a fixed and finite set A which consists of arguments, statements, or positions which shall be represented by an ADF. For the sake of brevity and uniformity, we will refer to elements of A as arguments. Formally, an ADF is a triple $D = (A, L, C)$ where A is a set of arguments, $L \subseteq A \times A$ is a set of (dependency) links, and $C = \{\varphi_a\}_{a \in A}$ is a collection of acceptance conditions for arguments $a \in A$, each given by a propositional formula over the parents of an argument: $par_D(a) = \{b \in A \mid (b, a) \in L\}$.

Example 2.1. In Fig. 1, one can see a graphical representation of an ADF $D = (A, L, C)$ with four arguments $A = \{a, b, c, d\}$, dependencies between arguments shown by the directed edges in the graph, and acceptance conditions shown as Boolean formulas close to the argument nodes. For instance, argument c depends on arguments a and b, i.e., $par_D(c) = \{a, b\}$. Concretely, the acceptance condition of c, denoted by φ_c, is $a \wedge b$.

As apparent from the definition, one can read dependency links from the acceptance conditions (i.e., for each ADF D we have $par_D(x)$ is equal to the Boolean variables occurring in φ_x). While in some other works in the literature the links are not part of a given ADF, we keep the definition of ADFs as a triple of arguments, links, and acceptance conditions to later highlight certain properties of the underlying directed graph given by the arguments and links. Nevertheless, when referring to concrete example ADFs, we oftentimes only define acceptance conditions explicitly and implicitly define links via the variables of the formulas.

The semantics of ADFs, as defined by [16], are based on (collections of) three-valued interpretations. An interpretation is a function I mapping arguments to one of the three truth values $I : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. That is, an interpretation maps each argument to either true ($\mathbf{t}$), false ($\mathbf{f}$), or undefined ($\mathbf{u}$). An interpretation I is *two-valued* if $I(a) \in \{\mathbf{t}, \mathbf{f}\}$ for all $a \in A$, and *trivial*, denoted by $I_{\mathbf{u}}$, if $I(a) = \mathbf{u}$ for all $a \in A$. We denote the update of an interpretation I with truth value $\mathbf{x} \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ for argument b by $I|^b_{\mathbf{x}}$, i.e., $I|^b_{\mathbf{x}}(b) = \mathbf{x}$ and $I|^b_{\mathbf{x}}(a) = I(a)$ for $a \neq b$.

For a two-valued interpretation I, $I(\varphi)$ extends to the evaluation of a formula φ under I as usual. Interpretation I is equally or more informative than J, denoted by $J \leq_i I$, if $J(a) \in \{\mathbf{t}, \mathbf{f}\}$ implies $J(a) = I(a)$ for all $a \in A$. We denote by $<_i$ the strict version of $\leq_i$, i.e., $J <_i I$ if $J \leq_i I$ and $\exists a \in A$ such that $J(a) = \mathbf{u}$ and $I(a) \in \{\mathbf{t}, \mathbf{f}\}$. A completion of an interpretation I is a two-valued interpretation J such that $I \leq_i J$. In Fig. 2, an example interpretation, and more informative interpretations and completions of this interpretation are shown. Further, $\varphi[I]$ is the formula obtained from φ with each argument that I assigns to either true or false being replaced by the corresponding truth constant, i.e., $\varphi[I] = \varphi[x \mapsto \top \mid I(x) = \mathbf{t}][x \mapsto \bot \mid I(x) = \mathbf{f}]$; arguments assigned to undefined are not modified in the resulting formula.

Key definition for the semantics of ADFs is the characteristic operator Γ_D, which maps interpretations to updated interpretations.

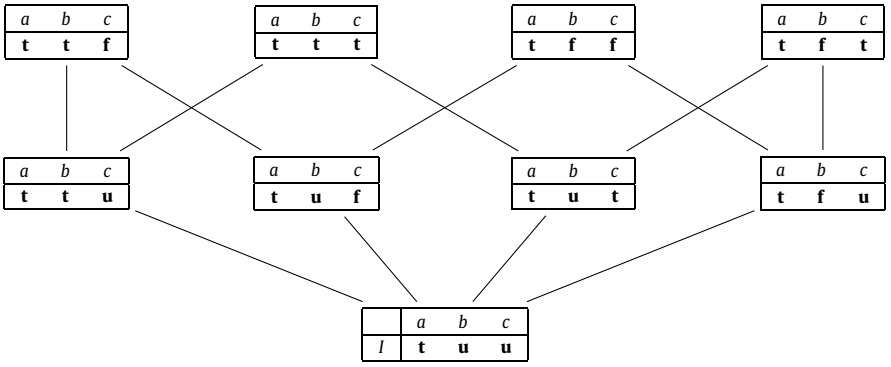

Fig. 2. Information ordering for three-valued interpretations. Top row shows all completions of the three-valued interpretation I.

Definition 2.2. Let $D = (A, L, C)$ be an ADF and v a three-valued interpretation over A. Define the characteristic operator $\Gamma_D(I) = J$ as

$$J(a) = \begin{cases} \mathbf{t} & \text{if } \varphi_a[I] \text{ is a tautology} \\ \mathbf{f} & \text{if } \varphi_a[I] \text{ is unsatisfiable} \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

The intuition behind the characteristic operator is that an argument a is assigned true (false) if all completions of the given interpretation I satisfy (do not satisfy) the acceptance condition of a.

Example 2.3. Consider ADF D from Example 2.1, and interpretation $I_1 = I_{\mathbf{u}}$ (assigning all arguments to undefined). We have $\Gamma_D(I_1) = I_1$. Consider, e.g., argument a with $\varphi_a = a$. It holds that $\varphi_a[I_1] = \varphi_a$, which is neither a tautology nor unsatisfiable. Thus, $\Gamma_D(I_1)(a) = \mathbf{u} = I_1(a)$.

Let us also consider $I_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\}$. We have $\Gamma_D(I_2) = I_3$ with $I_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\}$. For instance, $\varphi_c[I_2] = \top \wedge \bot$ (since $\varphi_c = a \wedge b$). Thus, $\Gamma_D(I_2)(c) = \mathbf{f} = I_3(c)$.

We are now ready to define several of the semantics of ADFs. A semantics σ maps an ADF to a set of three-valued interpretations over the arguments of the given ADF. We consider in this paper the admissible (*adm*), complete (*com*), grounded (*grd*), and preferred (*prf*) semantics, defined next.

Definition 2.4. Given an ADF D, a three-valued interpretation I

- is admissible in D if $I \leq_i \Gamma_D(I)$;
- is complete in D if $I = \Gamma_D(I)$;
- is grounded in D if I is the least fixed-point of Γ_D;
- is preferred in D if I is $\leq_i$-maximal admissible in D.

That is, a three-valued interpretation is admissible if the update via the characteristic operator produces a more or equally informative interpretation. Or, in other words, an interpretation is admissible if there is no argument assigned to either **t** or **f** such that the update via the characteristic operator for this argument results in either the opposite truth value, or undefined (which amounts to a certain conflict in assigning the value). Complete interpretations are fixed-points of Γ_D, the grounded interpretation is the least fixed-point of Γ_D (which always exists and is unique), and preferred interpretations are information maximal admissible interpretations, or equivalently, information maximal complete interpretations.

Example 2.5. Let us have a look at the ADF from Example 2.1 again, and consider several three-valued interpretations and whether they are part of a semantics. In Table 1, we show five interpretations $I \in \{I_1, I_2, I_3, I_4, I_5\}$, the result of applying the characteristic operator $\Gamma_D(I)$, and whether the interpretation is part of a semantics. For instance, I_1 is the unique grounded interpretation, which in this case is the trivial interpretation. The interpretations I_3 and I_4 are both complete and preferred. Interpretation I_5 is not admissible (and, thus, not complete, grounded, or preferred), since $\Gamma_D(I_5) = I_1 = I_{\mathbf{u}}$. To see this, consider $\varphi_b = \neg a$: $\varphi_b[I_5] = \varphi_b$ and this formula is neither tautological nor unsatisfiable. Overall, the example ADF D has three complete interpretations and two preferred interpretations (all shown in Table 1), and eleven admissible interpretations.

Table 1
Interpretations for ADF from Example 2.1and admissibility-based semantics.

interpretation	a	b	c	d	$\Gamma(\cdot)$	part of semantics
I_1	**u**	**u**	**u**	**u**	$\Gamma(I_1) = I_1$	*adm*, *com*, *grd*
I_2	**t**	**f**	**u**	**u**	$\Gamma(I_2) = I_3$	*adm*
I_3	**t**	**f**	**f**	**t**	$\Gamma(I_3) = I_3$	*adm*, *com*, *prf*
I_4	**f**	**t**	**f**	**f**	$\Gamma(I_4) = I_4$	*adm*, *com*, *prf*
I_5	**u**	**t**	**u**	**f**	$\Gamma(I_5) = I_1$	-

Table 2
Interpretations for ADF from Example 2.1and conflict-free-based semantics.

interpretation	a	b	c	d	part of semantics
J_1	**u**	**u**	**u**	**u**	*cf*, *adm*
J_2	**t**	**f**	**f**	**t**	*cf*, *nai*, *adm*, *prf*
J_3	**f**	**t**	**f**	**f**	*cf*, *nai*, *adm*, *prf*
J_4	**t**	**u**	**f**	**f**	*cf*, *nai*
J_5	**t**	**u**	**t**	**t**	*cf*, *nai*
J_6	**u**	**t**	**t**	**t**	*cf*, *nai*
J_7	**f**	**t**	**u**	**t**	*cf*, *nai*
J_8	**f**	**u**	**f**	**t**	*cf*, *nai*
J_9	**t**	**t**	**u**	**u**	-

In this paper we also consider conflict-free and naive interpretations, analogous to the concepts on argumentation frameworks. We recall conflict-free and naive semantics of ADFs as defined by [69], which they called "ultimate" conflict-free and naive semantics originally.[1] For the sake of readability and uniformity, we drop the word "ultimate", and refer to these semantics simply as conflict-free and naive semantics.

Definition 2.6. Let $D = (A, L, C)$ be an ADF, and I a three-valued interpretation. Interpretation I is conflict-free in D if, for all $a \in A$,

- $I(a) = \mathbf{t}$ implies $\varphi_a[I]$ is satisfiable, and
- $I(a) = \mathbf{f}$ implies $\varphi_a[I]$ is refutable.

A conflict-free interpretation I in D is naive in D if there is no conflict-free interpretation J in D such that $I <_i J$.

Recall that a formula is refutable if the formula is not a tautology. That is, conflict-free interpretations are defined by weakening the condition of admissible interpretations. Concretely, this definition differs from admissibility by requiring satisfiability instead of a tautology and refutability instead of unsatisfiability. The property of being naive is then defined as being maximal conflict-free w.r.t. the information order.

Example 2.7. Let us consider several interpretations and see whether they are conflict-free or naive (see Table 2). First, consider interpretation $J_1 = I_{\mathbf{u}}$: this interpretation is trivially conflict-free (and admissible) by definition of conflict-freeness. In the table, we list all naive interpretations $J_2, \ldots, J_8$ of the ADF from Example 2.1. For each of these interpretations we also list whether they are in addition admissible or not. By definition, if an interpretation is admissible then this interpretation is also conflict-free. In the example ADF, all preferred interpretations are also naive. There are five more naive interpretations $J_4, \ldots, J_8$, which are not admissible. Let us have a look at one of these. Consider $J_6 = \{a \mapsto \mathbf{u}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{t}\}$, i.e., a is assigned to be undefined, and all others are assigned to be true. While not admissible ($\Gamma_D(J_6)(b) = \mathbf{u}$ since $\varphi_b[J_6] = \neg a[J_6] = \neg a$) this interpretation is conflict-free. All acceptance conditions for b, c, and d are satisfiable under the partial evaluation by J_6: $\varphi_c[J_6] = a \wedge b[J_6] = a \wedge \top$ and $\varphi_d[J_6] = (\neg b \vee c)[J_6] = \neg\top \vee \top \equiv \top$. To see that J_6 is naive, consider any J such that $J_6 <_i J$, implying that J assigns all b, c, and d to true, but a to either true or false. Consider first $J(a) = \mathbf{t}$. Then $\varphi_b[J] = \neg a[J] = \neg\top \equiv \bot$, which contradicts assignment of b to true. Consider the other case, i.e., $J(a) = \mathbf{f}$. Then $\varphi_c[J] = a \wedge b[J] = \bot \wedge \top \equiv \bot$, which contradicts assignment of c to true. This implies that any more informative interpretation of J_6 is not conflict-free, and thus J_6 is naive. Finally, J_9 is an example interpretation that is not conflict-free: $\varphi_b[J_9] \equiv \bot$, contradicting the assignment of b to true. Overall, there are 42 conflict-free interpretations of the example ADF.

We refer to the set of all conflict-free, naive, admissible, complete, grounded, and preferred interpretations of an ADF D as $cf(D)$, $nai(D)$, $adm(D)$, $com(D)$, $grd(D)$, and $prf(D)$, respectively. In any ADF D, it holds that $prf(D) \subseteq com(D) \subseteq adm(D) \subseteq cf(D)$.

[1] A different definition of conflict-free and naive semantics for three-valued interpretations is given by [38,39], where an argument can be assigned to false if the partially evaluated acceptance condition is unsatisfiable.

Important reasoning problems in ADFs are credulous and skeptical acceptance of arguments, existence of a non-trivial interpretation, and verification of an interpretation. They are defined as follows for semantics σ, given ADF $D=(A, L, C)$:

- $Cred_\sigma$: given $a \in A$, is there an $I \in \sigma(D)$ such that $I(a) = \mathbf{t}$?
- $Skept_\sigma$: given $a \in A$, is $I(a) = \mathbf{t}$ for all $I \in \sigma(D)$?
- $Exists_\sigma$: is there an $I \in \sigma(D)$ with $I \neq I_\mathbf{u}$?
- Ver_σ: given an interpretation I, is $I \in \sigma(D)$?

Example 2.8. Consider again our ADF D from Example 2.1. Since there is an admissible interpretation assigning a to true (e.g., $I_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\}$, see Table 1) it holds that a is credulously accepted under admissible semantics in the ADF D. Argument c is not credulously accepted under admissible semantics in D, since there is no admissible interpretation of D assigning c to true. This can be seen by looking at the preferred interpretations of D, which all assign c to false. Any admissible interpretation must be equally or less informative than at least one preferred interpretation. No argument is skeptically accepted under preferred semantics in D: for each argument there is a preferred interpretation assigning that argument not to true (e.g., I_4 assigns a to false).

Under certain semantics we can find relations between reasoning tasks. Credulous acceptance under admissible, complete, and preferred semantics coincides, which is due to the fact that for each admissible (complete) interpretation there is a preferred interpretation that is equally or more informative. Similarly, credulous acceptance under conflict-free and naive semantics coincides. The same coincidences hold for existence of non-trivial interpretations: there is a non-trivial admissible interpretation iff there is a non-trivial complete interpretation iff there is a non-trivial preferred interpretation. Likewise, there is a non-trivial conflict-free interpretation iff there is a non-trivial naive interpretation. Under grounded semantics, credulous and skeptical acceptance is the same (there is only one grounded interpretation), and, furthermore, skeptical acceptance under complete and grounded semantics coincides, since the grounded interpretation is the $\leq_i$-minimal complete interpretation. Finally, the problem of deciding whether an argument is skeptically accepted under conflict-free or admissible semantics is trivial: the trivial interpretation $I_\mathbf{u}$ is always conflict-free and admissible, and, thus, no argument is skeptically accepted under conflict-free or admissible semantics. We summarize these results in the following proposition.

Proposition 2.9. *It holds that*

- $Cred_{adm}$, $Cred_{com}$, *and* $Cred_{prf}$ *coincide,*
- $Exists_{adm}$, $Exists_{com}$, *and* $Exists_{prf}$ *coincide,*
- $Cred_{cf}$ *and* $Cred_{nai}$ *coincide,*
- $Exists_{cf}$ *and* $Exists_{nai}$ *coincide,*
- $Cred_{grd}$, $Skept_{grd}$, *and* $Skept_{com}$ *coincide, and*
- $Skept_{adm}$ *and* $Skept_{cf}$ *are trivial.*

Complexity of reasoning in ADFs has been analyzed by [69]. Towards recalling their main results, we recap some complexity classes, but refer for the reader for a thorough introduction to complexity to other sources (e.g., [5,60]). We assume some familiarity with decision problems, complexity classes, and oracles, in particular (polynomial) reductions and complexity classes P and NP, which contain problems that can be solved via a deterministic algorithm in polynomial time and via a non-deterministic algorithm in polynomial time, respectively. The class coNP is the complementary class of NP. Further, we utilize some complexity classes of the polynomial hierarchy. A problem is in DP if the problem is the intersection of a problem in NP and a problem in coNP. A problem is in Δ_2^P if there is a deterministic polynomial time algorithm that solves the problem that may access an NP oracle. Similarly, a problem is in Σ_i^P if the problem can be solved via a non-deterministic polynomial time algorithm that can access a Σ_{i-1}^P oracle, with $\Sigma_0^\mathsf{P} = P$ and $\Sigma_1^\mathsf{P} = \mathsf{NP}$. The classes Π_i^P are the complementary classes to Σ_i^P.

The complexity landscape of reasoning in ADFs is shown in Table 3. Except for the trivial tasks, all reasoning in ADFs is at least NP or coNP hard. Many tasks, e.g., credulous acceptance under admissible semantics, are hard for a class on the second level of the polynomial hierarchy. With respect to complexity classes, the most complex task is skeptical acceptance under preferred semantics, which is Π_3^P-complete.

3. Complexity of reasoning in subclasses of ADFs

In this section, we investigate the complexity of several subclasses of ADFs. We will consider (i) syntactic subclasses that are defined along the structure of links in the ADFs, and (ii) semantic subclasses which rely on the nature of the acceptance conditions, or more generally, on interpretations w.r.t. different semantics. For class (i), we borrow concepts from standard AF terminology and study acyclic, bipartite, and symmetric ADFs. Class (ii) includes bipolar ADFs which restrict the links

Table 3
Complexity of ADF reasoning [Strass and Wallner, 2015].

σ	$Cred_\sigma$	$Skept_\sigma$	$Exists_\sigma$	Ver_σ
cf	NP-c	trivial	NP-c	NP-c
nai	NP-c	Π_2^P-c	NP-c	DP-c
adm	Σ_2^P-c	trivial	Σ_2^P-c	coNP-c
grd	coNP-c	coNP-c	coNP-c	DP-c
com	Σ_2^P-c	coNP-c	Σ_2^P-c	DP-c
prf	Σ_2^P-c	Π_3^P-c	Σ_2^P-c	Π_2^P-c

Table 4
Complexity of subclasses of ADFs. Results for bipolar ADFs are due to [69]. † problem is also coNP-hard; entries in parentheses denote straightforward upper bounds from general ADFs.

σ	$Cred_\sigma$	$Skept_\sigma$	$Exists_\sigma$	Ver_σ	$Cred_\sigma$	$Skept_\sigma$	$Exists_\sigma$	Ver_σ
	acyclic				*k*-acyclic			
cf	NP-c	trivial	trivial	NP-c	NP-c	trivial	in P	NP-c
nai	NP-c	coNP-h	trivial	DP-c	NP-c	coNP-h	in P	DP-c
adm	in P	trivial	trivial	coNP-c	in Δ_2^P †	trivial	in Δ_2^P †	coNP-c
grd	in P	in P	trivial	in P	coNP-c	coNP-c	coNP-c	coNP-h
com	in P	in P	trivial	in P	in Δ_2^P †	coNP-c	in Δ_2^P †	coNP-h
prf	in P	in P	trivial	in P	in Δ_2^P †	in Δ_2^P †	in Δ_2^P †	in Δ_2^P †
	bipolar				*k*-bipolar			
cf	in P	trivial	in P	in P	in P	trivial	in P	in P
nai	in P	coNP-c	in P	in P	in P	coNP-c	in P	in P
adm	NP-c	trivial	NP-c	in P	NP-c	trivial	NP-c	in P
grd	in P	in P	in P	in P	in P	in P	in P	in P
com	NP-c	in P	NP-c	in P	NP-c	in P	NP-c	in P
prf	NP-c	Π_2^P-c	NP-c	coNP-c	NP-c	Π_2^P-c	NP-c	coNP-c
	σ-concise				k-σ-concise			
cf	trivial	trivial	trivial	trivial	NP-c	trivial	NP-c	NP-c
nai	NP-c	NP-c	NP-c	DP-c	NP-c	in Δ_2^P	NP-c	DP-c
adm	trivial	trivial	trivial	trivial	(in Σ_2^P)	trivial	(in Σ_2^P)	coNP-c
grd	coNP-c	coNP-c	coNP-c	DP-c	coNP-c	coNP-c	coNP-c	DP-c
com	coNP-c	coNP-c	coNP-c	DP-c	coNP-h	coNP-c	coNP-h	DP-c
prf	in Σ_2^P †	in Σ_2^P †	in Σ_2^P †	in Π_2^P †	coNP-h	in Δ_3^P †	coNP-h	coNP-h

between arguments to be of a supportive or a attacking nature[2] as well as σ-concise ADFs, i.e., ADFs for which exactly one σ-interpretation exists where σ ranges over the semantics as defined in the previous section.

Our strategy is as follows. We want to understand which of the aforementioned subclasses are promising for our purpose to design advanced algorithms for ADFs. To this end, we require a decrease in the complexity for the subclasses compared to the general case as provided in Table 3. In particular, we are aiming at decreasing the complexity for the central semantics of complete, grounded and preferred interpretations. As we will see, this is the case for the classes of acyclic, bipolar, and concise ADFs only. For these three subclasses we provide in this section a (nearly) complete complexity classification along the standard decision problems for all semantics. In Section 4 we will extend our analysis to ADFs which are close to these classes. Table 4 gives our results for these three classes as provided in this (tables on the left-hand-side) and the subsequent section (tables on the right-hand-side).

3.1. Acyclic ADFs

The first subclass of ADFs we consider are acyclic ADFs. In acyclic ADFs the underlying directed graph, given by arguments and links, is acyclic, formally defined next.

Definition 3.1. An ADF $D = (A, L, C)$ is acyclic if the directed graph (A, L) is acyclic.

Example 3.2. Consider an example acyclic ADF as shown in Fig. 3. The acyclic ADF is arranged as a directed acyclic graph (DAG), with "leaf" arguments a, b, and c, internal node arguments d and e, and root argument f. The acceptance conditions of the leaf arguments are all constant functions, i.e., Boolean formulas without variables. Acceptance conditions of non-leaf arguments depend only on parent arguments that are shown lower in the tree. In this ADF the grounded interpretation can be straightforwardly computed: for leaf arguments, assign the same truth value as their constant acceptance condition, and then propagate the truth values in a bottom-up manner in the tree. This yields the grounded interpretation $I = \{a \mapsto$

[2] This class of ADFs was already introduced by [18] as a vehicle to define their notion of stable semantics; the class was later studied in more detail by [69].

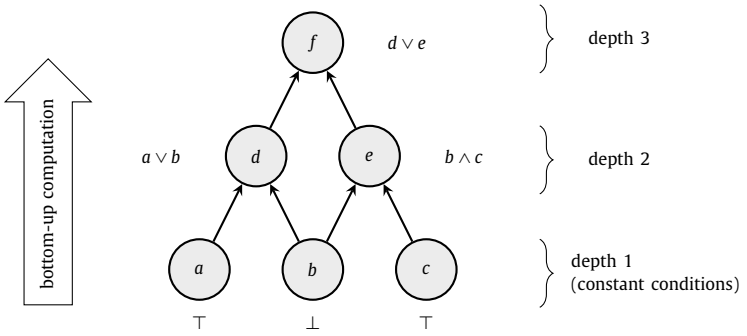

Fig. 3. Bottom-up computation for complete semantics in an acyclic ADF.

$\mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{t}, d \mapsto \mathbf{t}, e \mapsto \mathbf{f}, f \mapsto \mathbf{t}\}$. In this acyclic ADF the interpretation I is also the unique complete and preferred interpretation. In contrast, the naive semantics does not yield a unique naive interpretation, in fact six naive interpretations exist for the acyclic ADF, among them the following: $J = \{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}, e \mapsto \mathbf{t}, f \mapsto \mathbf{t}\}$.

As the preceding example suggests, in acyclic ADFs we have arguments without parents (initial arguments) whose acceptance conditions are constant. These are the leaf arguments (see also Fig. 3). Further, as illustrated in the preceding example, a bottom-up computation yields in general the unique complete (grounded, preferred) interpretation of any acyclic ADF. Furthermore, the unique preferred interpretation is, in fact, two-valued (i.e., all arguments are assigned either true or false). The formal result underlying this property was shown by [45] and [30].

Proposition 3.3 *([30, rephrased from Proposition 1]). Any acyclic ADF has a unique complete interpretation, and this complete interpretation is two-valued.*

Since we will use similar ideas of why this proposition holds for other results, we present a variant of the proof by [45] and [30], for the sake of completeness.

Proof. Given any acyclic ADF $D = (A, L, C)$, we partition the arguments into sets according to their "depth" in the ADF, starting from arguments without parents. A path from an argument s_1 to a argument s_n, in D, is defined as $p = (s_1, s_2, ..., s_{n-1}, s_n)$ with each $s_i \in A$ and for all s_i, s_{i+1} $(1 \leq i \leq n-1)$ we have $s_i \in par_D(s_{i+1})$. The length of p (denoted as $|p|$) is n. We define partition $S_1 = \{s \in S \mid par_D(s) = \emptyset\}$ (depth 1 for all arguments without parents), and $S_i = \{s \in S \mid max\{|p = (s', ..., s)| \mid par_D(s') = \emptyset\} = i\}$, i.e., S_i contains all arguments where the maximum length of paths to that argument from an argument without parents (leaf in the graph) is i. It holds that $A = S_1 \cup S_2 \cup \cdots \cup S_n$ for some finite n (recall that $|A|$ is finite), that $S_i \cap S_j = \emptyset$ for $i \neq j$, and that for each $a \in S_i$ we have $par_D(a) \subseteq S_1 \cup \cdots \cup S_{i-1}$. Further, recall that by $I_{\mathbf{u}}$ we denote the interpretation assigning all arguments in D to $\mathbf{u}$.

Within this proof, we use the model semantics of ADFs. A two-valued interpretation I is a model of a given ADF $D = (A, L, C)$ iff for each $a \in A$ it holds that

- $I(a) = \mathbf{t}$ implies $\varphi_a[I]$ is a tautology and
- $I(a) = \mathbf{f}$ implies $\varphi_a[I]$ is unsatisfiable.

Equivalently, I is a model iff I is complete and two-valued. A model of an ADF D is a preferred interpretation of D.

We show, by induction, that an acyclic ADF has a two-valued grounded interpretation, which, in turn, implies that the grounded interpretation is a model of the acyclic ADF, and that the ADF has a unique model. Formally, we show:

Property: let $D = (A, L, C)$ be an acyclic ADF. It holds that $\Gamma^i_D(I_{\mathbf{u}})(s) \neq \mathbf{u}$ for any $s \in S_1 \cup \cdots \cup S_i$. That is, after i applications of the function, all arguments within depth i have been assigned a value differently to $\mathbf{u}$.

Before showing the property, recall that for $I \leq_i J$ we have $\Gamma_D(I) \leq_i \Gamma_D(J)$ for any D (Γ_D is $\leq_i$-monotone). In particular, this implies $\Gamma^i_D(I_{\mathbf{u}}) \leq_i \Gamma^j_D(I_{\mathbf{u}})$ for $i \leq j$.

Induction base: let $s \in S_1$. It holds that $par_D(s) = \emptyset$, and, thus, for all interpretations I, it holds that $\Gamma_D(I)(s) = \Gamma^1_D(I)(s) = v$ maps to a unique value $v \in \{\mathbf{t}, \mathbf{f}\}$.

Induction step: assume the induction hypothesis holds, i.e., assume that the property holds for i. Consider now $s \in S_{i+1}$: by induction hypothesis, acyclicity, and by construction of the partitions, it holds that $par_D(s) \subseteq S_1 \cup \cdots \cup S_i$ (all parents are in partitions strictly lower than $i+1$). By assumption, it holds that for all $s' \in par_D(s)$ we have $\Gamma^i(I_{\mathbf{u}})(s') \neq \mathbf{u}$. This implies that for all completions of $\Gamma^i(I_{\mathbf{u}})$ the acceptance condition of s evaluates to the same value (the acceptance condition is fully determined by two-valued assignments on $par_D(s)$; each of $par_D(s)$ is assigned the same value in all completions). Thus, $\Gamma^{i+1}(I_{\mathbf{u}})(s) \neq \mathbf{u}$.

The previous line of reasoning proves the property. Suppose that G is the grounded interpretation and there is an argument a such that $G(a) = \mathbf{u}$ (i.e., the grounded interpretation does not assign true or false to a). Due to acyclicity, we have a is in some partition S_i. W.l.o.g., let a be an argument in partition S_i assigned to undefined by G such that there is no argument b in partition S_j assigned undefined by G with $j < i$ (i.e., regarding to partition "depth" a is at minimum depth). It holds that G assigns all arguments in $S_1 \cup \cdots \cup S_{i-1}$ to true or false. But then $\varphi_a[G]$ is either a tautology or unsatisfiable, a contradiction that $G(a) = \mathbf{u}$. Thus, the grounded interpretation G of D assigns each argument in A a value differently to $\mathbf{u}$.

Finally, if $I \in com(D)$ is two-valued, then there is no $J \in com(D)$ with $I <_i J$ (two-valued interpretations are information maximal). Thus, $|com(D)| = 1$ (since the information minimal complete interpretation is two-valued). Since every model of an ADF is a complete interpretation, we infer that there is at most one model in an acyclic ADF. The grounded interpretation is two-valued (as shown above) in an acyclic ADF. This means the grounded interpretation is a model of the ADF. □

The main insight of the proof of the preceding proposition is that when iteratively applying Γ_D to $I_{\mathbf{u}}$ one directly computes the unique preferred interpretation. To see this, consider the depth of arguments, defined as the longest directed path from an initial argument (argument without parents) to that argument (see Fig. 3 for an illustration). Computing $\Gamma_D(I_{\mathbf{u}})$ directly yields a truth value for all initial arguments (these are always tautological or unsatisfiable). Applying the characteristic operator at most twice yields a truth value for all arguments of depth two, and so on for further arguments of deeper depths. Thus, based on the proof of the preceding result, one can compute the grounded (unique preferred) interpretation in polynomial time, since the depth is bounded by the number of arguments.

Corollary 3.4. *One can compute the unique preferred interpretation of acyclic ADFs in polynomial time.*

Furthermore, credulous and skeptical acceptance coincides for all complete-based semantics, i.e., for complete, grounded, and preferred semantics. Moreover, credulous acceptance under admissible, complete, and preferred semantics coincides. Therefore, the associated tasks can be computed in polynomial time. The task of skeptical acceptance under admissible semantics is trivial.

Corollary 3.5. *In acyclic ADFs credulous acceptance can be decided in polynomial time for admissible, grounded, complete, and preferred semantics, and skeptical acceptance can be decided in polynomial time for grounded, complete, and preferred semantics.*

Thus, for acyclic ADFs we see a drastic drop in the complexity of acceptance problems for the most important semantics compared to the case of general ADFs. Therefore, we continue to give a detailed complexity analysis for acyclic ADFs in terms of all semantics and decision problems as introduced in Section 2.

Similarly to the previous results, one can decide the verification problem for grounded, complete, and preferred semantics in polynomial time for acyclic ADFs, by computing the unique preferred (unique complete) interpretation in polynomial time.

Corollary 3.6. *In acyclic ADFs one can verify whether a given interpretation is grounded, complete, or preferred in polynomial time.*

Notably absent in the preceding corollary is the admissible semantics. In fact, acyclicity does not help in verifying whether an interpretation is admissible (or conflict-free): e.g., asking whether it is admissible to assign a non-initial argument to true and its parents to undefined still requires to consider all completions.

Proposition 3.7. *In acyclic ADFs the problem of verifying whether a given interpretation is admissible is* coNP*-complete, and verifying whether a given interpretation is conflict-free is* NP*-complete.*

Proof. Membership follows from general ADFs. For hardness, we reduce from the problem of deciding whether a given Boolean formula ϕ, over vocabulary X, is satisfiable for showing hardness under conflict-free semantics, and a tautology for showing hardness under admissible semantics (see, e.g., [60]). Construct ADF $D = (X \cup \{f\}, L, C)$ with f a fresh argument not in X, $\varphi_x = \top$ for $x \in X$, and $\varphi_f = \phi$. It follows that D is acyclic and can be computed in polynomial time. We now claim that ϕ is satisfiable (a tautology) iff $I = \{x \mapsto \mathbf{u} \mid x \in X\} \cup \{f \mapsto \mathbf{t}\}$ is conflict-free (admissible) in D. Assume ϕ is satisfiable, then, by definition, I is conflict-free in D (only f is true, and φ_f is satisfiable by assumption). If ϕ is a tautology, then I is admissible ($\Gamma_D(I)(f) = \mathbf{t}$). Assume now that I is conflict-free in D. By definition, and since $I(f) = \mathbf{t}$, it holds that $\varphi_f[I] = \phi$ is satisfiable. If I is admissible in D, it holds that $\Gamma_D(I)(f) = \mathbf{t}$ iff $\varphi_f[I]$ is a tautology iff ϕ is a tautology. □

However, existence of non-trivial interpretations, for all semantics considered in this paper, is trivial. Consider an initial argument (leaf argument). This argument has a constant acceptance condition, which always is either tautological or unsatisfiable (and thus satisfiable or refutable). This means that there is an admissible and a conflict-free interpretation assigning this argument to true or false.

Proposition 3.8. *Existence of non-trivial interpretations, for all semantics considered in this paper, is trivial for acyclic ADFs.*

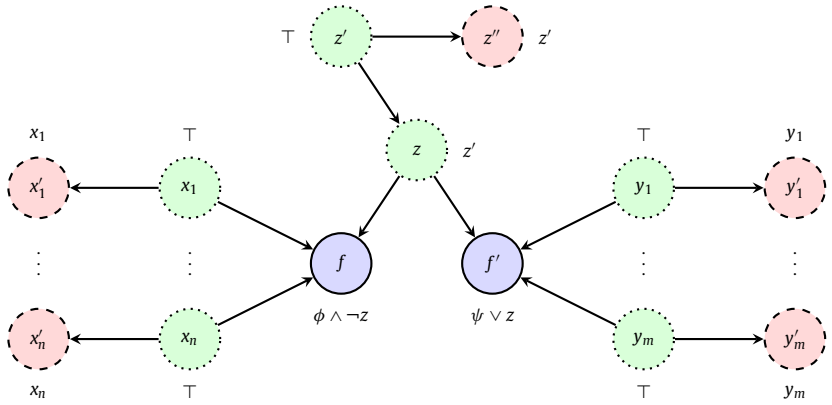

Fig. 4. Reduction for hardness proof of Proposition 3.10 and verifying whether interpretation is naive (interpretation to verify illustrated by solid blue are assigned true, dashed red are assigned false, and dotted green are assigned undefined). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Proof. Given an acyclic ADF $D = (A, L, C)$, if $A = \emptyset$, the problem is trivial. If $A \neq \emptyset$, then, since D is acyclic, there is a leaf a in the graph (A, L). It holds that φ_a does not contain variables (since $par_D(a) = \emptyset$). This implies that for any three-valued interpretation I we have $\varphi_a[I]$ is either tautological or unsatisfiable. In the former case, there is a conflict-free (admissible) interpretation of D that assigns true to a, in the latter case there is a conflict-free (admissible) interpretation of D that assigns false to a. In both cases there exists a non-trivial conflict-free (admissible) interpretation of D. If there is a non-trivial conflict-free (admissible) interpretation in an ADF, then there is also a non-trivial naive (complete, preferred) interpretation of that ADF. For grounded semantics, recall that the unique preferred interpretation is also grounded in any acyclic ADF. □

While reasoning under complete-based semantics is significantly easier in acyclic ADFs, it seems that acyclicity does not result in milder complexity for reasoning under conflict-free or naive semantics (except, as just shown, for existence of non-trivial interpretations). As an intuition why there is a difference between complete-based and naive reasoning, consider Example 3.2: in the unique preferred interpretation it holds that e is false, yet in one naive interpretation e is in fact true. This suggests that naive and, e.g., preferred semantics significantly diverge on acyclic ADFs.

Proposition 3.9. *In acyclic ADFs credulous acceptance under conflict-free and naive semantics is* NP*-complete.*

Proof. Membership follows from general ADFs. For hardness, consider the same reduction from the problem of deciding whether a given Boolean formula is satisfiable as in the proof of Proposition 3.7. There argument f is credulously accepted under conflict-free semantics (and thus also for naive semantics) iff ϕ is satisfiable (there is a conflict-free interpretation assigning f to true iff ϕ is satisfiable). □

Also verification under naive semantics remains as complex on acyclic ADFs as on general ADFs.

Proposition 3.10. *In acyclic ADFs verifying whether an interpretation is naive is DP-complete.*

Proof. Membership follows from general ADFs. For hardness, we reduce from the problem of verifying whether a given pair of Boolean formulas (ϕ, ψ), with vocabularies X and Y respectively, it holds that ϕ is satisfiable and ψ is unsatisfiable. For a set B define $B' = \{b' \mid b \in B\}$. Construct ADF $D = (A, L, C)$ with $A = X \cup X' \cup Y \cup Y' \cup \{z, z', z'', f, f'\}$ where $\{z, z', z'', f, f'\}$ are fresh arguments not in $X \cup X' \cup Y \cup Y'$. Define acceptance conditions as follows:

- $\varphi_x = \top$ for $x \in X$, $\varphi_{x'} = x$ for $x' \in X'$,
- $\varphi_y = \top$ for $y \in Y$, $\varphi_{y'} = y$ for $y' \in Y$,
- $\varphi_z = z'$, $\varphi_{z'} = \top$, $\varphi_{z''} = z'$,
- $\varphi_f = \phi \wedge \neg z$, and $\varphi_{f'} = \psi \vee z$.

We claim that (ϕ, ψ) is a yes instance of SAT-UNSAT iff I is naive in D with $I = \{x' \mapsto \mathbf{f} \mid x' \in X'\} \cup \{y' \mapsto \mathbf{f} \mid y' \in Y'\} \cup \{a \mapsto \mathbf{u} \mid a \in X \cup Y\} \cup \{f \mapsto \mathbf{t}, f' \mapsto \mathbf{t}, z \mapsto \mathbf{u}, z' \mapsto \mathbf{u}, z'' \mapsto \mathbf{f}\}$ (Fig. 4).

Assume ϕ is satisfiable and ψ is unsatisfiable. We first show that then I is conflict-free in D. It follows that $\varphi_{x'}[I] = x$ is refutable (similarly for y'). Further, $\varphi_f[I] = \phi \wedge \neg z$ is satisfiable since ϕ is satisfiable and $\varphi_{f'}[I] = \psi \vee z$ is satisfiable (independently of ψ). Finally $\varphi_{z''}[i] = z'$ is refutable. Thus I is conflict-free in D. Now we show that I is naive. Suppose

the contrary, i.e., there is an $I' \in cf(D)$ such that $I \leq_i I'$. The set of arguments that are undefined in I are $X \cup Y \cup \{z, z'\}$. It follows that some argument a in this set is assigned differently to $\mathbf{u}$ by I' (possibly more than one argument a; we focus on existence of one such argument). Assume $a \in X \cup Y$. Then, due to conflict-freeness, we have $I'(a) = \mathbf{t}$ (a's acceptance condition is $\top$). Then, $\varphi_{a'}[I'] = \top$, which implies that I' is not conflict-free (a' is assigned false, requiring a refutable $\varphi_{a'}[I']$). This means $a \in \{z, z'\}$. If $a = z'$, then $\varphi_{z''}[I'] = \top$ (since $I'(z') = \mathbf{t}$ due to $\varphi_{z'} = \top$). This contradicts that $I'(z'') = \mathbf{f}$ and I' being conflict-free. If $a = z$, then $I'(z) = \mathbf{t}$ or $I'(z) = \mathbf{f}$. In the former case it holds that $\varphi_f[I'] = \phi \wedge \neg\top$, which contradicts that I' is conflict-free and $I'(f) = \mathbf{t}$. In the latter case it holds that $\varphi_{f'}[I'] = \psi \vee \bot \equiv \psi$, which contradicts that I' is conflict-free and $I'(f') = \mathbf{t}$. Thus, I' cannot be conflict-free, and, in turn, I is naive.

Assume that ϕ is unsatisfiable or ψ is satisfiable. If ϕ is unsatisfiable, then I is not conflict-free, since $\varphi_f[I] = \phi \wedge \neg z$ which is unsatisfiable. If both ϕ and ψ are satisfiable, then we show that $I' = I|^z_{\mathbf{f}}$ is conflict-free in D (i.e., I' is equal to I, except for assigning z to false, and we have $I \leq_i I'$). For any $a \in X' \cup Y'$, the condition for conflict-freeness can be shown as above for the other direction. For the remaining arguments assigned differently to $\mathbf{u}$ by I', i.e., the arguments $\{f, f', z, z''\}$, we now show that, likewise, the condition of conflict-freeness holds. Formula $\varphi_f[I'] = \phi \wedge \top \equiv \phi$ is satisfiable and formula $\varphi_{f'}[I'] = \psi \vee \bot \equiv \psi$ is satisfiable. Finally $\varphi_z[I'] = z'$ is satisfiable. Thus I' is conflict-free and $I <_i I'$. This implies that I is not naive in D. □

Finally, we show coNP hardness for the skeptical acceptance problem under naive semantics. This concludes the analysis for acyclic ADFs. Note that with the exception of that last problem, we were able to show tight complexity bounds.

Proposition 3.11. *In acyclic ADFs it holds that skeptical acceptance under naive semantics is* coNP*-hard.*

Proof. Let ϕ be a Boolean formula over vocabulary X, and an instance of the UNSAT problem. Construct $D = \{X \cup \{f\}, L, C\}$ with f a fresh argument, and $\varphi_x = \top$ and $\varphi_f = \neg\phi$. It holds that ϕ is unsatisfiable iff φ_f is valid iff f is skeptically accepted under naive semantics. To see this, consider any conflict-free interpretation I of D in case ϕ is unsatisfiable. It follows that I assigns each $x \in X$ to either true or undefined. If $I(f) = \mathbf{u}$, then $I|^f_{\mathbf{t}}$ is also conflict-free: $\neg\phi[I|^f_{\mathbf{t}}]$ is satisfiable (in fact valid). This means, any naive interpretation assigns f to true in case ϕ is unsatisfiable. Assume ϕ is satisfiable. Then $\neg\phi$ is refutable. This implies that there is a conflict-free interpretation assigning f to false (and everything else, e.g., to undefined). This means there is a naive interpretation assigning f not to true. □

Summarizing, acyclicity of the underling graph structure does indeed support milder complexity of reasoning under reasoning tasks on ADFs, with the intuition that acyclic dependency graphs allow for a kind of "bottom-up" computation in polynomial time for several tasks (excluding, e.g., some tasks under naive semantics).

3.2. Bipartite ADFs

A well-studied graph class for AFs is that of bipartite AFs. A direct generalization of bipartite AFs (see, e.g., [36]) to ADFs is given next.

Definition 3.12. An ADF $D = (A, L, C)$ is bipartite if the set of arguments A can be partitioned into two sets A_1 and A_2 such that $A_1 \cap A_2 = \emptyset$ and $A = A_1 \cup A_2$, and there is no link $(a, b) \in L$ with both a and b in one set, i.e., $\nexists (a, b) \in L$ with $\{a, b\} \subseteq A_i$ for $i \in \{1, 2\}$.

That is, an ADF is bipartite if the underlying directed graph (A, L) is bipartite.

We show that bipartiteness does not pose a serious restriction on ADFs, since any ADF can be translated into a bipartite ADF with a strong correspondence between the respective complete interpretations. The main idea for the translation is to replace each link $l = (x, y)$ by two links (x, a_l) and (a_l, y) with a newly introduced argument a_l. The translation is defined next formally.

Definition 3.13. Let $D = (A, L, C)$ be an ADF. Define $bip(D) = D' = (A', L', C')$ with

- $A' = A \cup \{a_l \mid l \in L\}$,
- $L' = \{(x, a_l), (a_l, y) \mid l = (x, y), l \in L\}$, and
- $C' = \{\varphi'_x \mid \varphi_x \in C\}$ with

$$\varphi'_x = \begin{cases} \varphi_x[y_1 \mapsto a_{(y_1,x)}, \ldots, y_n \mapsto a_{(y_n,x)}] \text{ with } par_D(x) = \{y_1, \ldots, y_n\} & \text{if } x \in A \\ z & \text{if } x = a_{(z,y)} \in A' \setminus A. \end{cases}$$

Example 3.14. In Fig. 5 an example translation is shown for a given ADF D to ADF D'. In this figure, the partition of the arguments is indicated by blue solid circles and red dashed circles. The complete interpretations are as follows.

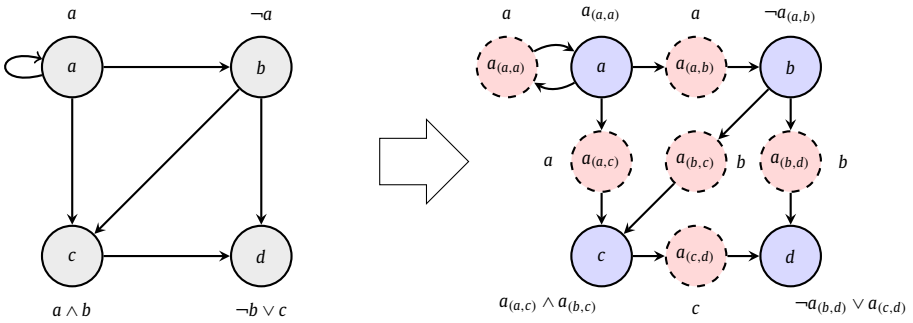

Fig. 5. Adding arguments to an ADF to achieve a corresponding bipartite ADF.

$$
\begin{aligned}
com(D) = \{&\{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\},\\
&\{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\}\\
&\{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}, c \mapsto \mathbf{f}, d \mapsto \mathbf{f}\}\}\\
com(D') = \{&\{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u},\\
&a_{(a,a)} \mapsto \mathbf{u}, a_{(a,b)} \mapsto \mathbf{u}, a_{(a,c)} \mapsto \mathbf{u}, a_{(b,c)} \mapsto \mathbf{u}, a_{(b,d)} \mapsto \mathbf{u}, a_{(c,d)} \mapsto \mathbf{u}\},\\
&\{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t},\\
&a_{(a,a)} \mapsto \mathbf{t}, a_{(a,b)} \mapsto \mathbf{t}, a_{(a,c)} \mapsto \mathbf{t}, a_{(b,c)} \mapsto \mathbf{f}, a_{(b,d)} \mapsto \mathbf{f}, a_{(c,d)} \mapsto \mathbf{f}\},\\
&\{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}, c \mapsto \mathbf{f}, d \mapsto \mathbf{f},\\
&a_{(a,a)} \mapsto \mathbf{f}, a_{(a,b)} \mapsto \mathbf{f}, a_{(a,c)} \mapsto \mathbf{f}, a_{(b,c)} \mapsto \mathbf{t}, a_{(b,d)} \mapsto \mathbf{t}, a_{(c,d)} \mapsto \mathbf{f}\}\}
\end{aligned}
$$

Before diving into the formal statement for the translation, we show an auxiliary lemma stating a simple observation: if an acceptance condition of an argument a is $\varphi_a = b$, then in any complete interpretation I it holds that both arguments are assigned the same value, i.e., $I(a) = I(b)$. Intuitively, this simple acceptance condition, utilized for the translation *bip*, acts as a propagator.

Lemma 3.15. *Let $D = (A, L, C)$ be an ADF with $a \in A$ and $\varphi_a = b$. It holds that $I \in com(D)$ implies $I(a) = I(b)$.*

Proof. Assume interpretation I is complete in D. Since I is complete we can infer $I(a) = \mathbf{t}$ iff $\Gamma_D(I)(a) = \mathbf{t}$ iff $\varphi_a[I]$ is tautological iff $b[I]$ is tautological iff $I(b) = \mathbf{t}$. Analogously, one can infer $I(a) = \mathbf{f}$ iff $I(b) = \mathbf{f}$. Further, $I(a) = \mathbf{u}$ iff $\Gamma_D(I)(a) = \mathbf{u}$ iff $\varphi_a[I]$ is satisfiable and refutable iff $b[I]$ is satisfiable and refutable iff $I(b) = \mathbf{u}$. □

In the following lemma we prove that the translation *bip* transforms any arbitrary ADF to a bipartite ADF with a direct correspondence for the complete semantics.

Lemma 3.16. *Let $D = (A, L, C)$ be an ADF. For $D' = bip(D)$ it holds that*

1. *D' is a bipartite ADF,*
2. *$I \in com(D)$ implies $\exists J \in com(D')$ with $I(x) = J(x)$ for each $x \in A$, and*
3. *$J \in com(D')$ implies $\exists I \in com(D)$ with $I(x) = J(x)$ for each $x \in A$.*

Proof. We first prove that $D' = (A', L', C')$ is an ADF. That is, we first show that if a variable occurs in an acceptance conditions a corresponding link is present (otherwise, a link would be missing). Let y be a variable in φ_x with $x \in A'$. If $x = a_{(u,v)} \in A' \setminus A$, then $y = u$ and there must be $(u, v) \in L$ (original ADF). This means a link $(u, a_{(u,v)})$ is defined in the construction. If $x \in A$ (not a "new" argument introduced in the construction), $y = a_{(u,x)}$ for some $u \in A$. Since u occurs in the acceptance condition of φ_x in the unmodified ADF D (by construction), it holds that $(u, x) \in L$, implying that $(u, a_{(u,x)}) \in L'$.

Next, in order to show that D' is bipartite, we claim that $A_1 = A$ and $A_2 = A' \setminus A$ is a partition of A' such that there is no link $(a, b) \in L'$ with $\{a, b\} \subseteq A_i$ for $i \in \{1, 2\}$. By construction, each link $l \in L'$ is either of form (x, a_l) or (a_l, x) for $x \in A$ and $a_l \in A' \setminus A$. This implies that (A', L') is a directed bipartite graph, and that D' is bipartite.

Assume that $I \in com(D)$. Extend I to J defined over A' by $J(x) = I(x)$ if $x \in A$ and $J(a_{(y,x)}) = I(y)$ for $a_l \in A' \setminus A$. We claim that $J \in com(D')$. Towards the result we show $\Gamma_{D'}(J)(x) = J(x)$ for $x \in A'$. We proceed by case distinction: (i) $x \in A$ and (ii) $x \in A' \setminus A$. Consider the first case (i). It holds that $\varphi_x[I] = \varphi'_x[J]$. To see this, consider any variable of

y of φ_x: $y[I] = (y[y \mapsto a_{(y,x)}])[a_{(y,x)} \mapsto I(y)] = a_{(y,x)}[J]$ since $J(a_{(y,x)}) = J(y) = I(y)$. Since any y is replaced by $a_{(y,x)}$ by construction, it holds that $\varphi_x[I] = \varphi'_x[J]$. This implies that $\Gamma_{D'}(J)(x) = J(x)$ for $x \in A$. Consider case (ii): by similar reasoning as in Lemma 3.15 it holds that J is complete.

Assume that $J \in com(D)$. By Lemma 3.15 it holds that $J(y) = J(a_{(y,x)})$ for all $y \in A$ and $a_{(y,x)} \in A' \setminus A$. Let I be a three-valued interpretation over A defined by $I(x) = J(x)$ for $x \in A$. We claim that $I \in com(D)$. By similar reasoning as above, we conclude that $\varphi_x[I] = \varphi'_x[J]$ for $x \in A$, since all parents of x in D are assigned the same value by I as by J in D'. □

The previous lemma immediately shows that the complexity of the decision problems we consider cannot be easier than in the case of general ADFs. Otherwise, using the previous lemma, we would be able to transform any ADF into the easier class of bipartite ADFs in polynomial time (note that the transformation of Definition 3.13 is indeed efficiently computable) and solve the problem in that class; a contradiction. More specifically, given the relations 2 and 3 in Lemma 3.16, it holds, for instance that, given an arbitrary ADF $D = (A, L, C)$ an argument $a \in A$ is credulously (resp. skeptically) accepted in D w.r.t. complete semantics iff a is credulously (resp. skeptically) accepted in $bip(D)$ w.r.t. complete semantics.

Hence, compared to the class of acyclic ADFs, bipartite ADFs do not show the desired drop in complexity for complete (and likewise, preferred or grounded) semantics. We thus do not further study the complexity of bipartite ADFs and proceed with another syntactic subclass.

3.3. Symmetric ADFs

Another well known subclass of AFs is that of symmetric AFs, where the underlying graph structure is symmetric and irreflexive. We adapt this definition in terms of ADFs.

Definition 3.17. An ADF $D = (A, L, C)$ is symmetric if the binary relation L is symmetric and irreflexive.

Similarly to bipartite ADFs, one can translate any ADF to a symmetric one with a correspondence on the complete semantics. We define that translation next. The translation takes care of two issues: first, arguments that are self-reflexive need particular treatment, i.e., we copy each such argument and provide a simple symmetric link between the original and the copied argument. For making the remaining relations symmetric, we just add a redundant subformula to each acceptance conditions in order to increase the set of parent nodes without changing the semantics of the acceptance conditions at all.

Definition 3.18. Let $D = (A, L, C)$ be an ADF. Define $symm(D) = D' = (A', L', C')$ with

- $A' = A \cup \{x' \mid (x, x) \in L\}$,
- $L' = (L \setminus \{(x, x) \mid x \in A\}) \cup \{(x, x'), (x', x) \mid x \in A, x' \in A' \setminus A\} \cup \{(a, b) \mid (b, a) \in L, a \neq b\}$, and
- $C' = \{\varphi'_x \mid \varphi_x \in C\}$ with

$$\varphi'_x = \begin{cases} \varphi_x \wedge (\top \vee \bigvee_{(x,y)\in L,(y,x)\notin L} y) & \text{if } x \in A, \nexists (x, x) \in L \\ \varphi_x[x \mapsto x'] \wedge (\top \vee \bigvee_{(x,y)\in L,(y,x)\notin L} y) & \text{if } x \in A, \exists (x, x) \in L \\ y & \text{if } x \in A' \setminus A \text{ and } x = y' \end{cases}$$

Example 3.19. Consider the ADF D shown in Fig. 6 on the left side. The translation $symm(D) = D'$ is shown on the right side of that figure. The changed acceptance conditions are

- $\varphi'_a = a' \wedge (\top \vee b \vee c)$,
- $\varphi'_b = \neg a \wedge (\top \vee c \vee d)$,
- $\varphi'_c = a \wedge b \wedge (\top \vee d)$, and
- $\varphi'_d = (\neg b \vee c) \wedge (\top) = \varphi_d \wedge (\top)$.

The added "redundant" links (e.g., from b to a) do not affect the complete semantics. These links are also redundant in a formally defined sense: they are both attacking and supporting according to the definition of bipolar ADFs, see subsequent section for a formal definition. The complete interpretations align with the exception that D' has one additional argument, that receives the same truth value as a (due to Lemma 3.15).

We prove the semantical correspondence induced by the translation next.

Lemma 3.20. *Let $D = (A, L, C)$ be an ADF. For $D' = symm(D)$ it holds that*

1. *D' is a symmetric ADF,*

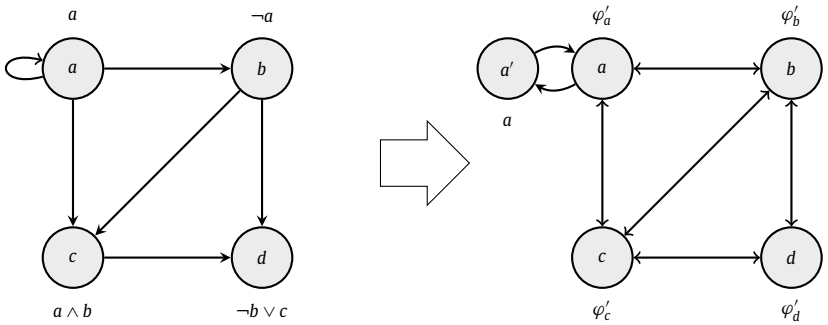

Fig. 6. Adding redundant links to an ADF to achieve a corresponding symmetric ADF.

2. *$I \in com(D)$ implies there is a $J \in com(D')$ with $I(x) = J(x)$ for each $x \in A$, and*
3. *$J \in com(D')$ implies there is a $I \in com(D)$ with $I(x) = J(x)$ for each $x \in A$.*

Proof. It holds that L' contains all links from L, except for the self-loops, for each pair of arguments a, b with $(a, b) \in L$ also $(b, a) \in L'$, and for each new argument $x' \in A' \setminus A$ we have a symmetric link from x to x'. This implies that D' is symmetric (no self-loops and only mutual links).

Towards the other two items, note that if $a \in A$ and $(a, a) \notin L$, we have $\varphi'_a[I] \equiv \varphi_a[I]$ for any two-valued interpretation I, since $\varphi'_a[I] = \varphi_a[I] \wedge (\top \vee \phi[I])$ and $\phi[I]$ contains no variables. This implies that $\Gamma_D(J)(a) = \Gamma_{D'}(J)(a)$ for any $a \in A$ and any three-valued interpretation J, and, moreover, $\Gamma_D(J)(a) = \Gamma_{D'}(J')(a)$ for any two three-valued interpretations J and J' which assign the same values to parents of a.

Assume $I \in com(D)$. We show that $J \in com(D')$ with $I(x) = J(x)$ for $x \in A$ and $J(x') = J(x)$ for $x' \in A' \setminus A$ (i.e., all arguments in D are assigned the same value, and arguments x' introduced due to self-loops of arguments x are assigned the same value as x). By the observation above, for any $x \in A$ with $(x, x) \notin L$ it follows that $I(x) = \Gamma_D(I)(x) = \Gamma_{D'}(J)(x) = J(x)$ (the parents are assigned the same values and the acceptance conditions evaluate to the same value). If $x' \in A' \setminus A$, then $\varphi'_{x'} = x$ and since $J(x') = J(x)$ also $\Gamma_{D'}(J)(x') = J(x')$ holds. The last case to consider is $x \in A$ and there is a self-loop on x in D, i.e., $(x, x) \in L$. In this case, the variable x in φ_x is replaced by x' in φ'_x. Since $J(x) = J(x')$, by construction, it follows that J is complete in D'.

Assume $J \in com(D')$. By Lemma 3.15 if $x \in A$ and $(x, x) \in L$, it holds that $J(x) = J(x')$ (the arguments introduced due to self-loops have the same value as the former self-looping argument in complete interpretations). Consider three-valued interpretation I such that $I(x) = J(x)$ for $x \in A$. If $\nexists(x, x) \in L$, by the observation above, it follows that $I(x) = \Gamma_D(I)(x) = \Gamma_{D'}(J) = J(x)$. Similar as in the other direction, if there is a link $(x, x) \in L$, then $I(x) = \Gamma_D(I)(x) = \Gamma_{D'}(J) = J(x)$ follows, as well, since the corresponding acceptance conditions are evaluated in the same way, except that the variable x is replaced by x' in φ'_x, but both are assigned the same truth value. □

As in the case of bipartite ADFs, the previous lemma (together with the fact that the translation in Definition 3.18 can be computed in polynomial time) shows that symmetric ADFs do not yield a milder complexity for complete-based semantics. Hence, they are not suited for our purposes.

3.4. Bipolar ADFs

A prominent fragment of ADFs are so-called bipolar ADFs. In general, the term bipolarity, within formal argumentation, typically refers to two contrasting positions or relations between arguments. In the case of bipolarity of ADFs, the focus is on attacking and supporting relations between arguments, which are represented by bipolar acceptance conditions. Bipolar ADFs were first introduced by [18], and have been investigated to quite some depth later on regarding complexity issues [18, 69], relations to other argumentation formalisms [61], number of bipolar acceptance conditions [7], importing ideas of bipolarity of ADFs to logic programming [4], studying expressivity of bipolar ADFs [51,66], and bipolarity was used in investigations of admissibility [62]. More generally, bipolarity in formal argumentation has, likewise, received attention from the community [21].

Bipolarity in ADFs is defined on the links (a, b) between arguments, by considering how (non-)acceptance of a influences (non-)acceptability of b, which is specified in the acceptance conditions. We follow the definition originally from [18,14]. A link between two arguments is classified according to four classes and can be

- attacking,
- supporting,
- dependent, and
- redundant.

These four classes are not distinct. In particular, the last two classes are defined via the first two. Towards the definition, we make use of an auxiliary concept: we denote the update of an interpretation I with truth value $\mathbf{x} \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ for argument b by $I|^b_{\mathbf{x}}$, i.e., $I|^b_{\mathbf{x}}(b) = \mathbf{x}$ and $I|^b_{\mathbf{x}}(a) = I(a)$ for $a \neq b$. Using this auxiliary concept we define attacking and supporting links, as follows.

Definition 3.21. Let $D = (A, L, C)$ be an ADF, and $(b, a) \in L$ a link. This link is called

- *supporting* (in D) if for every two-valued interpretation I, $I(\varphi_a) = \mathbf{t}$ implies $I|^b_{\mathbf{t}}(\varphi_a) = \mathbf{t}$; or
- *attacking* (in D) if for every two-valued interpretation I, $I(\varphi_a) = \mathbf{f}$ implies $I|^b_{\mathbf{t}}(\varphi_a) = \mathbf{f}$.

A link is said to be dependent if the link is neither attacking nor supporting, and a link is redundant if the link is both attacking and supporting.

For an ADF D, we denote the corresponding sets of links as L^-_D, for attacking links, L^+_D, for supporting links, and $L^?_D$ for dependent links.

Example 3.22. Consider acceptance condition $\varphi_c = a \rightarrow b \equiv \neg a \vee b$. Let us look at the links (a, c) and (b, c). With respect to φ_c's vocabulary of $\{a, b\}$, there are four relevant two-valued interpretations: assigning true or false to a and b, in each combination. There are three satisfying two-valued interpretations, and one that does not satisfy the formula, if a is assigned true and b is assigned to false. This interpretation witnesses that (a, c) is attacking: assigning a to true preserves that interpretation's property of not satisfying φ_c (in fact updating a to true yields the same interpretation). Likewise, this interpretation witnesses that (b, c) is not attacking: updating b to true results in a model of φ_c, violating the condition of attacking links. By analogous arguments, one can show that (a, c) is not supporting and (b, c) is supporting. This means that both links are neither dependent nor redundant.

As an example for a link that is both attacking and supporting, consider $\varphi_b = a \vee \neg a \equiv \top$, a tautological acceptance condition. Then (a, b) is both attacking and supporting, since the corresponding conditions are trivially satisfied. An example for a link where neither the condition of attacking nor supporting is satisfied is $\varphi_c = (a \rightarrow \neg b) \wedge (\neg b \rightarrow a)$. One can find both violations of (a, c) (and due to symmetry also for (b, c)) being attacking and for it being supporting.

As can be seen from the example, a link can be both attacking and supporting. In such a case a link is called redundant, intuitively specifying no effect of that link on acceptability. If a link is neither attacking nor supporting, such a link is called dependent, with the intuition that dependent on (non-)acceptance of other arguments the influence can be either positive or negative.

Definition 3.23. Let D be an ADF. If each link is attacking or supporting then D is a bipolar ADF.

It has been shown that reasoning in bipolar ADFs is easier [69], when for each link its type is known. The results are shown in Table 4. Towards an intuition (which will be useful for the next section on distance to bipolar ADFs), recall the following alternative way for defining the characteristic function. Given a three-valued interpretation I and an argument a within an ADF $D = (S, L, C)$, applying $\Gamma_D(I)$ yields an assignment of true for a iff $\varphi_a[I]$ is tautological. Alternatively, a is assigned true by $\Gamma_D(I)$ iff all two-valued completions of I satisfy φ_a. Recall that a completion J of I is such that $I \leq_i J$ and J is two valued. That is, a completion J may assign true or false to arguments assigned undefined by I, but must assign the same value as I whenever an argument is already assigned to true or false.

For general ADFs, to check whether $\Gamma_D(I)(a) = \mathbf{t}$ (an ingredient for checking admissibility of I) amounts to testing whether $\varphi_a[I]$ is a tautology (or checking all completions). In bipolar ADFs, it is sufficient to consider a single completion to check whether this completion satisfies φ_a or not, in order to decide whether $\Gamma_D(I)(a) = \mathbf{t}$ holds.

Example 3.24. Consider three-valued interpretation $I = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}\}$ that assigns a to true and b and c to undefined. Assume that φ_a has variables b and c, and that link (b, a) is attacking and (c, a) is supporting. Let us approach the question whether $I(a) \leq_i \Gamma_D(I)(a)$ holds. The last statements holds iff $\varphi_a[I]$ is tautological iff all two-valued completions of I satisfy φ_a. Consider the specific two-valued completion $J = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{f}\}$, where we assigned the attacking argument to true and the supporting argument to false. Intuitively, this is the "worst case" for acceptability of a. More formally, whenever there is doubt that a should be acceptable, w.r.t. three-valued interpretation I, then there is a two-valued completion J' of I that does not satisfy φ_a. Say $J' = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{t}\}$ (dual to J on b and c) does not satisfy φ_a. By assumption, (b, a) is attacking. This implies that if $J'(\varphi_a) = \mathbf{f}$ we have $J'|^b_{\mathbf{t}}(\varphi_a) = \mathbf{f}$, i.e., when assigning b to true we still get non-satisfiability of φ_a. Further, by the converse of supporting links we have $W|^c_{\mathbf{t}}(\varphi_a) = \mathbf{f}$ implies $W(\varphi_a) = \mathbf{f}$, in particular for $W = J$. That is, assigning c to false, from true, preserves non satisfiability. Taken together, if J' does not satisfy φ_a, then also J does not satisfy φ_a. In this case, J is the unique completion that is sufficient to check whether $\Gamma_D(I)(a) = \mathbf{t}$ holds.

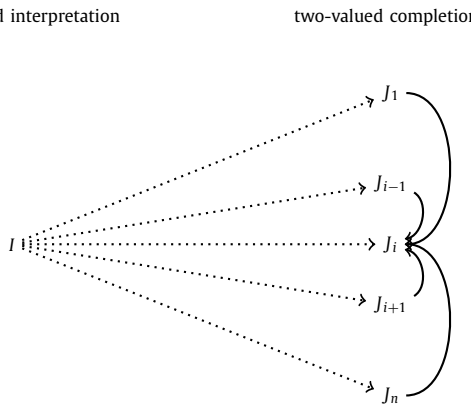

Fig. 7. Conceptual view on bipolar acceptance conditions. Consider the case when checking whether a three-valued interpretation I and bipolar acceptance condition φ_a satisfy $\mathbf{t} = I(a) \leq_i \Gamma_D(I)(a)$. Out of all two-valued completions J of I (completions denoted by dashed arrows) exactly one completion J_i assigns all undefined arguments to true when they attack a and to false when they support a. It holds that J is not a model of φ_a implies that J_i is not a model of φ_a (this implication is represented by solid arrows).

As exemplified just now, checking admissibility (specifically checking whether $\Gamma_D(I)(a) = \mathbf{t}$ holds) can be reduced to considering one single two-valued completion. In Fig. 7 this behavior is illustrated: while there are exponentially many completions, in the worst case, for a given three-valued interpretation I, in bipolar ADFs one can restrict to checking only one two-valued interpretation J_i. Non-satisfiability of any two-valued completion J is preserved by J_i (solid arrows in the figure).

3.5. Concise ADFs

The next subclass we consider are so-called concise ADFs. Most of the subclasses up to now have a syntactic nature, i.e., are defined on the syntactic graph structure: acyclicity, bipartiteness, or symmetry. Bipolar ADFs are defined differently, namely on the semantics of the acceptance conditions. Nevertheless, they are not defined "purely" on the semantics of ADFs (rather on semantics of Boolean formulas). Concise ADFs, on the other hand, are defined solely based on semantic notions. For a given ADF semantics σ, an ADF D is σ-concise if there is exactly one σ-interpretation in D. An analogous class was already considered on AFs by [34], and found to be computationally beneficial.

Definition 3.25. An ADF D is concise for semantics σ (or σ-concise) if $|\sigma(D)| = 1$.

Example 3.26. A simple ADF that is not *com*-concise is one representing a mutual attack between two arguments: $\varphi_a = \neg b$ and $\varphi_b = \neg a$. In the ADF $D = (\{a, b\}, L, C)$, we have three complete interpretations: the trivial interpretation, one assigning a to true and b to false, and one assigning a to false and b to true.

The ADF $D' = (\{a, b\}, L', C')$ with $\varphi'_a = \neg a \wedge \neg b$ and $\varphi'_b = \neg a$ is not *com*-concise, either: $com(D') = \{\{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}\}, \{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}\}\}$. However, D' is *prf*-concise, since $\{a \mapsto \mathbf{f}, b \mapsto \mathbf{t}\}$ is the only preferred interpretation of D'.

Immediate from the definitions, we have a straightforward result: for *cf*-concise (resp. *adm*-concise) ADFs there is only one interpretation that is conflict-free (resp. admissible), namely $I_{\mathbf{u}}$ (since $I_{\mathbf{u}}$ is always conflict-free and admissible). Hence, the decision problems $Cred_\sigma$, $Skept_\sigma$, $Exists_\sigma$ for $\sigma \in \{cf, adm\}$ are always answered negatively, since no argument can be in a σ-interpretation for *cf*-concise (*adm*-concise) ADFs. The problem Ver_σ is positive only for $I_{\mathbf{u}}$ (by definition no other σ-interpretations exist).

Proposition 3.27. *For $\sigma \in \{cf, adm\}$, it holds that the reasoning tasks of credulous and skeptical acceptance, as well as existence of non-trivial interpretations and verification, are trivial for semantics σ in σ-concise ADFs.*

Since grounded semantics is a unique-status semantics it holds that any ADF is *grd*-concise. This means that complexity of reasoning under grounded semantics for general ADFs is the same as for *grd*-concise ADFs.

The grounded interpretation is the least complete interpretation, and, thus, reasoning under grounded and complete semantics coincides for *com*-concise ADFs. Moreover, complexity of reasoning under complete semantics in *com*-concise ADFs has the same complexity as reasoning under grounded semantics in general ADFs. While seemingly straightforward, a proof is nevertheless required, since it is not immediate that reasoning under grounded semantics has the "full" complexity (i.e., same as for general ADFs) also in the case of *com*-concise ADFs. Towards the formal results, we introduce a reduction and show some of the properties of the constructed ADF.

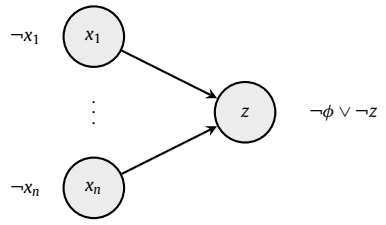

Fig. 8. Graphical representation of reduction 3.28.

Reduction 3.28. Let ϕ be a formula over vocabulary X. Define the ADF $D_\phi = (A, L, C)$ such that $A = X \cup \{z\}$ with z a fresh argument, $\varphi_x = \neg x$ for $x \in X$, and $\varphi_z = \neg\phi \vee \neg z$.

That is, all variables of the Boolean formula are translated to self-attacking arguments, and the formula to a distinguished argument that is also attacking itself, however differently than the others. We illustrate the reduction in Fig. 8. It is immediate that in any complete interpretation of this ADF all arguments $x \in X$ are assigned undefined. Further, whenever an argument z has an acceptance condition of the form $\psi \vee \neg z$, for any ψ (in our reduction $\psi = \neg\phi$), it holds that no admissible interpretation assigns false to z. Assigning false to z violates admissibility: $\mathbf{f} = I(z) \not\leq_i \Gamma_D(I)(z) = \mathbf{t}$, since $\psi \vee \neg z[I] \equiv \psi \vee \neg\bot \equiv \psi \vee \top \equiv \top$. In turn, there are only two candidates for complete interpretations, which are in a direct correspondence to satisfiability of ϕ, shown formally next.

Lemma 3.29. *Let ϕ be a formula over vocabulary X and D_ϕ the ADF obtained by Reduction 3.28. It holds that*

1. *If ϕ is unsatisfiable then $com(D_\phi) = \{\{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}\}$.*
2. *If ϕ is satisfiable then $com(D_\phi) = \{I_\mathbf{u}\}$.*

Proof. First observe that for each $x \in X$ it holds that $I(x) = \mathbf{u}$ in every $I \in com(D_\phi)$. (1) If ϕ is unsatisfiable, then $\varphi_z \equiv \top$. Hence $\Gamma_{D_\phi}(I_\mathbf{u}) = J = \{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}$. Moreover $\Gamma_{D_\phi}(J) = J$. Hence J is the grounded interpretation and, by the previous observation, the only complete interpretation. (2) Assume ϕ is not unsatisfiable. For $J = \{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}$ we get $\varphi_z[J] = \neg\phi \vee \neg\top \equiv \neg\phi \vee \bot$. Since ϕ is not unsatisfiable we have $\neg\phi \not\equiv \top$, hence $\Gamma_{D_\phi}(J) \neq J$ and $J \notin com(D_\phi)$. For $J' = \{z \mapsto \mathbf{f}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}$ we get $\varphi_z[J'] = \neg\phi \vee \neg\bot \equiv \top$. Hence $\Gamma_{D_\phi}(J') \neq J$ and $J' \notin com(D_\phi)$. Finally, for $I_\mathbf{u}$, we get $\varphi_z[I_\mathbf{u}] = \neg\phi \vee \neg z$. Since ϕ is satisfiable, $\neg\phi$ is not a tautology. Hence $\varphi_z[I_\mathbf{u}]$ is neither a tautology nor unsatisfiable. Therefore $\Gamma_{D_\phi}(I_\mathbf{u}) = I_\mathbf{u}$. By the initial observation that $I(x) = \mathbf{u}$ in every $I \in com(D_\phi)$, we conclude that $com(D_\phi) = \{I_\mathbf{u}\}$. □

Using the reduction and lemma, we prove the complexity of reasoning under complete semantics in *com*-concise ADFs next.

Proposition 3.30. *In com-concise ADFs, the acceptance tasks of credulous and skeptical reasoning, as well as existence of non-trivial interpretations, are* coNP*-complete problems under complete semantics.*

Proof. Membership results for all three tasks follows from general ADFs (i.e., $Cred_{com}$, $Skept_{com}$, and $Exists_{com}$ are in coNP, since the task coincides with the corresponding question for grounded semantics for *com*-concise ADFs, see also Table 3).

For hardness we give a reduction from UNSAT via Reduction 3.28. That is, given an arbitrary instance of UNSAT, i.e., a Boolean formula ϕ, construct D_ϕ via the reduction presented above. By inspecting the construction it is immediate that this reduction is feasible in polynomial time. By Lemma 3.29 it holds that if ϕ is satisfiable only the trivial interpretation is complete, otherwise, if ϕ is unsatisfiable only the interpretation assigning the distinguished argument z to true and all other arguments to undefined is complete. Thus, the resulting ADF is *com*-concise. By Lemma 3.29 it follows that in the framework D_ϕ it holds that z is credulously (skeptically) accepted under complete semantics iff there is a non-trivial complete interpretation iff ϕ is unsatisfiable. □

From the complexity landscape of complete semantics in *com*-concise ADFs the verification task is missing in the preceding proposition. The complexity of this task also coincides with verification complexity of complete (and grounded semantics) in general ADFs, however the hardness proof needs some adaptation.

Proposition 3.31. *In com-concise ADFs, verifying whether a three-valued interpretation is complete is DP-complete.*

Proof. Membership follows from general ADFs. We show hardness by a reduction from SAT-UNSAT, i.e., the problem of deciding whether from a pair of Boolean formula the first is satisfiable and the second is unsatisfiable. Given a pair of propositional formulas (ϕ, ψ) over vocabularies X_1 and X_2, respectively, with $X_1 \cap X_2 = \emptyset$, let $D_{(\phi,\psi)} = (A, L, C)$ with $A = X_1 \cup X_2 \cup \{y, z\}$ s.t. y and z are fresh arguments, and

- $\varphi_x = \neg x$ for $x \in X_1 \cup X_2$,
- $\varphi_y = \phi \wedge \neg y$, and
- $\varphi_z = \psi \wedge \neg z$.

Further, define $I = \{x \mapsto \mathbf{u} \mid x \in X_1 \cup X_2\} \cup \{y \mapsto \mathbf{u}, z \mapsto \mathbf{f}\}$. We have to show that (1) ϕ is satisfiable and ψ is unsatisfiable iff $I \in com(D_{(\phi,\psi)})$; and (2) $D_{(\phi,\psi)}$ is *com*-concise. First observe that, for each $x \in X_1 \cup X_2$, it must hold that $J(x) = \mathbf{u}$ in every $J \in com(D_{(\phi,\psi)})$. Assume ϕ is satisfiable and let J be an interpretation such that $J(x) = \mathbf{u}$ for each $x \in X_1 \cup X_2$. If $J(y) = \mathbf{t}$ then $\varphi_y[J] = \phi \wedge \neg\top \equiv \bot \not\equiv \top$, hence $J \notin com(D_{(\phi,\psi)}))$. If $J(y) = \mathbf{f}$ then $\varphi_y[J] = \phi \wedge \neg\bot \equiv \phi \not\equiv \bot$, hence $J \notin com(D_{(\phi,\psi)}))$. Hence $J(y) = \mathbf{u}$ in every complete interpretation of $D_{(\phi,\psi)}$. On the other hand assume ϕ is unsatisfiable and again let J be an interpretation such that $J(x) = \mathbf{u}$ for each $x \in X_1 \cup X_2$. It holds that $\varphi_y[J] \equiv \bot$, hence for $J \in com(D_{(\phi,\psi)})$ it must be that $J(y) = \mathbf{f}$. Likewise, we get that, for every $J \in com(D_{(\phi,\psi)})$, it must hold that $J(z) = \mathbf{u}$ if ψ is satisfiable and $J(z) = \mathbf{f}$ if ψ is unsatisfiable. Hence we get a single complete interpretation in any case, showing (2). Moreover it holds that (1) $I \in com(D_{(\phi,\psi)})$ iff ϕ is satisfiable and ψ is unsatisfiable. □

Complexity of reasoning under naive semantics, except for skeptical acceptance under this semantics, does not deviate from general ADFs. The hardness proof uses a variant of Reduction 3.28 from above.

Proposition 3.32. *In nai-concise ADFs, the reasoning tasks of credulous and skeptical acceptance, as well existence of non-trivial interpretations, under naive semantics, are all* NP*-complete problems.*

Proof. Given a formula ϕ over X, let $D_\phi = (X \cup \{z\}, L, C)$ with z a fresh argument, $\varphi_x = \neg x$ for $x \in X$, and $\varphi_z = \phi \vee \neg z$. For any $I \in cf(D_\phi)$ we have $I(x) = \mathbf{u}$. We have $I(z) = \mathbf{t}$ for some $I \in cf(D_\phi)$ iff $\varphi_z[I]$ is satisfiable iff ϕ is satisfiable. Finally, $I(z) \neq \mathbf{f}$ in every $I \in cf(D_\phi)$, as $\varphi_z[I]$ is then a tautology. Hence, $cf(D_\phi) = \{I_\mathbf{u}, \{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}\}$ if ϕ is satisfiable and $cf(D_\phi) = \{I_\mathbf{u}\}$ otherwise. □

Similarly as for complete semantics, we show verification complexity of naive semantics in *nai*-concise ADFs via a separate proof that adapts the previous hardness proof of Proposition 3.31.

Proposition 3.33. *In nai-concise ADFs, it holds that verifying whether an interpretation is naive is DP-complete.*

Proof. Membership in DP follows from general ADFs. It remains to show that Ver_{nai} is DP-hard. We show hardness by a reduction from SAT-UNSAT. Given a pair of propositional formulas (ϕ, ψ) over vocabularies X_1 and X_2, respectively, with $X_1 \cap X_2 = \emptyset$, let $D_{(\phi,\psi)} = (A, L, C)$ with $A = X_1 \cup X_2 \cup \{y, z\}$ s.t. y and z are fresh arguments not in X_1 and X_2,

- $\varphi_x = \neg x$ for $x \in X_1 \cup X_2$,
- $\varphi_y = \phi \vee \neg y$, and
- $\varphi_z = \psi \vee \neg z$.

We set $I = \{x \mapsto \mathbf{u} \mid x \in X_1 \cup X_2\} \cup \{y \mapsto \mathbf{t}, z \mapsto \mathbf{u}\}$. We have to show that

(1) ϕ is satisfiable and ψ is unsatisfiable iff $I \in nai(D_{(\phi,\psi)})$, and
(2) $D_{(\phi,\psi)}$ is *nai*-concise.

First observe that, for each $x \in X_1 \cup X_2$, it must hold that $J(x) = \mathbf{u}$ in every $J \in nai(D_{(\phi,\psi)})$. Assume that ϕ is satisfiable. Further let J be an interpretation with $J(y) = \mathbf{f}$. Then $\varphi_y[J] = \phi \vee \neg\bot \equiv \top$ and therefore $J \notin cf(D_{(\phi,\psi)})$. On the other hand, φ_y is clearly satisfiable, hence y is $\mathbf{t}$ in every naive interpretation of $D_{(\phi,\psi)}$. Likewise, if ψ is satisfiable, z is $\mathbf{t}$ in every naive interpretation of $D_{(\phi,\psi)}$. Assume that ψ is unsatisfiable. Then $\varphi_z \equiv \neg z$, hence z is $\mathbf{u}$ in every naive interpretation of $D_{(\phi,\psi)}$. We therefore get the following sets of naive interpretations, where $J(x) = \mathbf{t}$ holds for all J and all $x \in X$:

ϕ	ψ	$nai(D_{(\phi,\psi)})$
SAT	UNSAT	$\{I\}$
SAT	SAT	$\{J\}$ such that $J(y) = \mathbf{t}$, $J(z) = \mathbf{t}$
UNSAT	UNSAT	$\{J\}$ such that $J(y) = \mathbf{u}$, $J(z) = \mathbf{u}$
UNSAT	SAT	$\{J\}$ such that $J(y) = \mathbf{u}$, $J(z) = \mathbf{t}$

Hence (1) and (2) holds and the result follows. □

We move on to preferred semantics. It directly follows that for any ADF D which is *prf*-concise it holds that $Cred_{prf}$ and $Skept_{prf}$ coincide. From this observation, and general ADF complexity results, the following proposition follows.

Proposition 3.34. *For prf-concise ADFs, the problems of credulous and skeptical reasoning, and existence of non-trivial interpretations, under preferred semantics, are in* Σ_2^P *and verification, under preferred semantics, is in* Π_2^P.

As for hardness, we show that reasoning under preferred semantics in *prf*-concise ADFs is not polynomial, under complexity theoretic assumptions.

Proposition 3.35. *For prf-concise ADFs and preferred semantics, the problems of credulous and skeptical acceptance, verification, and non-trivial existence are* coNP*-hard.*

Proof. Let ϕ be a Boolean formula over variables X, and consider the coNP-complete problem of verifying whether ϕ is tautological. Construct ADF $D_\phi = (A, L, C)$ with $A = X \cup \{f\}$ s.t. f is a fresh argument not in X, $\varphi_x = \neg x$ for $x \in X$ and $\varphi_f = \phi \vee \neg f$. Observe that for any $I \in adm(D_\phi)$ we have $I(x) = \mathbf{u}$ for $x \in X$ (due to the self-attack). We show that (i) D_ϕ is *prf*-concise, and that (ii) for $I \in prf(D_\phi)$ it holds that $I(f) = \mathbf{t}$ iff ϕ is tautological. Then, ϕ is tautological iff f is credulously (skeptically) accepted iff there is a non-trivial preferred interpretation in D_ϕ iff $\{x \mapsto \mathbf{u} \mid x \in X\} \cup \{f \mapsto \mathbf{t}\}$ is preferred (i.e., verification is coNP-hard).

Consider any $I \in adm(D_\phi)$. If $I(f) = \mathbf{f}$, then $\varphi_f[I] \equiv \top$, a contradiction. A further observation is that if $I \in adm(D_\phi)$, then I assigns all $x \in X$ to undefined and f not to false. Thus, the only two possibilities for $I(f)$ are true or undefined. An interpretation satisfying $I(f) = \mathbf{u}$ is trivial, and always admissible regardless of ϕ. This implies that D_ϕ is *prf*-concise: if there is no non-trivial admissible interpretation, then the unique preferred interpretation is trivial. Otherwise, we have two admissible interpretations, one assigning f to undefined and one assigning f to true, with the latter being the unique preferred interpretation of D_ϕ.

Consider two cases: (a) ϕ is tautological and (b) ϕ is refutable. If ϕ is tautological, then $I = \{x \mapsto \mathbf{u} \mid x \in X\} \cup \{f \mapsto \mathbf{t}\}$ is admissible in D_ϕ ($\varphi_f[I] \equiv \phi[I] \vee \neg\top \equiv \phi[I] \equiv \top$), and, by the above, the unique preferred interpretation of D_ϕ. If ϕ is refutable, then $\phi[I_\mathbf{u}]$ is refutable. Let, again, be $I = \{x \mapsto \mathbf{u} \mid x \in X\} \cup \{f \mapsto \mathbf{t}\}$. It follows that I is not admissible: $\varphi_f[I] \equiv \phi[i] \vee \neg\top \equiv \phi[I] \equiv \phi[I_\mathbf{u}]$ is refutable. Thus, only $I_\mathbf{u}$ is admissible in D_ϕ, and $I_\mathbf{u}$ is the unique preferred interpretation of D_ϕ. In conclusion, for $I \in prf(D_\phi)$ we show that $I(f) = \mathbf{t}$ iff ϕ is tautological. □

Remark 1. The hardness results presented in this article for σ-concise and k-σ-concise ADFs differ from the results presented in preliminary conference version of this article [50]. The reason is that the proofs for the hardness results in the preliminary conference version had a flaw: the problem that was used to reduce from was not shown to be C-complete under randomized reductions. We nevertheless show that the fragments of σ-concise and k-σ-concise ADFs exhibit the main complexity results we aim for: (i) both classes do not result in tractability (no "in P" results for main reasoning tasks), and (ii) the same membership as shown in the preliminary conference version hold, which we require partially for our algorithms and encodings. In Table 4 one sees a overview of the results.

Summarizing this subsection, we first remark that conciseness leads to triviality for conflict-freeness and admissibility and does not affect grounded semantics due to ADFs always being *grd*-concise. For *com*-concise ADFs, reasoning has milder complexity, since one can refer to grounded reasoning. Except for skeptical reasoning, *nai*-conciseness does not support milder reasoning complexity under naive semantics. For preferred semantics and *prf*-concise ADFs, complexity is milder due to credulous and skeptical reasoning being the same. Hardness for coNP suggests that at least polynomial-time reasoning is not to be expected.

4. Distance to subclasses

In this section we extend subclasses from the previous section by considering distance notions to the subclass in question. We focus on those subclasses that exhibit milder complexity than the general class of ADFs, with the reason being that if a subclass already has "full" complexity (regarding general ADFs), then considering distances to that class cannot lead to a similar (milder) complexity than the subclass without considering distances. That is, we focus on distance notions to acyclic ADFs, bipolar ADFs, and concise ADFs, and, on the other hand, do not consider symmetric and bipartite ADFs, as these have the same complexity of reasoning as general ADFs. We uniformly name subclasses arising from distance notions to a subclass $\mathcal{C}$ by k-$\mathcal{C}$, with an integer $k \geq 0$ that represents the (numeric) distance allowed to $\mathcal{C}$. By the subsequent definitions, it follows that k-$\mathcal{C} \supseteq \mathcal{C}$ for $k \geq 1$ (i.e., considering distance notions extends a subclass). Further, it holds that 0-$\mathcal{C} = \mathcal{C}$ for the classes of acyclic and bipolar ADFs (an ADF has 0 distance to $\mathcal{C}$ iff that ADF is in $\mathcal{C}$), for concise a similar relation holds: an ADF is 1-σ-concise iff the ADF is σ-concise. A summary of our results can be seen in Table 4, when k is a fixed constant.

4.1. Distance to acyclic ADFs

We consider the following distance notion to acyclic ADFs: an ADF D is k-acyclic if removing links from parents of k arguments results in acyclicity.

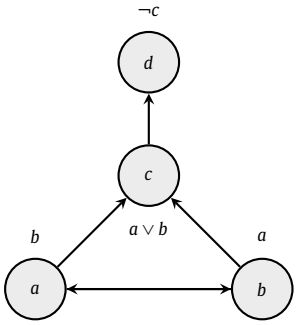

Fig. 9. A 2-acyclic ADF ($B = \{a, b\}$).

Definition 4.1. An ADF $D = (A, L, C)$ is k-acyclic if there is a set $B \subseteq A$ with $|B| \leq k$ such that $(A, L \setminus (A \times B))$ is acyclic.

Example 4.2. Consider ADF $D = (A, L, C)$ with $A = \{a, b, c, d\}$, and links and acceptance conditions as in Fig. 9. We find that removing links to $B = \{a, b\}$ results in an acyclic $L \setminus \{(a, b), (b, a)\} = \{(a, c), (b, c), (c, d)\}$.

The basic idea of moving towards milder complexity (compared to general ADFs) is the following. By "fixing" a truth value assignment on B and then computing the grounded interpretation on the resulting acyclic ADF (via polynomially many NP oracle calls) we get a potential preferred interpretation of the original ADF. By trying out all such assignments (bounded by 3^k), we enumerate preferred interpretations.

We first prove an auxiliary result: by fixing B, we have $I \in prf(D)$ is the grounded interpretation of a specifically crafted D', and if an I is grounded in D' and satisfies further consistency checks, we have $I \in adm(D)$.

Lemma 4.3. *Let $D = (A, L, C)$ be an ADF and $A' \subseteq A$ such that $(A, L \setminus (A \times A'))$ is acyclic. Further, let I be a three-valued interpretation over A, $D' = (A, L \setminus (A \times A'), \{\varphi'_a\}_{a \in A})$, $\varphi'_a = \varphi_a$ for $a \in (A \setminus A')$, and*

$$\varphi'_{a'} = \begin{cases} \top & \text{if } I(a') = \mathbf{t} \\ \bot & \text{if } I(a') = \mathbf{f} \\ \neg a' & \text{if } I(a') = \mathbf{u} \end{cases}$$

for $a' \in A'$. It holds that

- *if $I \in prf(D)$ then $I \in grd(D')$, and*
- *if $I \in grd(D')$ and $\forall a' \in A' : I(a') \leq_i \Gamma_D(I)(a')$ then $I \in adm(D)$.*

Proof. Let I be a preferred interpretation of D and G be the grounded interpretation of D' constructed from I (as defined in the lemma). From the assumptions in the lemma, it follows that $I(a') = G(a')$ for $a' \in A'$. Let $a \in A \setminus A'$. We now show that $I(a) = G(a)$. This can be proven by induction, which follows the same line of reasoning as the proof of Proposition 3.3, except that S_0 contains all arguments in A' (the self-attackers are put into the same partition as the arguments with constant acceptance condition). This implies that D' has exactly one complete interpretation (note that arguments with self-attacking parents are uniquely determined via interpretations that are admissible via $\Gamma_{D'}$). Now suppose that there is an $a \in A \setminus A'$ such that $I(a) \neq G(a)$. Then there is a partition S_i (as in the proof of Proposition 3.3) with $a \in S_i$. W.l.o.g., assume that a is such that i is minimum over all arguments in disagreement for I and G. It follows that $i \neq 1$. Then it follows that $\varphi_a = \varphi'_a$ (same acceptance condition in both D and D'). Further, all parents of a (in both D and D') are assigned the same value in I and G. Thus, $\Gamma_D(I)(a) = \Gamma_{D'}(G)(a)$, implying that I and G are the same.

For the second item, assume that I is grounded in D', and that for all $a' \in A'$ we have $I(a') \leq_i \Gamma_D(I)(a')$ (I is "admissible" w.r.t. all arguments in A' and acceptance conditions in D). We now show that I is admissible in D. Suppose the contrary, i.e., I is not admissible in D, then $\exists a \in A \setminus A'$ with $I(a) \not\leq_i \Gamma_D(I)(a)$. This immediately implies a contradiction: we know that $I(a) = \Gamma_{D'}(I)(a)$ (I is grounded in D'), and for all $a \in A \setminus A'$ it holds that $\varphi_a = \varphi'_a$. □

From the preceding lemma we can infer an algorithm for computing all preferred interpretations of an ADF D: find a $B \subseteq A$ that induces an acyclic ADF if fixed. Since each preferred interpretation I of D assigns some values to the arguments in B, if we construct for each of these assignments on B the corresponding ADF D', we can collect all preferred interpretations.

Proposition 4.4. *For $k \geq 0$ and k-acyclic ADFs, the following problems are in Δ_2^{P}:*

- *credulous and skeptical acceptance under $\sigma \in \{adm, com, prf\}$, and*

- *verification and existence of non-trivial interpretations under preferred semantics.*

Proof. Let $D = (A, L, C)$ be a k-acyclic ADF and $a \in A$, for a constant integer $k \geq 1$. First, one can find a set A', with $|A'| \leq k$ such that $(A, L \setminus (A \times A'))$ is acyclic, in polynomial time by the procedure described in [36, Proposition 1]: iterate through all subsets $A' \subseteq A$ with $|A'| \leq k$ and check whether the property holds. There are $\sum_{i=0}^{k} \binom{|A|}{i} \leq |A|^k$ such sets. This means one can find such an A', if it exists, in polynomial time (assuming k is a constant).

There are 3^k many truth value assignments on A' for $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. Let I_i, for $i \in \{1, \ldots, 3^k\}$ be these assignments on A'. For each i do the following: construct $D_i = (A, L', C')$ with

- $\varphi_a = \varphi_a$ if $a \in A \setminus A'$,
- $\varphi_{a'} = \begin{cases} \top & \text{if } I_i(a') = \mathbf{t} \\ \bot & \text{if } I_i(a') = \mathbf{f} \\ \neg a' & \text{if } I_i(a') = \mathbf{u} \end{cases}$, for $a' \in A'$.

Now compute, in polynomially many steps, the grounded interpretation of D_i via iteratively applying Γ_{D_i} on the initial interpretation J assigning all arguments to undefined. This proceeds by checking, for each argument x, whether $\Gamma_{D_i}(J)(a) = v$ (which is in coNP for $v \in \{\mathbf{t}, \mathbf{f}\}$). If the answer is yes, then assign in J the corresponding value v to x. There are polynomially many arguments in D_i, and, thus, polynomially many calls to a coNP oracle. Let G be the grounded interpretation of D'. Then, check whether $I_i(a') \leq_i \Gamma_D(G)(a')$ for each $a' \in A'$ (thus, check for the original acceptance condition whether $\varphi_{a'}[G]$ is valid if $I_i(a')$ is true, and, similarly for other truth values). If this succeeds, then, by Lemma 4.3, we have $G \in adm(D)$. Let $I \in prf(D)$ (i.e., I is preferred in the original ADF D). It holds that there is an I_i (defined above) with $I_i \leq_i I$ (all three-valued combinations of assignments on A' are part of the I_i's, and each preferred interpretation of D must assign a value to each argument in A'). By constructing D_i and the corresponding grounded interpretation G_i and subsequent check as specified above, for all $i \in \{1, \ldots, 3^k\}$, we arrive, again by Lemma 4.3, at a superset of $prf(D)$ of admissible interpretations. This holds, since if $I \in prf(D)$, then $I \in grd(D_i)$ for some of the i's above, which we collect all. Note that the check whether a grounded interpretation is admissible in the original D is required: otherwise we might get a non-admissible interpretation that is more informative than a preferred interpretation. By filtering all information maximal interpretations (which can be done by pairwise comparison), one can compute $prf(D)$ in polynomial time and a coNP oracle. All reasoning tasks in the proposition can then be answered by checking whether the queried argument is assigned to true in one (all) preferred interpretation(s) of D. □

Nevertheless, k-acyclicity does not imply full tractability, in the sense of having (many) problems in P, like for acyclic ADFs. Reasoning under *grd* is not easier on k-acyclic ADFs, since, intuitively, the undefined value may be propagated, which can imply that one still has to check for tautologies or unsatisfiability of formulas.

Proposition 4.5. *For 1-acyclic ADFs, it holds that credulous and skeptical acceptance, verification, and existence of non-trivial interpretations, are* coNP*-hard under grounded semantics.*

Proof. We show a reduction from the problem of deciding whether a given Boolean formula ϕ, with vocabulary X, is a tautology (Fig. 10). Construct ADF $D = (X \cup \{a, b, c, f\}, L, C)$ with a, b, c, and f fresh arguments not in X, $\varphi_x = c$ for $x \in X$, $\varphi_a = \neg c$, $\varphi_b = \neg a$, $\varphi_c = \neg b$, and $\varphi_f = \phi$. The graph $(A \setminus \{c\}, L \setminus (A \times \{c\}))$ is acyclic. Let I be an arbitrary admissible interpretation of D. It holds that $I(x) = \mathbf{u} = I(a) = I(b) = I(c)$ for $x \in X$ (arguments a, b, and c are in an odd attack cycle, and the arguments representing variables may only have a truth value differently than undefined if c is assigned true or false). We claim that f is credulously (skeptically) accepted in D under grounded semantics iff ϕ is a tautology. Assume that ϕ is a tautology. Then the grounded interpretation of D assigns all arguments to undefined, except for f which is assigned true. For the other direction, assume that f is assigned to true in the grounded interpretation I of D. Since the grounded interpretation assigns all other arguments, then, to undefined (the grounded interpretation is admissible), it holds that $\varphi_f[I] = \phi$ is tautological. □

The proof of this result relies on a small odd-attack-cycle so that an argument can be assigned to true iff a formula is unsatisfiable. This can be used to show the next results. In particular, the proof can be straightforwardly adapted for credulous acceptance under admissible, complete and preferred, and skeptical acceptance under complete and preferred semantics: the only non-trivial admissible interpretation in this ADF assigns f to true iff ϕ is a tautology. Note that we can assume that ϕ in the proof is satisfiable and f is never assigned false in an admissible interpretation: ψ is a tautology iff $\psi \vee d = \phi$ is a tautology for d a fresh atom not occurring in ψ.

Corollary 4.6. *For 1-acyclic ADFs, we find that*

- *checking credulous acceptance, verification, as well as existence of non-trivial interpretations, are* coNP*-hard problems under* $\sigma \in \{adm, com, prf\}$*, and*

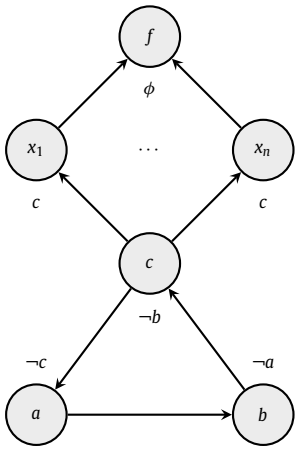

Fig. 10. Illustration of hardness construction in proof of Proposition 4.5.

- *skeptical acceptance is* coNP*-hard for* $\tau \in \{com, prf\}$.

Existence of non-trivial conflict-free (naive) interpretations in k-acyclic ADFs is trivial if $k < |A|$ and decidable in polynomial time if $k \geq |A|$. In the former case, there is an argument not depending on itself (whose acceptance condition is either satisfiable or refutable, and the argument can be assigned true or false without conflict). In the latter case, one can enumerate all truth value assignments and check for conflict-freeness.

Proposition 4.7. *For k-acyclic ADFs, existence of non-trivial interpretations is in* P *for* $\sigma \in \{cf, nai\}$.

For the remaining reasoning tasks, we refer to the acyclic (0-acyclic) case for hardness.

In this subsection we conclude that k-acyclicity does indeed support milder complexity for admissibility, and complete and preferred semantics. By finding a set of arguments witnessing k-acyclicity and one can check each possibility of assigning these arguments values and propagate the result. If the set is small, overall running time can benefit. Under grounded semantics, k-acyclicity does not appear to help much.

4.2. Distance to bipolar ADFs

We consider ADFs that have a distance k to bipolar ADFs. Recall that we refer to attacking links by L_D^-, to supporting links by L_D^+, and to dependent links by $L_D^?$, for an ADF D. A natural approach to such a distance is to view bipolar ADFs as ADFs with no non-bipolar, i.e., no dependent links, and to extend this fragment of ADFs by considering all ADFs $D = (A, L, C)$ that have at most $k \geq 0$ dependent links per argument $a \in A$. That is, we might have, overall, more than k dependent links $L_D^? > k$, yet, for each $a \in A$ we require that $|\{(a, b) \in L_D^?\}|$ is at most k.

Definition 4.8. An ADF D is k-bipolar for some integer $k \geq 0$ if for each $a \in A$ it holds that $|\{(b, a) \in L_D^?\}| \leq k$.

It follows directly that the set of 0-bipolar ADFs coincides with the set of bipolar ADFs.

Similarly as for bipolar ADFs, we assume that for a k-bipolar ADF D, the (non-)polarity of links is known, i.e, the sets L_D^-, L_D^+, and $L_D^?$ are given.

For an intuition on the complexity of k-bipolar ADFs, a similar concept as for bipolar can be applied. Recall that in an ADF $D = (A, L, C)$ a link $(b, a) \in L$ is

- supporting if for every two-valued interpretation I, $I(\varphi_a) = \mathbf{t}$ implies $I|_{\mathbf{t}}^b(\varphi_a) = \mathbf{t}$; or
- attacking if for every two-valued interpretation I, $I(\varphi_a) = \mathbf{f}$ implies $I|_{\mathbf{t}}^b(\varphi_a) = \mathbf{f}$.

Note that we can write these two conditions slightly differently, as link $(b, a) \in L$ is

- supporting if for every two-valued interpretation I, $I|_{\mathbf{t}}^b(\varphi_a) = \mathbf{f}$ implies $I(\varphi_a) = \mathbf{f}$; or
- attacking if for every two-valued interpretation I, $I|_{\mathbf{t}}^b(\varphi_a) = \mathbf{t}$ implies $I(\varphi_a) = \mathbf{t}$.

Thus, if a two-valued interpretation refutes φ_a, then assigning supporter b of a from true to false does not lead to acceptance of a (otherwise we would have a case where a two-valued interpretation satisfies φ_a and assigning a supporter to true refutes the condition).

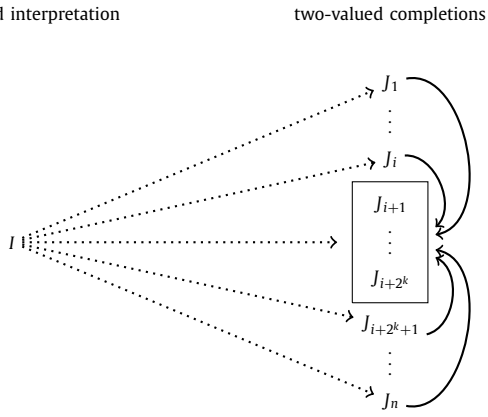

Fig. 11. Conceptual view on k-bipolar acceptance conditions. Consider the case when checking whether a three-valued interpretation I and k-bipolar acceptance condition φ_a satisfy $\mathbf{t} = I(a) \leq_i \Gamma_D(I)(a)$. Out of all two-valued completions J of I (completions denoted by dashed arrows), 2^k many completions J_{i+1} to J_{i+2^k} assign all undefined arguments to true when they attack a and to false when they support a, and all two-valued truth value combinations to the dependent links. It holds that J is not a model of φ_a implies that some J_x, $i+1 \leq x \leq i+2^k$, is not a model of φ_a (this implication is represented by solid arrows).

Consider some interpretation I for the ADF D, and some $a \in A$. Say, we want to check whether $I(a) \leq_i \Gamma_D(I)(a)$ holds. Thus, we want to check whether $\varphi_a[I]$ is (i) a tautology if $I(a) = \mathbf{t}$, unsatisfiable if $I(a) = \mathbf{f}$, or whether $I(a) = \mathbf{u}$. The latter case is trivial. Let us focus on the first one, the case with false is analogous. Say that $\varphi_a[I]$ is not a tautology, implying that there is a completion J of I such that $J(\varphi_a) = \mathbf{f}$. Since $I \leq_i J$, it follows that I and J differ only in the assignment to arguments which I assigns to undefined. Pick an argument b, such that (b, a) is supporting. By the above, $J|^b_{\mathbf{f}}(\varphi_a) = \mathbf{f}$. This is preserved if we assign all supporters of a to false. Likewise, if we assign all attackers to true, the refutation is preserved. If D is bipolar, this results in a uniquely determined two-valued interpretation J' with the appealing property that if a J refuting φ_a exists, a J' can be determined that refutes φ_a, as well. Further, we can restrict re-assignment by only considering parents of a that have been assigned undefined by I to get a uniquely determined J' with the same properties. If one computes such a J' we have $I \leq_i J'$, and one can determine J' solely by I and knowledge of supporters and attackers of a (in polynomial time).

For the case of k-bipolar ADFs, say we are again given a three-valued interpretation I and want to check whether $\varphi_a[I]$ is not a tautology. Assume that $\varphi_a[I]$ is not a tautology, we have a J with $I \leq_i J$ that refutes φ_a. By assigning all attackers of a to true in J that are undefined in I, and all supporters of a to false that are undefined in I, we get a J' that, again, refutes φ_a, and $I \leq_i J'$. Now, note that if we construct an I' from I by assigning among the arguments assigned to undefined by I all supporters of a to false and all attackers of a to true, we get an I' with $I \leq_i I'$. Importantly, $I' \leq_i J'$. Since for any witnessing refuting J of $\varphi_a[I]$ there is a corresponding J' with $I' \leq_i J'$, it follows that it suffices to consider $\varphi_a[I']$. Notably, in a k-bipolar ADF the number of undefined arguments in I' is at most k. Thus, checking whether $\varphi_a[I']$ is a tautology can be achieved by considering 2^k completions, a number that is exponential in k, but polynomial if k is assumed to be a fixed constant. An illustration of this process is also shown in Fig. 11.

After the above intuition, we formally prove the main result of for k-bipolar ADFs: as stated above, the main ingredient for the complexity of k-bipolar ADFs is a generalization of why bipolar ADFs exhibit milder complexity [69].

Lemma 4.9. *Let $k \geq 0$ be a fixed constant. Given a k-bipolar ADF $D = (A, L, C)$, an interpretation I over A, and an argument $a \in A$, deciding whether*

- *$\varphi_a[I]$ is a tautology,*
- *$\varphi_a[I]$ is unsatisfiable,*
- *$\varphi_a[I]$ is satisfiable, or deciding whether*
- *$\varphi_a[I]$ is refutable*

is in P.

Proof. Let $I' = \{b \mapsto \mathbf{t} \mid I(b) = \mathbf{t}\} \cup \{b \mapsto \mathbf{t} \mid I(b) = \mathbf{u}, (b, a) \in L^-_D\} \cup \{b \mapsto \mathbf{f} \mid I(b) = \mathbf{f}\} \cup \{b \mapsto \mathbf{f} \mid I(b) = \mathbf{u}, (b, a) \in L^+_D\} \cup \{b \mapsto \mathbf{u} \mid I(b) = \mathbf{u}, (b, a) \in L^?_D\}$. It was shown in the proof of [69, Proposition 5.1] that $\varphi_a[I]$ is a tautology iff $\varphi_a[I']$ is a tautology. Let X be the set of arguments mapped to $\mathbf{u}$ by I'. It holds that $\varphi_a[I']$ is a tautology iff $\varphi_a[I'][I_X]$ evaluates to true under every two-valued interpretation I_X over X. Since $X \subseteq \{(b, a) \in L^?_D\}$ and therefore $|X| \leq k$ and k is constant, this can be done in polynomial time. Hence checking whether $\varphi_a[I]$ is a tautology is in P.

The same reasoning applies to deciding whether $\varphi_a[I]$ is unsatisfiable, which, according to [69, Proposition 5.1], is equivalent with deciding whether $\varphi_a[I'']$ is unsatisfiable, where $I'' = \{b \mapsto \mathbf{t} \mid I(b) = \mathbf{t}\} \cup \{b \mapsto \mathbf{t} \mid I(b) = \mathbf{u}, (b, a) \in L^+_D\} \cup \{b \mapsto \mathbf{f} \mid$

$I(b) = \mathbf{f}\} \cup \{b \mapsto \mathbf{f} \mid I(b) = \mathbf{u}, (b,a) \in L_D^-\} \cup \{b \mapsto \mathbf{u} \mid I(b) = \mathbf{u}, (b,a) \in L_D^?\}$. Finally, a formula is satisfiable if the formula is not unsatisfiable, and a formula is refutable if the formula is not a tautology. □

It follows that reasoning on k-bipolar and bipolar ADFs has the same complexity; the proof uses the generic results of [69, Theorem 3.18 and Corollary 3.19] and Lemma 4.9. We rephrase (specialize) this theorem for our purposes and definitions, next.

Theorem 4.10 *(adapted from [69, Theorem 3.18 and Corollary 3.19]). Let $\mathcal{D}$ be a set of ADFs. If for each $D = (A, L, C) \in \mathcal{D}$ it holds that one can check $\Gamma_D(D)(a) = v$ in polynomial time, for any I over A, $a \in A$, and any $v \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, then*

- *Ver_{cf}, $Cred_{cf}$, Ver_{nai}, $Cred_{nai}$, Ver_{adm}, Ver_{com}, Ver_{grd} and $Exists_{grd}$ are problems in P,*
- *$Cred_{adm}$, $Cred_{com}$, and $Cred_{prf}$ are problems in* NP*,*
- *Ver_{prf} is in* coNP*, and*
- *$Skept_{prf}$ is in Π_2^P,*

for the fragment $\mathcal{D}$ of ADFs.

Theorem 4.11. *Let $k \geq 0$ be a constant and $\sigma \in \{cf, nai, adm, com, grd, prf\}$. It holds that the complexity of Ver_σ, $Exists_\sigma$, $Skept_\sigma$, and $Cred_\sigma$ coincides for k-bipolar and bipolar ADFs.*

Proof. We first note that hardness carries over from bipolar ADFs (cf. Table 3). By Theorem 4.10, several membership results follow; it remains to show membership for $Cred_{grd}$, $Exists_\tau$, and $Skept_{\tau'}$ with $\tau \in \{cf, nai, adm, com, prf\}$ and $\tau' \in \{cf, nai, adm, com, grd\}$.

$Exists_{\sigma'}$ for $\sigma' \in \{cf, nai\}$ is in P as, given a k-bipolar ADF $D = (A, L, C)$, we can check for each $a \in A$ whether the interpretation $I_a^{\mathbf{t}} = \{a \mapsto \mathbf{t}\} \cup \{b \mapsto \mathbf{u} \mid b \in A \setminus \{a\}\}$ or the interpretation $I_a^{\mathbf{f}} = \{a \mapsto \mathbf{f}\} \cup \{b \mapsto \mathbf{u} \mid b \in A \setminus \{a\}\}$ is conflict-free in D in polynomial time (by checking whether $\varphi_a[I_a^{\mathbf{t}}]$ is satisfiable or $\varphi_a[I_a^{\mathbf{f}}]$ is refutable, which is both in P by Lemma 4.9). If one of these interpretations is conflict-free, the answer to $Exists_{\sigma'}$ is yes, otherwise it is no. For the admissibility-based semantics, $Exists_{\sigma'}$ for $\sigma' \in \{adm, com, prf\}$ is in NP as we can guess a non-trivial interpretation an check if it is admissible in polynomial time due to Lemma 4.9.

All reasoning tasks associated with grounded semantics can be decided in polynomial time, since the unique grounded interpretation can be computed in polynomial time (by iteratively computing Γ_D starting with the trivial interpretation).

Regarding skeptical reasoning, for *cf* and *adm* skeptical reasoning is trivial. We have $Skept_{com}$ coincides with $Skept_{grd}$, implying that both can be decided in polynomial time. Finally, $Skept_{nai}$ is in coNP as we can guess an interpretation that assigns false or undefined to a queried argument and verify naiveness in polynomial time. □

The preceding theorem suggests that k-bipolarity indeed offers a great opportunity for milder complexity of reasoning, since complexity coincides with full bipolar ADFs for the mentioned cases.

4.3. Distance to concise ADFs

The final distance notion we consider is k-conciseness. An ADF D is σ-concise if $|\sigma(D)| = 1$. A straightforward generalization is to say that D is k-σ-concise, which holds if D has at most k σ-interpretations.

Definition 4.12. An ADF D is k-σ-concise for a semantics σ and some integer $k \geq 1$ if $|\sigma(D)| \leq k$.

Similarly as for σ-concise ADFs, k-σ-conciseness restricts existence of too many "diverging" σ-interpretations. In terms of computation, (k-)σ-conciseness translates into less non-determinism and smaller search spaces. We formalize this thought below, but an intuition why this is the case can be seen when considering k-*prf*-concise ADFs. Such an ADF has at most k preferred interpretations. This means when computing preferred interpretations in an iterative manner, we can stop after reaching k preferred interpretations. If we face $|A|$ many arguments, and k is significantly smaller than $3^{|A|}$, a large portion of potential preferred interpretations can be ruled out after finding k many.

We remark here that the following complexity analyses for k-σ-concise ADFs assume that the given ADF is k-σ-concise. That is, we presume that there are at most k σ-interpretations. For general ADFs (not necessarily being k-σ-concise) there does not appear to be an efficient procedure to determine the actual number of σ-interpretations, e.g., in a preprocessing step. However, implementations of the principles behind our complexity analyses here are useful in terms of that an algorithm does not need to know the exact number beforehand: search can continue until no more (preferred) interpretations are found. If the actual number of preferred interpretations is low, the overall running time benefits (e.g., in Algorithm 2).

Nevertheless, k-σ-conciseness does not imply mild(er) complexity in all cases. We begin our complexity analysis of this fragment with conflict-free semantics.

Proposition 4.13. *For k-cf-concise ADFs, it holds that credulous acceptance, existence of non-trivial interpretations and verification under conflict-free semantics are* NP*-complete problems for every* $k \geq 2$.

Proof. The membership results follow from general ADFs. We proceed by showing NP-hardness of all three problems by a construction that is a slight variation of [69, Reduction 3.1]. Let ϕ be a formula over vocabulary X. Define the ADF $D_\phi = (A, L, C)$ such that $A = X \cup \{z\}$ with z a fresh argument not in X, $\varphi_x = \neg x$ for $x \in X$, and $\varphi_z = \phi \vee \neg z$. It holds that

1. If ϕ is satisfiable then $cf(D_\phi) = \{I_{\mathbf{u}}, \{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}\}$.
2. If ϕ is unsatisfiable then $cf(D_\phi) = \{I_{\mathbf{u}}\}$.

Let $I = \{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}$. (1) Assume ϕ is satisfiable. We show that, besides $I_{\mathbf{u}}$, I is the only conflict-free interpretation of D_ϕ. We get $\varphi_z[I] = \phi \vee \neg\top \equiv \phi$, hence $\varphi_z[I]$ is satisfiable and, consequently, I is conflict-free. On the other hand, consider I' such that $I'(z) = \mathbf{f}$. Then $\varphi_z[I'] = \phi \vee \neg\bot \equiv \top$, hence $\varphi_z[I']$ is not refutable and, consequently, I' is not conflict-free. Moreover, consider I' such that $I(x) \neq \mathbf{u}$ for some $x \in X$. If $I'(x) = \mathbf{t}$ then $\varphi_x[I'] = \neg\top \equiv \bot$, i.e., $\varphi_x[I']$ is not satisfiable; if $I'(x) = \mathbf{f}$ then $\varphi_x[I'] = \neg\bot \equiv \top$, i.e., $\varphi_x[I']$ is not refutable. In both cases, I' is not conflict-free. We conclude that $cf(D_\phi) = \{I_{\mathbf{u}}, I\}$. (2) Assume ϕ is unsatisfiable. Then $\varphi_z[I] \equiv \phi$ is also not satisfiable, hence I is not conflict-free. Also for I' with $I'(z) = \mathbf{f}$ we again have $\varphi_z[I'] \equiv \top$, i.e., I' is not conflict-free. Since, as before, any I' such that $I'(x) \neq \mathbf{u}$ for some $x \in X$ is not conflict-free in D_ϕ, we conclude that $I_{\mathbf{u}}$ is the only conflict-free interpretation of D_ϕ.

It follows that D_ϕ is 2-*cf*-concise and ϕ is satisfiable iff z is credulously accepted in D_ϕ under *cf* iff $I \in cf(D_\phi)$ (note that I is non-trivial). Hence NP-hardness follows for $Cred_{cf}$, $Exists_{cf}$, and Ver_{cf}. □

Verifying admissibility of an interpretation does also not benefit from a bounded number of interpretations.

Proposition 4.14. *For k-adm-concise ADFs, it holds that* Ver_{adm} *is* coNP*-complete for every* $k \geq 2$.

Proof. For hardness we use [69, Reduction 3.1]. The obtained ADF D_ϕ has either $adm(D_\phi) = \{I_{\mathbf{u}}\}$ in case that ϕ is not a tautology but satisfiable, $adm(D_\phi) = \{I_{\mathbf{u}}, \{z \mapsto \mathbf{t}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}\}$ in case that ϕ is a tautology, or $adm(D_\phi) = \{I_{\mathbf{u}}, \{z \mapsto \mathbf{f}\} \cup \{x \mapsto \mathbf{u} \mid x \in X\}\}$ in case that ϕ is unsatisfiable. Hence D_ϕ is 2-*adm*-concise □

Despite the preceding results, an upper bound on the number of interpretations is beneficial for skeptical acceptance under naive and preferred semantics. The idea is to iteratively construct conflict-free or admissible interpretations that are more informative than the previous ones, until an information-maximal interpretation is reached. If the number of information-maximal interpretation is bounded from above, the overall process results in a polynomial-time algorithm with access to certain oracles (depending on the complexity of finding more informative interpretations).

We first show complexity of finding more informative interpretations.

Definition 4.15. The problem $Exists^{>}_{\sigma}$ for a semantics σ decides, given an ADF D and an interpretation I, whether there exists an interpretation $J \in \sigma(D)$ such that $I <_i J$.

Proposition 4.16. $Exists^{>}_{\sigma}$ *is* NP*-complete for* $\sigma = \{cf, nai\}$, *and* Σ^{P}_2*-complete for* $\sigma \in \{adm, com, prf\}$.

Proof. Hardness is immediate for all semantics, as already $Exists_\sigma$ is a special case of $Exists^{>}_{\sigma}$ (with $I_{\mathbf{u}}$) and is already hard for the respective classes. Membership is by the following algorithms, given an ADF D and interpretation I:

- $\sigma = cf$: Guess an interpretation J and further two-valued interpretations $J_s \geq_i J$ for each $s \in A$ with $J(s) = \mathbf{t}$ or $J(s) = \mathbf{f}$. Then check for each J_s: if $J(s) = \mathbf{t}$ whether $J_s(\varphi_s) = \mathbf{t}$ holds, and if $J(s) = \mathbf{f}$ whether $J_s(\varphi_s) = \mathbf{f}$ holds (conflict-free property). Finally check $J >_i I$. All checks can be done in polynomial time.
- $\sigma = adm$: Guess an interpretation J and check whether $J \in adm(D)$ (in coNP) and $J >_i I$.
- $\sigma \in \{nai, com, prf\}$: For each $I \in cf(D)$ (resp. $I \in adm(D)$) there is some $J \in nai(D)$ (res. $J \in com(D)$ and $J \in prf(D)$) with $J \geq_i I$. □

Based on this result, we prove the next two results for skeptical reasoning under naive and preferred semantics.

Corollary 4.17. *For k-nai-concise ADFs,* $Skept_{nai}$ *is in* Δ^{P}_2.

Proof. We present an algorithm which, given an ADF D, iteratively computes the set $nai(D)$. We initialize the set $\mathcal{I}$ to collect the naive interpretations with $\emptyset$. We construct a new naive interpretation I of D as follows: start with $I = I_{\mathbf{u}}$ and ask an NP oracle whether there exists some $J \in cf(D)$ with $J >_i I$ (in NP by Proposition 4.16) and there is no $J' \in \mathcal{I}$ with $J \leq_i J'$. If yes, set $I = J$ and repeat the oracle call. If no, either $I = I_{\mathbf{u}}$ or $I \in nai(D)$. If $I \neq I_{\mathbf{u}}$, add I to $\mathcal{I}$ and repeat the

process to construct the next naive interpretation. If $I = I_{\mathbf{u}}$ there is no non-trivial conflict-free interpretation that is not contained in one of the naive interpretations already found, hence the algorithm terminates.

Note that an NP-oracle, by definition an oracle for a decision problem, does not directly yield an interpretation as witness for answering a decision problem. However, with polynomially many such oracle calls, the witness can be constructed: first ask whether there is a witness, if so ask whether the first argument (wrt. an arbitrary linear order over arguments) is true (or, if unsuccessful, whether it is false). Continuing over all arguments produces the witness interpretation.

By assumption, $nai(D)$ is of polynomial size and for finding a naive interpretation a linear number of oracle calls (limited by the number of arguments) is necessary. Hence the overall runtime is polynomial in the input size. □

Under *prf*, using a similar idea, we can decide the problem with a polynomial number of Σ_2^{P}-oracle calls.

Theorem 4.18. *For k-prf-concise ADFs, $Skept_{prf}$ is in Δ_3^{P}.*

Proof. Let $\mathcal{I} = \emptyset$. Construct a preferred interpretation I of $D = (A, L, C)$ by starting with $I = I_{\mathbf{u}}$ and ask a Σ_2^{P} oracle whether there is a $J \in adm(D)$ with $J >_i I$ and no $J' \in \mathcal{I}$ with $J \leq_i J'$. If yes, set $I = J$ and repeat the oracle call. Otherwise, if $I \neq I_{\mathbf{u}}$, add I to $\mathcal{I}$ and proceed to find the next preferred interpretation. $|prf(D)|$ is constant and a linear number of calls suffices to find a preferred interpretation. □

The preceding results show that we get milder complexity in k-concise ADFs. Interestingly, we can combine k-bipolarity and k-conciseness, for certain semantics, and show even milder complexity.

Inspecting Theorem 4.11, the only problem for (k-)bipolar which is hard for the second level of the polynomial hierarchy is $Skept_{prf}$. If, in addition to bipolarity, also the number of preferred interpretations is bounded, we can again show that a polynomial number of NP-oracle calls is sufficient.

We first prove the complexity of $Exists_\sigma^>$ for $\sigma \in \{adm, com, prf\}$ in k-bipolar ADFs.

Theorem 4.19. *Let $k \geq 0$ be a constant integer and $\sigma \in \{adm, com, prf\}$. For a given k-bipolar ADF, $Exists_\sigma^>$ is* NP*-complete.*

Proof. Hardness follows from AFs (which can be directly translated to bipolar ADFs, see [16]).

Membership in NP is by the following procedure given a k-bipolar ADF D and an interpretation I: guess an interpretation J with $I <_i J$ and check whether $J \in adm(D)$. Due to Lemma 4.9 the check is in polynomial time. As $J \in adm(D)$ implies that there also exists some $J' \in com(D)$ and $J' \in prf(D)$ with $J \leq_i J'$, the procedure is valid for $\sigma \in \{adm, com, prf\}$. □

Combining distances to bipolarity and conciseness yields membership in Δ_2^{P} for skeptical reasoning under preferred semantics.

Theorem 4.20. *For k_1-bipolar and k_2-prf-concise ADFs, it holds that $Skept_{prf}$ is in Δ_2^{P}.*

Proof. This works just as the proof of Theorem 4.17, just that we construct preferred interpretations by iterative calls of $Exists_{adm}^>$, which in turn is in NP for k-bipolar ADFs (cf. Theorem 4.19). □

According to our results, a main benefit of k-σ-conciseness is found when looking at naive or preferred semantics and skeptical reasoning. Then, algorithmic approaches can terminate earlier. We utilized the underlying approach, together with k-bipolarity, in our algorithm design in Section 5.

4.4. Relations between subclasses

For most subclasses considered in this article there is no direct relation, i.e., for most subclasses $\mathcal{C}$ and $\mathcal{C}'$ one can find an ADF D such that D is part of $\mathcal{C}$ but not in $\mathcal{C}'$. This is immediate for the subclasses restricting the graph structure: acyclic ADFs (i.e., directed acyclic graphs), symmetric directed graphs, and bipartite directed graphs do not enjoy any subset inclusion.[3]

Any subclass restricting the graph structure is not contained in, and does not contain, bipolar ADFs. The reason is that one can write acceptance conditions that are non-bipolar, if at least two parents are available.

Concise ADFs, while not in any direct relation with symmetric, bipartite, and bipolar ADFs (except for any ADF being trivially *grd*-concise), in fact, have a relation with acyclic ADFs. Recall Proposition 3.3 stating that each acyclic ADFs has

[3] We remark that we defined the notions of acyclicity, bipartiteness, and symmetry on directed graphs. For undirected graphs, it holds that an undirected acyclic graph (i.e., a forest) is an undirected bipartite graph. One can, alternatively, define the graph notions studied in this paper also on the undirected graph that can be obtained from an ADF's dependency graph (by omitting the directions), however, we opted for the current definitions that, we think, are more directly aligned to represent dependency structures of ADFs.

a unique two-valued complete interpretation and a unique preferred interpretation. This means that an acyclic ADF is σ-concise, for $\sigma \in \{com, prf\}$.

Corollary 4.21. *Let D be an acyclic ADF. It follows that D is σ-concise for $\sigma \in \{com, prf\}$.*

Our complexity results mirror this fact, in the sense that reasoning under complete and preferred semantics has milder complexity on acyclic ADFs than on σ-concise ADFs. Note that for other semantics this relationship does not hold in general. Acyclic ADFs may have multiple conflict-free, naive, and admissible interpretations. Finally, σ-concise ADFs might not be acyclic. The preceding corollary can be strengthened to the following statement: a k-acyclic ADF imposes also a restriction on the number of preferred interpretations.

Proposition 4.22. *Each k-acyclic ADF D is 3^k-prf-concise.*

Proof. This result follows from Lemma 4.3. Assume that $D = (A, L, C)$ is k-acyclic, then there exists an $A' \subseteq A$ such that $(A, L \setminus (A \times A'))$ is acyclic. If $I \in prf(D)$ then, by Lemma 4.3, it holds that $I \in grd(D')$ for ADF D' as defined in that lemma for I. Since there are at most 3^k many ADFs D' that can be constructed, as specified in the lemma, it follows that D has at most 3^k many preferred interpretations (for a given D' there can be at most one corresponding preferred interpretation). □

5. Encodings and algorithms

We now present our algorithms for ADF acceptance tasks based on results from Sections 3 and 4. In particular, we focus on $Cred_{adm} = Cred_{com} = Cred_{prf}$ and $Skept_{prf}$ for general, bipolar, and k-bipolar ADFs. This choice covers all acceptance tasks which remain NP-hard for bipolar and k-bipolar ADFs apart from $Skept_{nai}$. Note that our implementation covers all acceptance tasks considered in this work, including reasoning procedures for $Skept_{nai}$ and $Cred_{grd} = Skept_{grd} = Skept_{com}$, as detailed in Section 6.1.

Recall that $Cred_{adm}$ is Σ_2^{P}-complete for general ADFs and NP-complete for bipolar and k-bipolar ADFs. On the other hand, $Skept_{prf}$ is Π_3^{P}-complete for general ADFs and Π_2^{P}-complete for bipolar and k-bipolar ADFs. Hence, for (k-)bipolar ADFs we outline distinct procedures based on lower complexity. The backbone of our algorithms are SAT solvers, to which we delegate (sub)problems that are in NP. A central subproblem is $Exists_\sigma^>$, which asks for a σ-interpretation strictly more informative than a given interpretation.

Directly from previous results (Proposition 4.16, Theorem 4.19, and a non-deterministic construction of a three-valued interpretation and polynomial check for verifying conflict-freeness in k-bipolar ADFs) we obtain the following corollary which summarizes complexity of $Exists_\sigma^>$ for $\sigma \in \{cf, adm\}$ which we apply for subsequent algorithms.

Corollary 5.1. *Let $k \geq 0$ be a constant integer. The problem $Exists_{adm}^>$ is Σ_2^{P}-complete for general ADFs and NP-complete for (k-)bipolar ADFs. The problem $Exists_{cf}^>$ is in NP for general and (k-)bipolar ADFs.*

In the following subsections we first define and discuss the encodings utilized in SAT calls, and then present the base algorithms for the considered problems. For the rest of this section, let $D = (A, L, C)$ be an input ADF, and I a fixed (but arbitrary) interpretation for the task $Exists_\sigma^>$.

5.1. Encodings

Recall that the problem $Exists_{cf}^>$ is in NP, hence admitting a succinct encoding to SAT. We begin by declaring Boolean variables $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ for each argument $s \in A$, with the intended meaning that for a witnessing interpretation J, assigning $s^{\mathbf{t}}$ to true corresponds to $J(s) = \mathbf{t}$, assigning $s^{\mathbf{f}}$ to true corresponds to $J(s) = \mathbf{f}$, and assigning both $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ to false corresponds to $J(s) = \mathbf{u}$. In addition, we declare for each $(r, s) \in L$ variables p_s^r representing, for the acceptance condition of $s \in A$, a guess of a two-valued interpretation J_s satisfying $J_s \geq_i J$, aiming to satisfy (resp. refute) φ_s in accordance with Definition 2.6. We denote by $\varphi_s^{\downarrow} = \varphi_s[r \mapsto p_s^r \mid (r, s) \in L]$ the acceptance condition φ_s where each occurrence of $r \in par_D(s)$ is replaced by p_s^r. Now, the formula

$$\phi(D) = \bigwedge_{s \in A} (\neg s^{\mathbf{t}} \vee \neg s^{\mathbf{f}})$$

encodes that J is a valid interpretation by setting the constraint that both $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ cannot be set to true, and

$$\phi_{<_i}(I) = \bigwedge_{I(s)=\mathbf{t}} s^{\mathbf{t}} \wedge \bigwedge_{I(s)=\mathbf{f}} s^{\mathbf{f}} \wedge \bigvee_{I(s)=\mathbf{u}} (s^{\mathbf{t}} \vee s^{\mathbf{f}})$$

encodes $I <_i J$, stating that for all $s \in A$ with $I(s) = \mathbf{t}$, we must set $s^{\mathbf{t}}$ to true, similarly for $I(s) = \mathbf{f}$ and $s^{\mathbf{f}}$, and for some $s \in A$ with $I(s) = \mathbf{u}$, we must set either $s^{\mathbf{t}}$ or $s^{\mathbf{f}}$ to true.

To express that, for each $s \in A$, we have $J_s \geq_i J$ for the two-valued interpretation J_s represented by $\varphi_s^{\downarrow}$, we use the formula

$$\phi_s^{pr}(D) = \left(s^{\mathbf{t}} \rightarrow \bigwedge_{(s,r)\in L} p_r^s\right) \wedge \left(s^{\mathbf{f}} \rightarrow \bigwedge_{(s,r)\in L} \neg p_r^s\right)$$

which propagates the assignments to $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ (representing an interpretation J) to all p_r^s (representing two-valued interpretations J_s, $s \in A$). Further, we need to express that $J(s) = \mathbf{t}$ implies that $\varphi_s[J]$ is satisfiable, and that $J(s) = \mathbf{f}$ implies that $\varphi_s[J]$ is refutable. This is achieved for each $s \in A$ by the formula

$$\phi_s^{\Gamma}(D) = (s^{\mathbf{t}} \rightarrow \varphi_s^{\downarrow}) \wedge (s^{\mathbf{f}} \rightarrow \neg\varphi_s^{\downarrow}).$$

Now, the encoding for the decision problem $Exists_{cf}^{>}$ is formed by conjoining the formulas presented so far as

$$\Phi_{cf}(D, I) = \phi(D) \wedge \phi_{<_i}(I) \wedge \bigwedge_{s\in A} (\phi_s^{\Gamma}(D) \wedge \phi_s^{pr}(D)).$$

We have that $\Phi_{cf}(D, I)$ is satisfiable if and only if there exists an interpretation $J \in cf(D)$, $J >_i I$, represented by a satisfying truth assignment τ to the $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ variables via

$$J(s) = \begin{cases} \mathbf{t}, & \tau(s^{\mathbf{t}}) = 1, \\ \mathbf{f}, & \tau(s^{\mathbf{f}}) = 1, \\ \mathbf{u}, & \text{otherwise.} \end{cases}$$

This procedure allows us, given an ADF and an interpretation, to extract a strictly more informative conflict-free interpretation or to prove that none exists.

For general ADFs, checking admissibility of an interpretation is in coNP and can similarly be solved by employing a SAT solver by encoding the complement of the problem. To this end, for each $s \in A$, in addition to the $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ variables, we declare Boolean variables p^s representing a guess of an interpretation $J \geq_i I$ that acts as a witness for $I \notin adm(D)$. Define $\varphi_s^{\uparrow} = \varphi_s[r \mapsto p^r \mid (r, s) \in L]$. Now, the complement $\neg Ver_{adm}$ can be expressed as a formula

$$\Phi_{\neg Ver_{adm}}(D, I) = \bigwedge_{I(s)=\mathbf{t}} p^s \wedge \bigwedge_{I(s)=\mathbf{f}} \neg p^s \wedge \left(\bigvee_{I(s)=\mathbf{t}} \neg\varphi_s^{\uparrow} \vee \bigvee_{I(s)=\mathbf{f}} \varphi_s^{\uparrow}\right)$$

which is satisfiable if and only if $I(s) = \mathbf{t}$ but $\varphi_s[J]$ is refutable (then $\neg\varphi_s^{\uparrow}$ is true), or $I(s) = \mathbf{f}$ but $\varphi_s[J]$ is satisfiable (then $\varphi_s^{\uparrow}$ is true) for some $s \in A$. That is, $\Phi_{\neg Ver_{adm}}(D, I)$ is satisfiable iff $I \notin adm(D)$.

We move on to bipolar ADFs, for which the problem $Exists_{adm}^{>}$ is in NP. For a bipolar ADF D with attacking links L_D^- and supporting links L_D^+, we define $L_D^{\ominus} = L_D^- \setminus L_D^+$ and $L_D^{\oplus} = L_D^+ \setminus L_D^-$. In order to encode $Exists_{adm}^{>}$ to SAT, we need to encode the constraint $J \leq_i \Gamma_D(J)$ for the witnessing interpretation J, which requires encoding the operator Γ_D. To this end, we recall that $\Gamma_D(J)(s)$ can be computed in polynomial time for bipolar ADFs [69] by considering two interpretations $J_s^{(1)}$ and $J_s^{(2)}$ over the parents of s, defined as

$$J_s^{(1)}(r) = \begin{cases} \mathbf{t}, & \text{if } J(r) = \mathbf{t}, \text{ or } J(r) = \mathbf{u} \text{ and } (r, s) \in L_D^+, \\ \mathbf{f}, & \text{if } J(r) = \mathbf{f}, \text{ or } J(r) = \mathbf{u} \text{ and } (r, s) \in L_D^{\ominus}, \end{cases}$$

$$J_s^{(2)}(r) = \begin{cases} \mathbf{t}, & \text{if } J(r) = \mathbf{t}, \text{ or } J(r) = \mathbf{u} \text{ and } (r, s) \in L_D^-, \\ \mathbf{f}, & \text{if } J(r) = \mathbf{f}, \text{ or } J(r) = \mathbf{u} \text{ and } (r, s) \in L_D^{\oplus}, \end{cases}$$

and finally setting

$$\Gamma_D(J)(s) = \begin{cases} \mathbf{t}, & \text{if } J_s^{(2)}(\varphi_s) = \mathbf{t}, \\ \mathbf{f}, & \text{if } J_s^{(1)}(\varphi_s) = \mathbf{f}, \\ \mathbf{u}, & \text{otherwise.} \end{cases}$$

In our approach, this procedure is encoded as a propositional formula

$$\phi_{bip,adm}(D) = \bigwedge_{s\in A}\left(\left(s^{\mathbf{t}} \rightarrow \left(\bigwedge_{(r,s)\in L_D^{\ominus}} (\neg r^{\mathbf{f}} \rightarrow p_s^r) \wedge \bigwedge_{(r,s)\in L_D^{\oplus}} (\neg r^{\mathbf{t}} \rightarrow \neg p_s^r)\right)\right) \wedge \left(s^{\mathbf{f}} \rightarrow \left(\bigwedge_{(r,s)\in L_D^{\ominus}} (\neg r^{\mathbf{t}} \rightarrow \neg p_s^r) \wedge \bigwedge_{(r,s)\in L_D^{\oplus}} (\neg r^{\mathbf{f}} \rightarrow p_s^r)\right)\right)\right)$$

which captures admissibility for bipolar ADFs by requiring that if we set $J(s) = \mathbf{t}$, we need to set the values of parents in φ_s according to $J_s^{(2)}$, and if we set $J(s) = \mathbf{f}$, we must set the values of parents in φ_s according to $J_s^{(1)}$. This encodes $J \leq_i \Gamma_D(J)$, ensuring that J is admissible. Similarly as $Exists_{cf}^{>}$ for general ADFs, $Exists_{adm}^{>}$ for bipolar ADFs can be solved via the encoding

$$\Phi_{adm,bip}(D, I) = \phi(D) \wedge \phi_{<_i}(I) \wedge \phi_{bip,adm}(D) \wedge \bigwedge_{s\in A}(\phi_s^{\Gamma}(D) \wedge \phi_s^{pr}(D))$$

which is satisfiable if and only if there exists an interpretation $J \in adm(D)$ with $J >_i I$.

We can also capture complete interpretations for bipolar ADFs using a similar reasoning. Namely, for complete interpretations we need to encode $J \geq_i \Gamma_D(J)$ in addition to $J \leq_i \Gamma_D(J)$ (encoded by $\phi_{bip,adm}(D)$), which in turn shall imply $J = \Gamma_D(J)$. To this end, we declare additional variables $p_{r,s}^{(i)}$ for each $(r, s) \in L$, $i = 1, 2$, which will (similarly to p_s^r variables) correspond to interpretations $J_s^{(i)}$. We propagate the assignments on $s^{\mathbf{t}}$ and $s^{\mathbf{f}}$ (similarly to $\phi_s^{pr}(D)$) via

$$\phi_s^{pr(1,2)}(D) = \left(s^{\mathbf{t}} \rightarrow \bigwedge_{(s,r)\in L_D} (p_{s,r}^{(1)} \wedge p_{s,r}^{(2)})\right) \wedge \left(s^{\mathbf{f}} \rightarrow \bigwedge_{(s,r)\in L_D} (\neg p_{s,r}^{(1)} \wedge \neg p_{s,r}^{(2)})\right),$$

and make sure that we assign values to $p_{r,s}^{(i)}$ values exactly according to the definitions of $J_s^{(i)}$ by making use of the formula

$$\phi_s^{(1,2)}(D) = \bigwedge_{(r,s)\in L_D^{\ominus}} (\neg r^{\mathbf{f}} \rightarrow p_{r,s}^{(2)}) \wedge \bigwedge_{(r,s)\in L_D^{\oplus}} (\neg r^{\mathbf{t}} \rightarrow \neg p_{r,s}^{(2)}) \wedge \bigwedge_{(r,s)\in L_D^{\ominus}} (\neg r^{\mathbf{t}} \rightarrow \neg p_{r,s}^{(1)}) \wedge \bigwedge_{(r,s)\in L_D^{\oplus}} (\neg r^{\mathbf{f}} \rightarrow p_{r,s}^{(1)}).$$

Denoting by $\varphi_s^{(i)}$ the acceptance condition of $s \in A$, where each occurrence of $r \in par_D(s)$ is replaced with $p_{r,s}^{(i)}$, $i = 1, 2$, it remains to set the constraints that if $J_2(\varphi_s) = \mathbf{t}$, then we must set $J(s) = \mathbf{t}$, and if $J_1(\varphi_s) = \mathbf{f}$, we must set $J(s) = \mathbf{f}$. This is encoded as

$$\phi_{bip,com}(D) = \phi_{bip,adm}(D) \wedge \bigwedge_{s\in A}\left(\phi_s^{pr(1,2)}(D) \wedge \phi_s^{(1,2)}(D) \wedge (\varphi_s^{(2)} \rightarrow s^{\mathbf{t}}) \wedge (\neg\varphi_s^{(1)} \rightarrow s^{\mathbf{f}})\right).$$

Now, for bipolar ADFs,

$$\Phi_{com,bip}(D, I) = \phi(D) \wedge \phi_{<_i}(I) \wedge \phi_{bip,com}(D) \wedge \bigwedge_{s\in A}(\phi_s^{\Gamma}(D) \wedge \phi_s^{pr}(D))$$

is satisfiable if and only if there is an interpretation $J \in com(D)$ with $J >_i I$.

Finally, we consider k-bipolar ADFs, for which we develop encodings that are exponential in k. By Theorem 4.11, $Exists_{adm}^{>}$ is in NP also for k-bipolar ADFs. For $s \in A$, the values of the parents $r \in par_D(s)$ connected through non-bipolar links—up to k and identified by p_s^r—are not determined by $\phi_{bip,adm}(D)$, which encodes $J \leq_i \Gamma_D(J)$ for bipolar ADFs. Instead, all completions I_X of those r with $J(r) = \mathbf{u}$ must be checked in order to ensure that $\varphi_s[J]$ is a tautology (resp. unsatisfiable). For each $s \in A$, let

$$X_s = \{r \mid \exists (r, s) \in L_D^?, I(r) = \mathbf{u}\}$$

and $\mathcal{V}(X_s)$ be the set of two-valued interpretations over X_s. For $I_X \in \mathcal{V}(X_s)$, let

$$\varphi_s^{\downarrow, I_X} = \varphi_s[r \mapsto p_s^r \mid (r, s) \in L, r \notin X_s][r \mapsto I_X(r) \mid (r, s) \in L, r \in X_s].$$

For each $s \in A$, we make use of propositional formulas

$$\phi_s^{\Gamma_?}(D) = \left(s^{\mathbf{t}} \rightarrow \left(\bigwedge_{I_X\in\mathcal{V}(X_s)} \left(\varphi_s^{\downarrow, I_X} \vee \bigvee_{I_X(r)=\mathbf{t}} r^{\mathbf{f}} \vee \bigvee_{I_X(r)=\mathbf{f}} r^{\mathbf{t}}\right)\right)\right) \wedge \left(s^{\mathbf{f}} \rightarrow \left(\bigwedge_{I_X\in\mathcal{V}(X_s)} \left(\neg\varphi_s^{\downarrow, I_X} \vee \bigvee_{I_X(r)=\mathbf{t}} r^{\mathbf{f}} \vee \bigvee_{I_X(r)=\mathbf{f}} r^{\mathbf{t}}\right)\right)\right)$$

which guarantee that if we set $J(s) = \mathbf{t}$ (resp. $J(s) = \mathbf{f}$), all completions I_X of parents r with $J(r) = \mathbf{u}$ satisfy (resp. refute) $\varphi_s^{\downarrow, I_X}$. This check can be disregarded if the values of $I_X(r)$ and $J(r)$ differ for some non-bipolar parent r, which is achieved by disjunctions in $\phi_s^{\Gamma_?}$. For $\sigma \in \{adm, com\}$, define

$$\Phi_{\sigma,kbip}(D, I) = \phi(D) \wedge \phi_{<_i}(I) \wedge \phi_{bip,\sigma}(D) \wedge \bigwedge_{s \in A} (\phi_s^{\Gamma_k}(D) \wedge \phi_s^{pr}(D)),$$

which differs from $\Phi_{\sigma,bip}(D, I)$ by $\phi_s^{\Gamma_k}(D)$, which is equal to $\phi_s^{\Gamma}(D)$ if s has only bipolar incoming links, and equal to $\phi_s^{\Gamma_?}(D)$ otherwise. We have that for k-bipolar ADFs, $\Phi_{\sigma,kbip}(D, I)$ is satisfiable if and only if there exists an interpretation $J \in \sigma(D)$ with $J >_i I$.

Algorithm 1 Procedure for $Cred_{adm}(D, \alpha)$ with general ADF $D = (A, L, C)$, query $\alpha \in A$.

```
1: φ ← Φ_cf(D, I_u) ∧ α^t
2: while true do
3:     (τ, sat) ← SAT(φ)
4:     if not sat then return reject
5:     I ← INTERPRETATION(τ)
6:     (τ, sat) ← SAT(Φ_{¬Ver_adm}(D, I))
7:     if not sat then return accept
8:     φ ← φ ∧ REFINE_≠(I)
```

5.2. Algorithms

We continue by presenting concrete decision procedures for the acceptance problems of interest, i.e., $Cred_{adm} = Cred_{com} = Cred_{prf}$ and $Skept_{prf}$. Algorithm 1 solves $Cred_{adm}$ for general ADFs (a Σ_2^p-complete problem). We begin by initializing a formula φ with a suitable subproblem—in this case, credulous acceptance under conflict-free semantics (line 1). Then, we iteratively call a SAT solver on φ (line 3). If the solver reports φ to be unsatisfiable, we may immediately *reject* (line 4). Otherwise, we extract from the satisfying truth assignment τ a conflict-free interpretation I with $I(\alpha) = \mathbf{t}$ (line 5). We check using another SAT solver whether I is not admissible (line 6)—if it is, i.e. the SAT call is unsatisfiable, we may *accept* as I is a valid witness to credulous acceptance of α (line 7). Otherwise, we refine φ by excluding I out of further consideration using the clause

$$Refine_{\neq}(I) = \bigvee_{I(s)=\mathbf{t}} \neg s^{\mathbf{t}} \vee \bigvee_{I(s)=\mathbf{f}} \neg s^{\mathbf{f}} \vee \bigvee_{I(s)=\mathbf{u}} (s^{\mathbf{t}} \vee s^{\mathbf{f}}).$$

For k-bipolar ADFs, $Cred_{adm}$ (an NP-complete problem) can be decided via a single SAT call using

$$\Phi_{adm,kbip}(D, I_{\mathbf{u}}) \wedge \alpha^{\mathbf{t}}.$$

Algorithm 2 Procedure for $Skept_{prf}(D, \alpha)$ with k-bipolar ADF $D = (A, L, C)$, query $\alpha \in A$.

```
1: χ ← adm  or  χ ← com
2: φ ← Φ_{χ,kbip}(D, I_u) ∧ ¬α^t
3: while true do
4:     (τ, sat) ← SAT(φ)
5:     if not sat then return accept
6:     I ← INTERPRETATION(τ)
7:     while true do
8:         (τ, sat) ← SAT(Φ_{χ,kbip}(D, I) ∧ ¬α^t)
9:         if not sat then break
10:        I ← INTERPRETATION(τ)
11:    (τ, sat) ← SAT(Φ_{χ,kbip}(D, I) ∧ α^t)
12:    if not sat then return reject
13:    φ ← φ ∧ REFINE_>(I)
```

We continue by describing a procedure for $Skept_{prf}$ for k-bipolar ADFs (a Π_2^p-complete problem), outlined in Algorithm 2. This time, we may use $\Phi_{\chi,kbip} \wedge \neg\alpha^{\mathbf{t}}$ with $\chi \in \{adm, com\}$ as a suitable initial abstraction (lines 1 and 2). That is, we consider admissible (resp. complete) interpretations which do not assign the query to true, and are thus counterexample candidates to skeptical acceptance of α under preferred. We iteratively call a SAT solver on φ (line 4), and accept if φ turns out unsatisfiable (line 5). Otherwise, we extract the admissible (resp. complete) interpretation I with $I(\alpha) \neq \mathbf{t}$ from the satisfying truth assignment τ (line 6). Then, we iteratively maximize I w.r.t. the information order (lines 7–10) under the constraint $I(\alpha) \neq \mathbf{t}$, yielding an information-maximal admissible (resp. complete) interpretation not assigning α to true. We check whether assigning α to true results in a strictly more informative interpretation (line 11). If not, I is a preferred

interpretation and a counterexample to skeptical acceptance, so we reject (line 12). Otherwise, we refine φ (line 13) by excluding interpretations that are at least as informative as I via

$$\mathit{Refine}_{>}(I) = \bigvee_{I(s)=\mathbf{t}} s^{\mathbf{f}} \vee \bigvee_{I(s)=\mathbf{f}} s^{\mathbf{t}} \vee \bigvee_{I(s)=\mathbf{u}} (s^{\mathbf{t}} \vee s^{\mathbf{f}}).$$

By Theorem 4.18 the number of iterations in Algorithm 2 is polynomial in k for k-*prf*-concise ADFs, that is, the number of iterations is polynomially bounded by the number of interpretations.

Algorithm 3 Procedure for $\mathit{Skept}_{prf}(D, \alpha)$ with general ADF $D = (A, L, C)$, query $\alpha \in A$.

```
1: φ ← Φ_cf(D, I_u) ∧ ¬α^t
2: while true do
3:     (τ, sat) ← SAT(φ)
4:     if not sat then return accept
5:     I ← INTERPRETATION(τ)
6:     (τ, sat) ← SAT(Φ_¬Ver_adm(D, I))
7:     if sat then
8:         φ ← φ ∧ REFINE_≠(I)
9:         continue
10:    ψ ← Φ_cf(D, I) ∧ ¬α^t
11:    while true do
12:        (τ, sat) ← SAT(ψ)
13:        if not sat then break
14:        J ← INTERPRETATION(τ)
15:        (τ, sat) ← SAT(Φ_¬Ver_adm(D, J))
16:        if not sat then
17:            I ← J
18:            ψ ← ψ ∧ φ_<i(I)
19:        else ψ ← ψ ∧ REFINE_≠(I)
20:    ψ ← (ψ \ {¬α^t}) ∧ α^t
21:    while true do
22:        (τ, sat) ← SAT(ψ)
23:        if not sat then return reject
24:        J ← INTERPRETATION(τ)
25:        (τ, sat) ← SAT(Φ_¬Ver_adm(D, J))
26:        if not sat then break
27:        else ψ ← ψ ∧ REFINE_≠(I)
28:    φ ← φ ∧ REFINE_>(I)
```

For general ADFs, Skept_{prf} is tricky to solve as a Π_3^p-complete problem, since even Cred_{adm}—used as an abstraction in Algorithm 2—is already Σ_2^p-complete. We may, however, essentially replace every SAT call in Algorithm 2 with a call to a procedure for $\mathit{Exists}^{>}_{adm}(D, I)$. So, instead of a single SAT call, we iteratively

1. get a conflict-free interpretation J using $\Phi_{cf}(D, I)$, and
2. check whether it is admissible using $\Phi_{\neg Ver_{adm}}(D, I)$,
   - if it is (the call is unsatisfiable), we continue as in Algorithm 2,
   - if it is not (the call is satisfiable), we exclude it out of the search space using $\textsc{Refine}_{\neq}(J)$.

Algorithm 3 implements this idea. Similarly as in Algorithm 2, our goal is to find a preferred interpretation acting as a counterexample to skeptical acceptance, i.e., an interpretation $I \in \mathit{prf}(D)$ with $I(\alpha) \neq \mathbf{t}$. In more detail, we begin by initializing the abstraction φ which encodes the set of conflict-free interpretations which do not set the query argument α to true via $\Phi_{cf}(D, I_{\mathbf{u}}) \wedge \neg\alpha^{\mathbf{t}}$ (line 1). Then, we iteratively call a SAT solver on the abstraction φ (line 3), and accept if φ is unsatisfiable (line 4). Otherwise, we extract the conflict-free interpretation I with $I(\alpha) \neq \mathbf{t}$ from the truth assignment τ (line 5). Now, we need to check whether I is admissible using $\Phi_{\neg Ver_{adm}}(D, I)$ (line 6)—if this formula is satisfiable, I is not admissible, so we refine the abstraction φ by excluding the current interpretation I (lines 7–9), and continue with the next iteration. Otherwise, we know that I is admissible.

At this point, our goal is to maximize I w.r.t. the information order under the constraint $I(\alpha) \neq \mathbf{t}$ similarly as in Algorithm 2, but this time we need to check for admissibility at each iteration. To implement this maximization procedure, we need to make sure that adding new formulas $\phi_{<_i}(I)$ do not affect the original abstraction φ. To this end, we initialize a new abstraction ψ as $\Phi_{cf}(D, I) \wedge \neg\alpha^{\mathbf{t}}$, encoding conflict-free interpretations $J >_i I$ with $J(\alpha) \neq \mathbf{t}$ (line 10). The maximization procedure (lines 11–19) proceeds as follows. We iteratively call a SAT solver on the abstraction ψ (line 12). If the abstraction is unsatisfiable, we know that there are no more informative interpretations, so we break from this inner loop (line 13). Otherwise, we extract the more informative conflict-free interpretation $J >_i I$ from the truth assignment τ (line 14), and check whether it is admissible using $\Phi_{\neg Ver_{adm}}(D, J)$ (line 15). If this formula is unsatisfiable, J is admissible, so we update the current interpretation I to J and the abstraction ψ by adding the formula $\phi_{<_i}(I)$ asking for a strictly more informative

interpretation (lines 16–18). If, however, the formula is satisfiable, J is not admissible, so we refine the abstraction ψ by excluding the interpretation J (line 19).

Now we know that I is an information-maximal admissible interpretation satisfying the constraint $I(\alpha) \neq \mathbf{t}$. The final step is to check whether I is preferred. To this end, we instead try setting $I(\alpha) = \mathbf{t}$ by using the literal $\alpha^{\mathbf{t}}$ instead of $\neg\alpha^{\mathbf{t}}$ in the abstraction ψ (line 20), and check whether this way we can further maximize the interpretation I (lines 21–27). We iteratively call a SAT solver on the modified abstraction ψ (line 22). If ψ is unsatisfiable, I is in fact preferred, and thus a counterexample to skeptical acceptance of the query α, so we reject (line 23). Otherwise, we extract an interpretation J from the truth assignment τ (line 24), and check whether it is admissible via $\Phi_{\neg Ver_{adm}}(D, J)$ (line 25). If this formula is unsatisfiable, J is an admissible interpretation which is strictly more informative than I, meaning that I is not preferred, so we break from this inner loop (line 26) and refine the original abstraction φ by excluding interpretations that are at least as informative as I (line 28), and continue with the next iteration of the outer loop. If the formula is satisfiable, J is not admissible, so we exclude it from further consideration (line 27).

6. Implementation and experiments

This section is devoted to the presentation of the implementation of a system employing the algorithms described in the previous section, and of the related experimental analysis in comparison to rival systems, in two separate sub-sections.

6.1. Implementation

The system k++ADF is implemented in the C++ programming language, and is available in open source at:

https://bitbucket.org/andreasniskanen/k-adf

and includes MiniSAT (version 2.2.0) as the SAT solver.

Input format. As input, the system requires an ADF instance with the following syntax:

```
s(a).      # a is an argument
ac(a,f).   # the acceptance condition of a is f
```

where `f` is a propositional formula, defined recursively via:

1. all arguments `a`, the truth constant `c(v)`, and the falsity constant `c(f)` are formulas,
2. all strings of the form `neg(p)`, `and(p,q)`, `or(p,q)`, `xor(p,q)`, `imp(p,q)`, and `iff(p,q)`, where `p` and `q` are subformulas, are formulas.

The predicates `neg`, `and`, `or`, `xor`, `imp`, `iff` stand for negation, conjunction, disjunction, exclusive disjunction, implication, and equivalence, respectively. This input format is used by various other ADF reasoning systems, including those evaluated in this paper (apart from some minor syntactic differences, and possibly in restricted forms).

Usage and options. After compilation, k++ADF is called from the command line as:

```
./k++adf <sem> [options] <file>
```

where

- `<sem>` is an ADF semantics: `cf`, `nai`, `adm`, `com`, `prf`, `grd` stand for conflict-free, naive, admissible, complete, preferred, and grounded, respectively.
- `<file>` is the input ADF file.

The system defaults to the enumeration of interpretations unless an option for argument acceptance is specified. The following options may be used:

- `-a <arg>`: Query argument for acceptance problems.
- `-c`: Credulous acceptance. Requires the `-a` option.
- `-s`: Skeptical acceptance. Requires the `-a` option.
- `-l`: Output link types using `att` and `sup` predicates.
- `-f`: Do not use encodings for k-bipolarity.
- `-n`: Do not use shortcuts for skeptical acceptance.
- `-m`: Use complete semantics as an abstraction for preferred semantics.
- `-h` and `-v`: For help and version number.

Program flow. If an input ADF is provided, k++ADF parses it and stores the arguments and the acceptance conditions initially as strings. As the input format for acceptance conditions consists of binary predicates along with constants for arguments, truth, and falsity, and a unary predicate for negation, a natural way to represent a formula is a binary tree where non-leaf nodes are (possibly negated) connectives (OR, AND, XOR—implication and equivalence are simply shorthands), and leaves are (possibly negated) arguments or true/false.

After constructing the binary tree and the links of the ADF, k++ADF proceeds by translating the acceptance conditions to CNF via the Tseitin transformation [71], which in the context of SAT solving is a standard method for converting propositional formulas to CNF. Variables in CNF are positive integers. We begin by mapping truth and falsity to specific variables reserved for them, and arguments s to $s^{\mathbf{t}}$, $s^{\mathbf{f}}$, p^s, and links to p^r_s variables. We also map each non-leaf node to a fresh variable, and each node to a literal which is positive (i.e. the variable) if the node is not negated, and negative (i.e., the negation of the variable) otherwise. Then, we iterate through all acceptance conditions, generating a CNF formula recursively as follows, starting at the root:

1. If we are at a leaf node, we return nothing.
2. Otherwise, this is a node corresponding to a connective. Let x be the variable corresponding to it, y be the literal corresponding to its left child, and z be the literal corresponding to its right child. Initialize an empty container for clauses:
   - If the connective is AND, add clauses $(x \vee \neg y \vee \neg z) \wedge (\neg x \vee y) \wedge (\neg x \vee z)$.
   - If the connective is OR, add clauses $(\neg x \vee y \vee z) \wedge (x \vee \neg y) \wedge (x \vee \neg z)$.
   - Otherwise, add clauses $(\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z)$ corresponding to a XOR.
3. Recursively call this procedure for the left child and the right child and add the corresponding clauses to the container, and return the container.

Now, adding all these CNF formulas to a SAT solver, we have immediate access to $\varphi_s^{\downarrow}$ (resp. $\varphi_s^{\uparrow}$) via the literal corresponding to the root node.

If $\sigma \in \{adm, com, prf\}$, we continue by computing the polarity (attacking or supporting) for each link (unless option `-f` is used to disable k-bipolar encodings). To this end, we make use of the fact that the formula

$$\varphi_s^{\uparrow} \wedge \neg\varphi_s^{\neg,r} \wedge \neg p^r$$

is unsatisfiable iff the link (r, s) is attacking, and the formula

$$\neg\varphi_s^{\uparrow} \wedge \varphi_s^{\neg,r} \wedge \neg p^r$$

is unsatisfiable iff the link (r, s) is supporting, where

$$\varphi_s^{\neg,r} = \varphi_s[t \mapsto \neg p^t \mid (t, s) \in L, t \neq r][r \mapsto p^r].$$

Hence, computing the polarities amounts to $2 \cdot |L|$ SAT solver calls that are of linear size

Reasoning tasks. The system k++ADF implements the enumeration of interpretations (the default reasoning mode), and credulous and skeptical acceptance for conflict-free, naive, admissible, complete, preferred, and grounded semantics. The algorithms used for credulous acceptance under admissible, complete, and preferred, and skeptical acceptance under preferred, are covered in Section 5. Each of these has both a k-bipolar version, and a general version. The system defaults to the general version if k is detected to be larger than 30, or with the `-n` option. For credulous acceptance under conflict-free and naive semantics, we may use a direct SAT call similarly as for credulous acceptance under admissible for k-bipolar ADFs. Skeptical acceptance under conflict-free and admissible is a trivial *reject*, and for skeptical acceptance under naive, we use an iterative procedure similar to skeptical acceptance under preferred for k-bipolar ADFs.

For (credulous and skeptical) acceptance under grounded, which coincides to skeptical acceptance under complete, we simply compute the unique grounded interpretation and check the value of the query argument. This is achieved via an iterative SAT-based procedure that computes the characteristic operator until reaching the fixpoint:

1. For each argument s, add clauses $\varphi_s^{\uparrow}$ (with p^r variables for each parent r) to the solver.
2. Initialize an interpretation as $I = I_{\mathbf{u}}$.
3. Initialize another interpretation J, setting for each argument s:
   - $J(s) = \mathbf{t}$ iff solver returns UNSAT under assumption $\neg\varphi_s^{\uparrow}$ ($\varphi_s[I]$ is irrefutable),
   - $J(s) = \mathbf{f}$ iff solver returns UNSAT under assumption $\varphi_s^{\uparrow}$ ($\varphi_s[I]$ is unsatisfiable),
   - $J(s) = \mathbf{u}$ otherwise.
4. If we reached a fixpoint ($J = I$), return I.
5. For each argument, add a clause $s^{\mathbf{t}}$ if $J(s) = \mathbf{t}$, and a clause $s^{\mathbf{f}}$ if $J(s) = \mathbf{f}$.
6. Set $I = J$ and go to step 3.

Enumeration of interpretations is accomplished by iterative procedures which add blocking clauses of the form $\text{REFINE}_{\neq}(I)$ (or in the case of preferred and naive, a stronger $\text{REFINE}_{>}(I)$) at each iteration, making use of the encodings presented in Section 5.

Shortcuts. We know that every preferred interpretation is at least as informative as the grounded interpretation. Therefore, for reasoning under preferred semantics, we may compute the grounded interpretation G, and in case of acceptance, immediately accept if $G(\alpha) = \mathbf{t}$ or immediately reject if $G(\alpha) = \mathbf{f}$ for query argument α. Otherwise, we may use G instead of $I_{\mathbf{u}}$ as the least informative interpretation. These shortcuts are disabled via the `-n` option.

For the k-bipolar procedure for skeptical acceptance under preferred semantics (Algorithm 2), there are two options for the base semantics: admissible or complete semantics. By default, the choice is admissible. The user may switch to complete semantics using option `-com`.

6.2. Experiments

We continue by presenting an overview of the scalability of the approach, along with a comparison to alternative ADF systems, and between different configurations of k++ADF. For the experiments we used 2.4GHz Intel Xeon E5-2680 v4 14-core machines with 256GB RAM running CentOS 7, setting a 1800-second timeout limit and a 64-GB memory limit per instance.

We compared the performance of k++ADF with goDiamond 0.6.6 [68] with clingo 5.4.0 [40], QADF 0.4.0 [31] with bloqqer 037 [11] and DepQBF 6.03 [52], and YADF 0.1.1 [14] with lpopt 2.0 [10] and clingo 5.4.0. Briefly put, goDiamond is an implementation based on answer set programming (ASP) encodings which explicitly construct the truth table representation of each acceptance condition. On the other hand, QADF constructs encodings as quantified Boolean formulas (QBFs), allowing for solving acceptance problems with a single QBF solver call. Finally, YADF is also an ASP-based approach, but generates so-called dynamic encodings as opposed to static encodings used by goDiamond.

We considered the problems Cred_{adm} and Skept_{prf}, along with Skept_{nai}, which is currently not supported by other solvers. Furthermore, we considered different configurations of k++ADF. For the task Cred_{adm}, we consider configurations (*adm*)—a direct SAT call using the k-bipolar encoding—and (*cf*)—a SAT-based second-level CEGAR procedure implementing Algorithm 1. For the task Skept_{prf}, we consider the following configurations:

- (*com*): second-level CEGAR procedure implementing Algorithm 2 with complete as the base semantics,
- (*adm*): second-level CEGAR procedure implementing Algorithm 2 with admissible as the base semantics,
- (*cf* + *grd*): third-level CEGAR procedure implementing Algorithm 3, which begins by precomputing and assuming the grounded interpretation of the input ADF,
- (*cf*): third-level CEGAR procedure implementing Algorithm 3 without this shortcut.

For the first set of benchmarks, we selected the most challenging instances from the YADF system page.[4] These ADFs were generated from the AFs of domains ABA2AF, Planning, and Traffic from ICCMA'17[5] as follows [14]. Given an AF graph, acceptance conditions are constructed by splitting parents into 5 groups, each constituting a subformula representing attack, group-attack, support, group-support, and exclusive-or. Each parent is assigned to one of these groups with equal probability. In more detail, if parents $s_1, \ldots, s_n$ are chosen into a group representing

- attack, the subformula is $\neg s_1 \wedge \cdots \wedge \neg s_n$;
- group-attack, the subformula is $\neg s_1 \vee \cdots \vee \neg s_n$;
- support, the subformula is $s_1 \vee \cdots \vee s_n$;
- group-support, the subformula is $s_1 \wedge \cdots \wedge s_n$;
- exclusive-or, the subformula is $l_1 \oplus \cdots \oplus l_n$, where l_i is either s_i or $\neg s_i$ with equal probability.

The subformulas are connected via $\wedge$ or $\vee$ with equal probability, resulting in 300 ADFs (10 to 150 arguments for ABA2AF; 10 to 300 arguments else). Query arguments were selected by sampling uniformly at random from the set of all arguments. For the second set of benchmarks, in order to generate more challenging ADF instances, we used exactly the same instance generation model, and generated 276 ADFs from ICCMA'19[6] benchmark AFs with at most 1000 arguments.

All of the instances turned out to be relatively easy for our approach to solving Skept_{nai}. For the domains ABA2AF, Planning, and Traffic, the mean running times were 24 seconds, 146 seconds, and 1.8 seconds, with one, six, and zero timeouts, respectively. For the ICCMA'19 benchmark set, we had three timeouts with a mean running time of 35 seconds. Somewhat interestingly, for this problem the Planning instances turned out to be the hardest to solve.

Table 5 presents an overview of the mean running times (with timeouts included as the timeout limit of 1800 seconds) and the number of timeouts on the task Cred_{adm}, for each domain and solver considered. Clearly, on domains Planning and

[4] http://www.dbai.tuwien.ac.at/proj/adf/yadf/.
[5] http://argumentationcompetition.org/2017/.
[6] http://argumentationcompetition.org/2019/.

Table 5
Mean running times for $Cred_{adm}$ (timeouts included as the timeout limit of 1800s). Number of timeouts (out of 100 instances for ABA2AF, Planning, and Traffic; out of 276 instances for ICCMA'19) in brackets.

Domain	k++ADF (*adm*)	k++ADF (*cf*)	goDiamond	QADF	YADF
ABA2AF	247.03 (13)	**202.79** (**10**)	948.58 (51)	854.06 (47)	983.20 (53)
Planning	**0.07** (**0**)	254.64 (14)	3.41 (**0**)	1553.36 (86)	10.63 (**0**)
Traffic	**0.03** (**0**)	180.32 (10)	3.27 (**0**)	510.18 (28)	40.92 (2)
ICCMA'19	**535.18** (**74**)	1453.97 (222)	1285.89 (191)	1780.58 (273)	1608.01 (244)

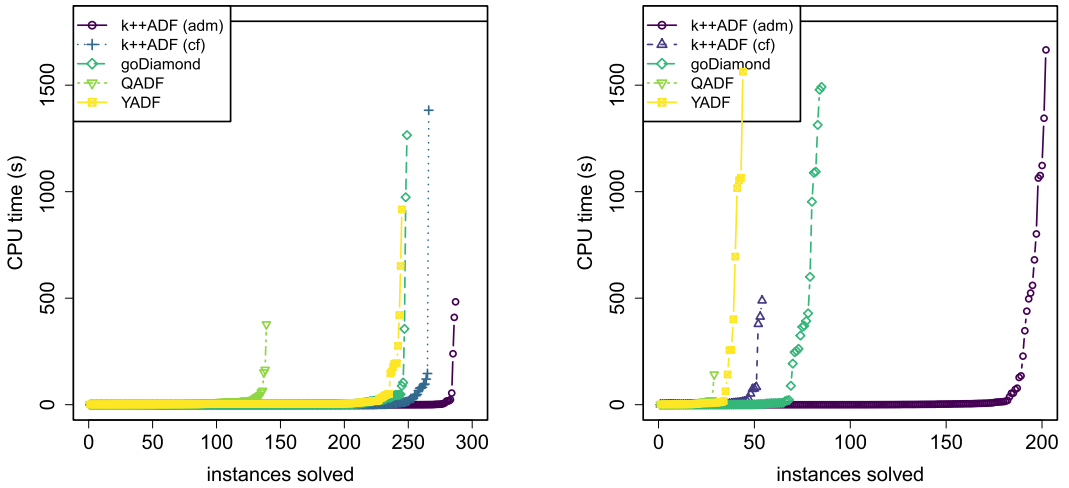

Fig. 12. Solver comparison for $Cred_{adm}$: number of solved instances (on the x-axis) given a per-instance time limit (on the y-axis) for ICCMA'17 (ABA2AF, Planning, Traffic; left) and ICCMA'19 (right) instances.

Traffic k++ADF (*adm*), which is a direct SAT call, is the dominant approach, with zero timeouts and a mean running time under 0.1 seconds. These instances are also very easy for goDiamond, also exhibiting zero timeouts. This is not the case for ICCMA'19 instances, on which k++ADF (*adm*) clearly has the lowest number of timeouts. Somewhat surprisingly, on ABA2AF using k++ADF (*adm*) results in a few more timeouts as compared to k++ADF (*cf*). Fig. 12 further visualizes these results by showing the number of solved instances (on the x-axis) given a per-instance time limit (on the y-axis). In particular, we observe that the ICCMA'19 instances are considerably harder to solve for all other approaches except for k++ADF (*adm*).

Similarly for the task $Skept_{prf}$, Table 6 provides an overview of the mean running times and the number of timeouts. Here again, on domains Planning and Traffic k++ADF (*com*), k++ADF (*adm*), and goDiamond produce zero timeouts, thus constituting the best approaches for these instances out of the solvers considered. The mean running times are slightly lower using k++ADF (*adm*) than the rest of the solvers. However, on domains ABA2AF and ICCMA'19, k++ADF (*com*) is clearly the dominant approach both in terms of mean running times and the number of solved instances, gaining a slight edge over k++ADF (*adm*). This suggests that using complete instead of admissible as the abstraction results in better performance in some cases. By comparing the results for k++ADF ($cf + grd$) and k++ADF (*cf*), we also observe that the effect of precomputing the grounded extension results in a considerable performance improvement for the third-level procedure, especially for the Planning domain. This can also be seen from Fig. 13 (left) by noting that over 50 more instances are solved using this shortcut. From Fig. 13 (right), we may observe that especially on the ICCMA'19 instances, using complete semantics as the base semantics results in k++ADF being slightly slower on easier instances, but being able to solve a handful instances more.

We visualize the effect of the parameter k on solving times in Fig. 14, both for $Cred_{adm}$ (left) and $Skept_{prf}$ (right). Interestingly, for both of these problems, goDiamond, QADF, and YADF reach the timeout limit of 1800 seconds for all instances already at $k = 8$, although these solvers do not have a direct dependency on k. However, instances with a high value of k tend to also have a high number of parents for some arguments. We also observe that k++ADF, with the k-bipolar encodings, is able to solve instances with $k \leq 15$ relatively easily, with at most a 250-second mean running time at $k = 15$. After this, instances become exponentially harder to solve in practice, which is due to the sheer size of the encodings, which are exponential in k. However, our approach—based on parameterizing the encodings via k-bipolarity—allows for scaling up much further than other solvers.

Thus, we have showed that k++ADF is in fact the state-of-the-art approach to reasoning over these acceptance problems in ADFs. Finally, we take a closer look at the differences between different configurations of k++ADF. From Fig. 15, we observe that for both $Cred_{adm}$ (upper right) and $Skept_{prf}$ (lower right), using the k-bipolar encoding produces a lot less timeouts than the second-level or third-level CEGAR procedure, respectively. However, in particular for some instances with

Table 6
Mean running times for $Skept_{prf}$ (timeouts included as the timeout limit of 1800s). Number of timeouts (out of 100 instances for ABA2AF, Planning, and Traffic; out of 276 instances for ICCMA'19) in brackets.

Domain	k++ADF (*com*)	k++ADF (*adm*)	k++ADF (*cf* + *grd*)	k++ADF (*cf*)	goDiamond	QADF	YADF
ABA2AF	**254.96** (**13**)	270.68 (14)	456.15 (25)	777.36 (41)	979.14 (53)	1481.34 (81)	1011.64 (53)
Planning	20.93 (**0**)	**8.26** (**0**)	681.93 (37)	1338.87 (72)	8.28 (**0**)	1800.00 (100)	1189.36 (56)
Traffic	16.57 (**0**)	**3.45** (**0**)	432.33 (24)	558.97 (29)	12.84 (**0**)	1591.96 (87)	582.33 (27)
ICCMA'19	**527.83** (**77**)	559.41 (83)	1513.65 (229)	1575.25 (240)	1319.45 (195)	1612.18 (247)	1537.22 (232)

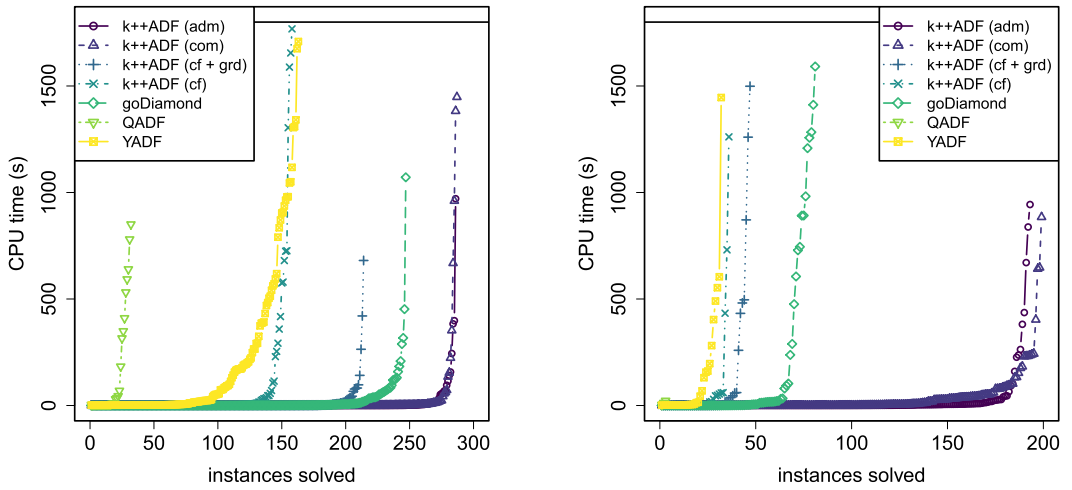

Fig. 13. Solver comparison for $Skept_{prf}$: number of solved instances (on the x-axis) given a per-instance time limit (on the y-axis) for ICCMA'17 (ABA2AF, Planning, Traffic; left) and ICCMA'19 (right) instances.

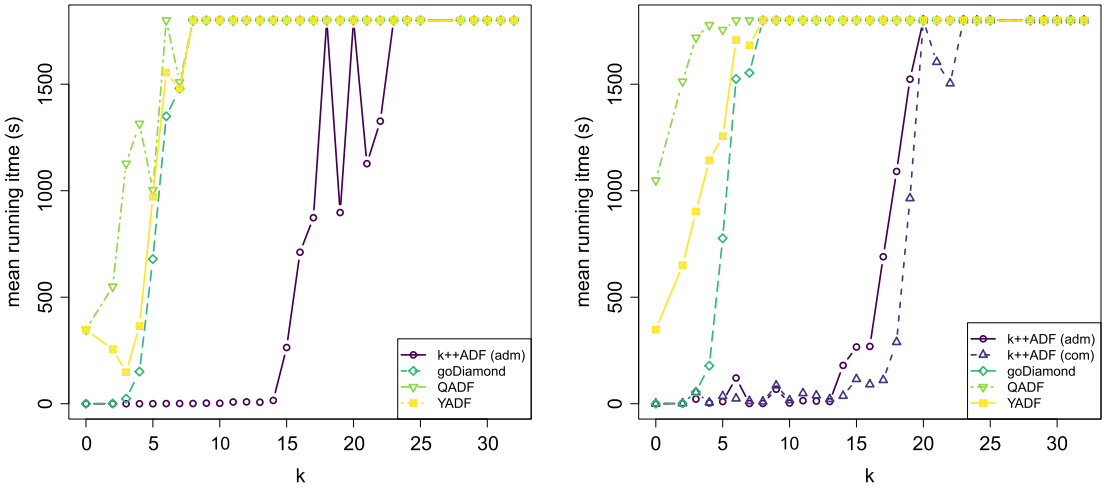

Fig. 14. Effect of the value of k (in terms of k-bipolarity) on solving times for $Cred_{adm}$ (left and for $Skept_{prf}$ (right).

a very high value of k, for both problems it is sometimes more advantageous to use the general CEGAR procedure than the k-bipolar encoding, as indicated by the timeouts for the k-bipolar encoding, consisting of yellow clusters of points on the right hand sides of the plots. This is not surprising, as the k-bipolar encoding is exponential in k, and hence its size becomes a large factor in solving these instances successfully. From Fig. 15 (lower left), we observe that for solving $Skept_{prf}$ it depends largely on the instance whether using complete as opposed to admissible as the base semantics is more efficient. However, a handful instances (with a somewhat high value of k) that time out using admissible are solved using complete semantics within the timeout limit. On the other hand, the majority of the instances are faster to solve using admissible, which is evident given the large size of the encoding for complete semantics.

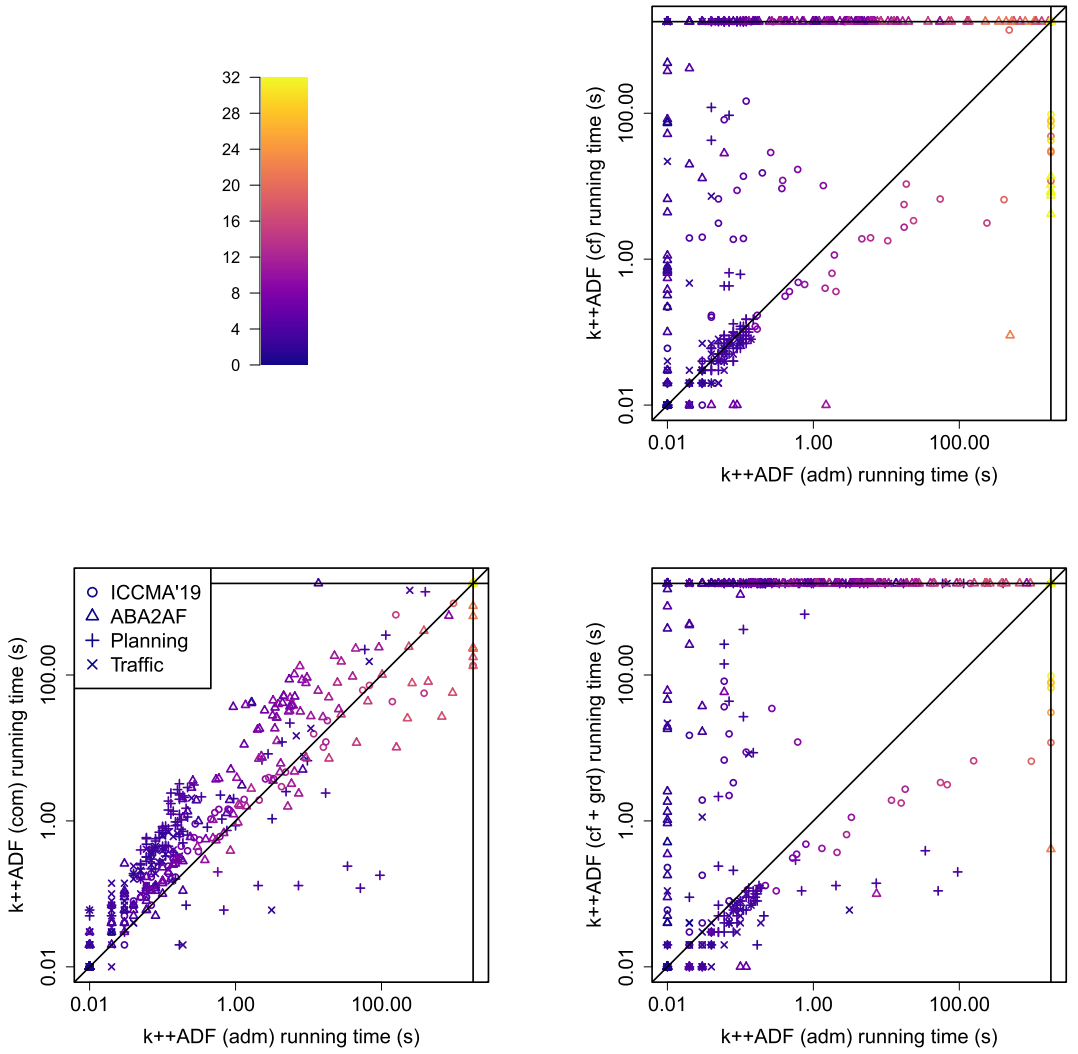

Fig. 15. Comparison of configurations for k++ADF: $Cred_{adm}$ (adm) versus (cf) (upper right), $Skept_{prf}$ (adm) versus (com) (lower left), $Skept_{prf}$ $(cf + grd)$ versus (adm) (lower right). Points in scatter plots are colored by the value of k, indicated by the colorbar (upper left).

7. Related work

The computational complexity of general and bipolar ADFs has been established by [69], showing that in most cases general ADFs yield higher complexity when compared to corresponding reasoning tasks over AFs, and bipolar ADFs exhibit the same complexity as AFs. Our work substantially extends these results by covering acyclic and concise ADFs along with distance-based subclasses. Recently [38] studied the complexity of general and bipolar ADFs under naive-based semantics using an asymmetric version of conflict-freeness, in contrast to the symmetric version introduced by [69] which we employ. Notably, the asymmetric definition results in higher complexity for verification and existence under conflict-free semantics [38]. Recalling that our SAT-based algorithms for general ADFs rely on NP membership, the symmetric version is more suitable from this perspective. For a more detailed discussion on these two versions, we refer the reader to [67].

While [69] study the same reasoning tasks as in this work, [38] also investigate credulous (resp. skeptical) rejection, which–in contrast to acceptance–asks to decide whether, given $D = (A, L, C)$ and $a \in A$, $I(a) = \mathbf{f}$ holds for some interpretation (resp. all interpretations) $I \in \sigma(D)$. Both credulous and skeptical rejection exhibit higher complexity under asymmetric conflict-free and naive semantics. In contrast, we conjecture that the complexity of acceptance and rejection coincides under all semantics and for all subclasses studied in this work. For complete-based semantics this can be confirmed: consider adding an auxiliary argument a' which is attacked by the query argument a (i.e. has the acceptance condition $\varphi_{a'} = \neg a$). It holds that a is credulously (resp. skeptically) accepted if and only if a' is credulously (resp. skeptically) rejected, and vice versa. In addition, this reduction respects all subclasses studied in this work, except for symmetric ADFs. Finally, from the

algorithmic point of view, we note that all of our procedures for acceptance are directly adaptable to rejection by switching the query literal from $a^{\mathbf{t}}$ (resp. $\neg a^{\mathbf{t}}$) to $a^{\mathbf{f}}$ (resp. $\neg a^{\mathbf{f}}$).

Existing implementations of ADFs are based on reductions either to QBF [31] or to ASP [68,14]; see also [29] for a detailed discussion. The approach by [31] proposes procedures for the main ADF reasoning tasks by means of "full" reduction to QBF, which is the prototypical PSPACE-complete problem, and thus can accommodate the tasks of interest. On the other hand, goDiamond [68] is a solver which utilizes reduction to ASP. It participated in ICCMA competitions, and it was also recently extended to work with ADFs. goDiamond requires two calls to an ASP solver when dealing with the hardest (third-level) reasoning tasks, with a possible blowup in space for the input of the second call. [14], instead, solved all ADF reasoning tasks with a single call to an ASP solver by making use of ASP programs with predicates of bounded arity. Genuine algorithms for ADFs are less explored, a recent exception is the work on the grounded semantics by [46].

Our approach resembles the CEGAR approach of Cegartix [34] for AFs, in the sense that it utilizes SAT calls and is complexity sensitive. While this approach has been proven useful for advanced reasoning tasks in the AF domain [57,58,8,59,56] recently, our aim was to extend the method to handle the more powerful formalism of ADFs. In particular, we investigated how to take advantage of the input being in advantageous ADF subclasses, also in low distance to such subclasses. A study of (distance to) subclasses was already in place in [34], but of course limited to AFs, and did not include some of the subclasses we utilize, e.g., symmetric frameworks. CEGAR approaches have been already proficiently used in other AI areas, e.g., model checking and QBFs. Starting from the original approach applied to symbolic model checking by [26,27], then instantiated by using SAT solvers [28], the approach has been applied, e.g., to the Verilog hardware description language [43] and to hybrid systems [25]. In QBF solving, [44] applied CEGAR first to 2-QBF formulas, then generalized to formulas containing an arbitrary number of quantification levels, and finally also to non-prenex and non-CNF QBF solving.

For what concerns the analysis of ADF subclasses, recently [30] investigated certain ADF subclasses also studied in our paper, i.e., acyclic and symmetric ADFs, but without an explicit complexity analysis, and without the concept of "distance to" such subclasses. Experiments are also provided and show the behavior of the three ADF systems we compared to on acyclic and cyclic instances from the domains we have analyzed in our paper. Another recent work [35] investigated ADF subclasses without supporting links in order to understand the relation between ADFs and AFs with collective attacks [55]; subclasses of the latter were also studied recently [48]. We recall that all subclasses we studied in this work allow for both attacking and supporting links.

8. Conclusion

In this work, we studied the complexity of subclasses of ADFs and incorporated these insights into a novel system for ADF acceptance problems based on incremental SAT solving. Our work is motivated by the observation that a growing number of application domains have been proposed in the literature (e.g., [1,53,20]), but the general high complexity of ADF reasoning tasks makes the design of efficient systems a challenging task. The system we provided proved to consistently outperform state-of-the-art ADF systems on benchmarks built from ICCMA'17 and ICCMA'19 instances. Our system handles six important semantics (conflict-free, naive, admissible, complete, preferred, grounded) and is available under https://bitbucket.org/andreasniskanen/k-adf.

For future work, we plan to investigate further ADF subclasses (see, e.g., [61]) and tune the system by exploiting acyclicity. Then, we plan to include other semantics; in particular stable and semi-stable [47] semantics are on our agenda next. Further avenues include (i) to get an understanding how ADF subclasses affect the expressibility of ADFs (see [51,63] for general investigations and [30] for investigation on particular ADF subclasses) and how such concepts can be integrated in our solving procedures; and (ii) designing SAT-based algorithms for ADFs involving more complex acceptance conditions, like the ones in GRAPPA ([19]) or weighted versions of ADFs [17,13].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Matti Järvisalo for discussions on the CNF translation. This work has been supported by the Austrian Science Fund (FWF): P30168, P32830, I2854, and Y698, the Vienna Science and Technology Fund (WWTF) project ICT19-065, the Academy of Finland (grants 312662, 328718), and the Doctoral Programme in Computer Science (DoCS) at the University of Helsinki. Computational resources were provided by Finnish Grid and Cloud Infrastructure (http://urn:nbn:fi:research-infras-2016072533).

References

[1] L. Al-Abdulkarim, K. Atkinson, T.J.M. Bench-Capon, A methodology for designing systems to reason with legal cases using abstract dialectical frameworks, Artif. Intell. Law 24 (2016) 1–49, https://doi.org/10.1007/s10506-016-9178-1.

[2] L. Al-Abdulkarim, K. Atkinson, T.J.M. Bench-Capon, S. Whittle, R. Williams, C. Wolfenden, Noise induced hearing loss: building an application using the ANGELIC methodology, Argum. Comput. 10 (2019) 5–22, https://doi.org/10.3233/AAC-181005.
[3] J.F.L. Alcântara, S. Sá, J.C.A. Guadarrama, On the equivalence between abstract dialectical frameworks and logic programs, Theory Pract. Log. Program. 19 (2019) 941–956, https://doi.org/10.1017/S1471068419000280.
[4] M. Alviano, W. Faber, H. Strass, Boolean functions with ordered domains in answer set programming, in: Proc. AAAI, AAAI Press, 2016, pp. 879–885.
[5] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.
[6] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G.R. Simari, M. Thimm, S. Villata, Towards artificial argumentation, AI Mag. 38 (2017) 25–36, https://doi.org/10.1609/aimag.v38i3.2704.
[7] R. Baumann, H. Strass, On the number of bipolar Boolean functions, J. Log. Comput. 27 (2017) 2431–2449, https://doi.org/10.1093/logcom/exx025.
[8] D. Baumeister, M. Järvisalo, D. Neugebauer, A. Niskanen, J. Rothe, Acceptance in incomplete argumentation frameworks, Artif. Intell. 295 (2021) 103470, https://doi.org/10.1016/j.artint.2021.103470.
[9] T.J.M. Bench-Capon, P.E. Dunne, Argumentation in artificial intelligence, Artif. Intell. 171 (2007) 619–641, https://doi.org/10.1016/j.artint.2007.05.001.
[10] M. Bichler, M. Morak, S. Woltran, lpopt: a rule optimization tool for answer set programming, Fundam. Inform. 177 (2020) 275–296, https://doi.org/10.3233/FI-2020-1990.
[11] A. Biere, F. Lonsing, M. Seidl, Blocked clause elimination for QBF, in: Proc. CADE, Springer, 2011, pp. 101–115.
[12] S. Bistarelli, L. Kotthoff, F. Santini, C. Taticchi, Containerisation and dynamic frameworks in ICCMA'19, in: Proc. SAFA, CEUR-WS.org, 2018, pp. 4–9.
[13] B. Bogaerts, Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory, in: Proc. AAAI, AAAI Press, 2019, pp. 2686–2693.
[14] G. Brewka, M. Diller, G. Heissenberger, T. Linsbichler, S. Woltran, Solving advanced argumentation problems with answer set programming, Theory Pract. Log. Program. 20 (2020) 391–431, https://doi.org/10.1017/S1471068419000474.
[15] G. Brewka, S. Ellmauthaler, H. Strass, J.P. Wallner, S. Woltran, Abstract dialectical frameworks revisited, in: Proc. IJCAI, IJCAI/AAAI, 2013, pp. 803–809.
[16] G. Brewka, S. Ellmauthaler, H. Strass, J.P. Wallner, S. Woltran, Abstract dialectical frameworks, in: Handbook of Formal Argumentation, College Publications, 2018, pp. 237–285, chapter 5.
[17] G. Brewka, H. Strass, J.P. Wallner, S. Woltran, Weighted abstract dialectical frameworks, in: Proc. AAAI, AAAI Press, 2018, pp. 1779–1786.
[18] G. Brewka, S. Woltran, Abstract dialectical frameworks, in: Proc. KR, AAAI Press, 2010, pp. 102–111.
[19] G. Brewka, S. Woltran, GRAPPA: a semantical framework for graph-based argument processing, in: Proc. ECAI, IOS Press, 2014, pp. 153–158.
[20] E. Cabrio, S. Villata, Abstract dialectical frameworks for text exploration, in: Proc. ICAART, SciTePress, 2016, pp. 85–95.
[21] C. Cayrol, M. Lagasquie-Schiex, Bipolarity in argumentation graphs: towards a better understanding, Int. J. Approx. Reason. 54 (2013) 876–899, https://doi.org/10.1016/j.ijar.2013.03.001.
[22] F. Cerutti, S.A. Gaggl, M. Thimm, J.P. Wallner, Foundations of implementations for formal argumentation, in: Handbook of Formal Argumentation, College Publications, 2018, pp. 688–767, chapter 15.
[23] F. Cerutti, M. Giacomin, M. Vallati, How we designed winning algorithms for abstract argumentation and which insight we attained, Artif. Intell. 276 (2019) 1–40, https://doi.org/10.1016/j.artint.2019.08.001.
[24] G. Charwat, W. Dvořák, S.A. Gaggl, J.P. Wallner, S. Woltran, Methods for solving reasoning problems in abstract argumentation - a survey, Artif. Intell. 220 (2015) 28–63, https://doi.org/10.1016/j.artint.2014.11.008.
[25] E.M. Clarke, A. Fehnker, Z. Han, B.H. Krogh, J. Ouaknine, O. Stursberg, M. Theobald, Abstraction and counterexample-guided refinement in model checking of hybrid systems, Int. J. Found. Comput. Sci. 14 (2003) 583–604, https://doi.org/10.1142/S012905410300190X.
[26] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, Counterexample-guided abstraction refinement, in: Proc. CAV, Springer, 2000, pp. 154–169.
[27] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, Counterexample-guided abstraction refinement for symbolic model checking, J. ACM 50 (2003) 752–794, https://doi.org/10.1145/876638.876643.
[28] E.M. Clarke, A. Gupta, O. Strichman, SAT-based counterexample-guided abstraction refinement, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 23 (2004) 1113–1123, https://doi.org/10.1109/TCAD.2004.829807.
[29] M. Diller, Realising Argumentation using Answer Set Programming and Quantified Boolean Formulas, Ph.D. thesis, TU Wien, 2019.
[30] M. Diller, A. Keshavarzi Zafarghandi, T. Linsbichler, S. Woltran, Investigating subclasses of abstract dialectical frameworks, Argum. Comput. 11 (2020) 191–219, https://doi.org/10.3233/AAC-190481.
[31] M. Diller, J.P. Wallner, S. Woltran, Reasoning in abstract dialectical frameworks using quantified Boolean formulas, Argum. Comput. 6 (2015) 149–177, https://doi.org/10.1080/19462166.2015.1036922.
[32] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artif. Intell. 77 (1995) 321–358, https://doi.org/10.1016/0004-3702(94)00041-X.
[33] W. Dvořák, P.E. Dunne, Computational problems in formal argumentation and their complexity, in: Handbook of Formal Argumentation, College Publications, 2018, pp. 631–688, chapter 13.
[34] W. Dvořák, M. Järvisalo, J.P. Wallner, S. Woltran, Complexity-sensitive decision procedures for abstract argumentation, Artif. Intell. 206 (2014) 53–78, https://doi.org/10.1016/j.artint.2013.10.001.
[35] W. Dvořák, A. Keshavarzi Zafarghandi, S. Woltran, Expressiveness of SETAFs and support-free ADFs under 3-valued semantics, in: Proc. COMMA, IOS Press, 2020, pp. 191–202.
[36] W. Dvořák, S. Ordyniak, S. Szeider, Augmenting tractable fragments of abstract argumentation, Artif. Intell. 186 (2012) 157–173, https://doi.org/10.1016/j.artint.2012.03.002.
[37] N. Eén, N. Sörensson, Temporal induction by incremental SAT solving, Electron. Notes Theor. Comput. Sci. 89 (2003) 543–560, https://doi.org/10.1016/S1571-0661(05)82542-3.
[38] S.A. Gaggl, S. Rudolph, H. Straß, On the decomposition of abstract dialectical frameworks and the complexity of naive-based semantics, J. Artif. Intell. Res. 70 (2021) 1–64, https://doi.org/10.1613/jair.1.11348.
[39] S.A. Gaggl, H. Strass, Decomposing abstract dialectical frameworks, in: Proc. COMMA, IOS Press, 2014, pp. 281–292.
[40] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: Techn. Commun. ICLP, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 2:1–2:15.
[41] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: from theory to practice, Artif. Intell. 187 (2012) 52–89, https://doi.org/10.1016/j.artint.2012.04.001.
[42] J. Heyninck, G. Kern-Isberner, M. Thimm, K. Skiba, On the correspondence between abstract dialectical frameworks and nonmonotonic conditional logics, Ann. Math. Artif. Intell. 89 (2021) 1075–1099, https://doi.org/10.1007/s10472-021-09758-y.
[43] H. Jain, D. Kroening, N. Sharygina, E.M. Clarke, VCEGAR: verilog counterexample guided abstraction refinement, in: Proc. TACAS, Springer, 2007, pp. 583–586.
[44] M. Janota, W. Klieber, J. Marques-Silva, E.M. Clarke, Solving QBF with counterexample guided refinement, Artif. Intell. 234 (2016) 1–25, https://doi.org/10.1016/j.artint.2016.01.004.
[45] A. Keshavarzi Zafarghandi, Investigating Subclasses of Abstract Dialectical Frameworks, Master's thesis, TU Wien, 2017.
[46] A. Keshavarzi Zafarghandi, R. Verbrugge, B. Verheij, A discussion game for the grounded semantics of abstract dialectical frameworks, in: Proc. COMMA, IOS Press, 2020, pp. 431–442.

[47] A. Keshavarzi Zafarghandi, R. Verbrugge, B. Verheij, Semi-stable semantics for abstract dialectical frameworks, in: Proc. KR, 2021, pp. 422–431.
[48] M. König, Graph-Classes of Argumentation Frameworks with Collective Attacks, Master's thesis, TU Wien, 2020.
[49] Y. Lierler, cmodels - SAT-based disjunctive answer set solver, in: Proc. LPNMR, Springer, 2005, pp. 447–451.
[50] T. Linsbichler, M. Maratea, A. Niskanen, J.P. Wallner, S. Woltran, Novel algorithms for abstract dialectical frameworks based on complexity analysis of subclasses and SAT solving, in: Proc. IJCAI, ijcai.org, 2018, pp. 1905–1911.
[51] T. Linsbichler, J. Pührer, H. Strass, A uniform account of realizability in abstract argumentation, in: Proc. ECAI, IOS Press, 2016, pp. 252–260.
[52] F. Lonsing, U. Egly, DepQBF 6.0: a search-based QBF solver beyond traditional QCDCL, in: Proc. CADE, Springer, 2017, pp. 371–384.
[53] D. Neugebauer, Generating defeasible knowledge bases from real-world argumentations using D-BAS, in: Proc. AI^3@AI*IA, CEUR-WS.org, 2017, pp. 105–110.
[54] D. Neugebauer, Bridging the Gap between Online Discussions and Formal Models of Argumentation, Ph.D. thesis, University of Düsseldorf, Germany, 2019.
[55] S.H. Nielsen, S. Parsons, A generalization of Dung's abstract framework for argumentation: arguing with sets of attacking arguments, in: Proc. ArgMAS, Springer, 2006, pp. 54–73.
[56] A. Niskanen, Computational Approaches to Dynamics and Uncertainty in Abstract Argumentation, Ph.D. thesis, University of Helsinki, Finland, 2020.
[57] A. Niskanen, M. Järvisalo, μ-toksia: an efficient abstract argumentation reasoner, in: Proc. KR, 2020, pp. 800–804.
[58] A. Niskanen, M. Järvisalo, Algorithms for dynamic argumentation frameworks: an incremental SAT-based approach, in: Proc. ECAI, IOS Press, 2020, pp. 849–856.
[59] A. Niskanen, D. Neugebauer, M. Järvisalo, Controllability of control argumentation frameworks, in: Proc. IJCAI, ijcai.org, 2020, pp. 1855–1861.
[60] C.H. Papadimitriou, Computational Complexity, Academic Internet Publ., 2007.
[61] S. Polberg, Understanding the abstract dialectical framework, in: Proc. JELIA, 2016, pp. 430–446.
[62] S. Polberg, J.P. Wallner, S. Woltran, Admissibility in the abstract dialectical framework, in: Proc. CLIMA, Springer, 2013, pp. 102–118.
[63] J. Pührer, Realizability of three-valued semantics for abstract dialectical frameworks, Artif. Intell. 278 (2020), https://doi.org/10.1016/j.artint.2019.103198.
[64] L.J. Stockmeyer, The polynomial-time hierarchy, Theor. Comput. Sci. 3 (1976) 1–22, https://doi.org/10.1016/0304-3975(76)90061-X.
[65] H. Strass, Implementing instantiation of knowledge bases in argumentation frameworks, in: Proc. COMMA, IOS Press, 2014, pp. 475–476.
[66] H. Strass, Expressiveness of two-valued semantics for abstract dialectical frameworks, J. Artif. Intell. Res. 54 (2015) 193–231, https://doi.org/10.1613/jair.4879.
[67] H. Strass, Abstract Dialectical Frameworks. An Analysis of Their Properties and Role in Knowledge Representation and Reasoning, Habilitationsschrift, Universität Leipzig, 2017.
[68] H. Strass, S. Ellmauthaler, goDiamond 0.6.6, http://dbai.tuwien.ac.at/iccma17/files/goDIAMOND.pdf, 2017.
[69] H. Strass, J.P. Wallner, Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory, Artif. Intell. 226 (2015) 34–74, https://doi.org/10.1016/j.artint.2015.05.003.
[70] M. Thimm, S. Villata, The first international competition on computational models of argumentation: results and analysis, Artif. Intell. 252 (2017) 267–294, https://doi.org/10.1016/j.artint.2017.08.006.
[71] G.S. Tseitin, On the complexity of derivation in propositional calculus, in: Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970, Springer, 1983, pp. 466–483.