Reduction-Based Approaches to Implement Modgil's Extended Argumentation Frameworks

Wolfgang Dvořák[1], Sarah Alice Gaggl[2], Thomas Linsbichler[3], and Johannes Peter Wallner[3]

[1] University of Vienna, Faculty of Computer Science, Austria
[2] Technische Universität Dresden, Computational Logic Group, Germany
[3] Vienna University of Technology, Institute of Information Systems, Austria

Abstract. This paper reconsiders Modgil's Extended Argumentation Frameworks (EAFs) that extend Dung's abstract argumentation frameworks by attacks on attacks. This allows to encode preferences directly in the framework and thus also to reason about the preferences themselves. As a first step to reduction-based approaches to implement EAFs, we give an alternative (but equivalent) characterization of acceptance in EAFs. Then we use this characterization to provide EAF encodings for answer set programming and propositional logic. Moreover, we address an open complexity question and the expressiveness of EAFs.

1 Introduction

Since the seminal paper of Dung in 1995 [9] argumentation has emerged to one of the major research fields in artificial intelligence and non-monotonic reasoning, with Dung's *abstract argumentation frameworks* (AFs) being one of the core formalisms. In this very simple yet expressive model, arguments and a binary *attack* relation between them, denoting conflicts, are the only components one needs for the representation of a wide range of problems and the reasoning therein. Nowadays numerous semantics exist to solve the inherent conflicts between the arguments by selecting sets of "acceptable" arguments.

In certain scenarios there are *preferences* about which arguments should go into the set of acceptable arguments, e.g. because the source of one argument is more trustworthy than the source of another [18]. Such preferences can have a significant impact on the evaluation of discussions. Consider for example a situation with two mutually conflicting arguments a and b. The only possibilities (under e.g. stable semantics of AFs) would be to accept either a or b. Thus, neither argument is skeptically justified, i.e. none of them appears in each solution, but given a preference of argument a over b one can resolve this situation such that a is skeptically justified. However, the basic Dung-style framework does not support the handling of preferences within the framework, neither on a syntactical nor on a semantical level. For example it is not possible to model a situation where one argument (resp. attack) is preferred over another one, or where some particular preference weakens an attack between two arguments.

T. Eiter et al. (Eds.): Brewka Festschrift, LNAI 9060, pp. 249–264, 2015.

Several approaches for incorporating preferences have been proposed in the literature. When *instantiating* an AF from a knowledge base one can deal with preferences in the underlying logical formalism and resolve them when building the framework (see e.g. [19]). Preferences can also be handled at the abstract level by generalizations of AFs. In *preference-based argumentation frameworks* (PAFs) [1] one has a partial ordering over the arguments, and an attack is only successful if the attacked argument is not preferred over the attacker with respect to the ordering. Thus the acceptability of an argument can be based either on defense or on preference with respect to the attacking arguments. *Value-based argumentation frameworks* (VAFs) [3] allow to assign values to the arguments. An additional ordering over the values can be used to evaluate preferences in a similar way as in PAFs. Brewka and Woltran introduced prioritized *bipolar abstract dialectical frameworks* (BADFs) [5] which allow to express for each statement a strict partial order on the links leading to it. Then, a statement is accepted unless its attackers are jointly preferred.

All these approaches have in common that they are tailored to *fixed* preferences. In some scenarios it might very well be the case that the assumed preference ordering is itself open to debate. Modgil's *extended argumentation frameworks* (EAFs) [18] are particularly appealing in this regard, as they allow to represent preferences as defeasible arguments themselves. More concretely, this approach is based on the idea that a preference for one argument a over another argument b can weaken an attack from b to a. Considering the example with mutually attacking arguments from above, in EAFs one can resolve this situation by introducing an argument c which stands for a preference of a over b by attacking the attack from b to a. Thereby, argument a is reinstated, while b cannot be accepted. However, if c is attacked by another argument d, the argument b can be reinstated again. Thus, EAFs can be used as a *meta-argumentation* approach to argue also about the preferences, where the acceptance of an argument depends on whether it can be *reinstated*. For instance one can encode VAFs as EAFs and then argue about the value ordering [18].

Although Modgil presented an extensive study of the new formalism and its extensions to VAFs and logic programs in [18], several computational properties of EAFs have been neglected therein. Dunne et al. [12] gave an exact *complexity* classification for reasoning in EAFs. They showed that whether an argument is acceptable w.r.t. a given set can be decided in polynomial time via a reduction to an AF. Hence the reasoning tasks in EAFs have the same complexity as in AFs. Later this reduction has also been turned into labeling-based algorithms [21]. In this work we will show the exact complexity of Grounded-Scepticism, i.e. of deciding whether the grounded extension is contained in all preferred extensions, which was left open in [12]. Moreover we will show that, despite reasoning tasks having the same complexity, EAFs enjoy higher *expressiveness* in terms of realizability [11] compared to Dung-style AFs.

Recently the reduction-based approach for the implementation of argumentation related problems became very popular. In particular reductions to well established formalisms like answer set programming (ASP) [6,20] and propositional

logic turned out to be suitable for the relevant reasoning problems [15,4,13]. So far, no such approach is known for EAFs. We believe this is partly due to fact that the given characterizations for the acceptance of an argument are not well suited for such encodings. Thus we will first present an alternative, but equivalent, characterization for the acceptance of an argument which then allows us to design succinct ASP encodings for all standard semantics of EAFs. These encodings have been incorporated in the web-interface *GERD - Genteel Extended argumentation Reasoning Device* and are freely accessible under `http://gerd.dbai.tuwien.ac.at`. Furthermore, the alternative characterization facilitates encodings in terms of propositional formulas which we will exemplify on the admissible semantics.

The organization of the remainder of the paper is as follows: In Section 2 we give the necessary background on argumentation and answer set programming. In Section 3 we first show an alternative characterization of acceptance and then exploit this characterization to encode the semantics in answer set programming and propositional logic. Further, in Section 4 we provide an exact complexity characterization of Grounded-Scepticism, an open problem raised in [12] and show that all the EAF semantics from [18], except grounded, are more expressive than their counterparts in standard Dung AFs. Finally we conclude in Section 5.

2 Background

In this section we briefly introduce Dung's abstract argumentation frameworks (AFs) [9] (for an introduction to abstract argumentation see [2]) and Modgil's extended argumentation frameworks (EAFs) [18]. We first give the definition of AFs. In contrast to [9] we restrict ourselves to finite frameworks.

Definition 1. *An* Argumentation Framework (AF) *is a pair* $F = (A, R)$ *where* A *is a non-empty, finite set of arguments and* $R \subseteq A \times A$ *is the attack relation.*

The idea of EAFs is to express preferences of arguments over each other by allowing attacks on attacks. This allows one to argue about the preferences themselves. Attacks on attacks are implemented by an additional relation D which relates arguments to attacks in R.

Definition 2. *An* Extended Argumentation Framework (EAF) *is a triple* $F = (A, R, D)$ *where* (A, R) *is an AF and* $D \subseteq A \times R$ *a relation describing an argument* x *attacking an attack* $(y, z) \in R$*. Moreover, whenever* $\{(x, (y, z)), (x', (z, y))\} \subseteq D$ *then* $\{(x, x'), (x', x)\} \subseteq R$*.* [1]

Given a set of arguments $S \subseteq A$*, an attack* $(x, y) \in R$ succeeds *w.r.t.* S *(we write* $x \rightarrowtail^S y$*) iff there is no* $z \in S$ *with* $(z, (x, y)) \in D$*. By* R_S *we denote the relation containing all attacks* $(x, y) \in R$ *that succeed w.r.t.* S*. A set* $S \subseteq A$ *is said to be* conflict-free *in* F*, i.e.* $S \in cf(F)$*, if* $x \not\rightarrowtail^S y$ *and* $\{(x, y), (y, x)\} \not\subseteq R$ *for all* $x, y \in S$*.*

[1] Note that this property is essential for showing Dung's fundamental lemma for EAFs. However our implementations would still work for EAFs violating this property.

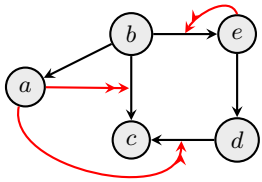


Fig. 1. The EAF F from Example 1

Now, as attacks can be defeated themselves, when defending an argument we have to make sure that also the used attacks are defended.

Definition 3. *Given an EAF $F = (A, R, D)$, $S \subseteq A$, and $v \rightarrowtail^S w$. Then $\mathcal{RS} \subseteq R_S$ is a* reinstatement set *for $v \rightarrowtail^S w$ if it satisfies the following conditions:*

R1 $v \rightarrowtail^S w \in \mathcal{RS}$.
R2 For each $(y, z) \in \mathcal{RS}$ *it holds that* $y \in S$.
R3 For every $(y, z) \in \mathcal{RS}$ *and every* $(x, (y, z)) \in D$ *there is a* $(y', x) \in \mathcal{RS}$.

An argument $a \in A$ is acceptable *w.r.t. (or* defended *by) a set $S \subseteq A$ if whenever $z \rightarrowtail^S a$ then there is a $y \in S$ with $y \rightarrowtail^S z$ and there is a reinstatement set for $y \rightarrowtail^S z$. For any $S \in cf(F)$ the* characteristic function $\mathcal{F}_F$ *is defined as* $\mathcal{F}_F(S) = \{x \mid x \text{ is acceptable w.r.t. } S\}$.

Example 1. Consider the EAF $F = (A, R, D)$ from Figure 1, and let $S = \{b, d\}$. Then, $R_S = \{(b, a), (b, c), (b, e), (d, c), (e, d)\}$, and there are the following reinstatement sets for the succeeding attacks:

- $\mathcal{RS}$ for $b \rightarrowtail^S a$: $\{(b, a)\}$;
- $\mathcal{RS}$ for $b \rightarrowtail^S c$: $\{(b, c), (b, a)\}$;
- $\mathcal{RS}$ for $b \rightarrowtail^S e$: $\{(b, e)\}$;
- $\mathcal{RS}$ for $d \rightarrowtail^S c$: $\{(d, c), (b, a)\}$.

There is no reinstatement set for $e \rightarrowtail^S d$, as $e \notin S$. Regarding acceptability, the argument d is acceptable w.r.t. S because for $e \rightarrowtail^S d$ we have $b \in S$ with $b \rightarrowtail^S e$ with $\mathcal{RS} = \{(b, e)\}$. Furthermore, b is acceptable w.r.t. S as well. ◇

Definition 4. *Given an EAF $F = (A, R, D)$, a conflict-free set S is*

- *an* admissible set, *i.e. $S \in adm(F)$, if each $a \in S$ is acceptable w.r.t. S,*
- *a* preferred extension, *i.e. $S \in prf(F)$, if S is a $\subseteq$-maximal admissible set,*
- *a* stable extension, *i.e. $S \in stb(F)$, if for each $b \notin S$, there is some $a \in S$ with $a \rightarrowtail^S b$,*
- *a* complete extension, *i.e. $S \in com(F)$, if $a \in S$ iff a is acceptable w.r.t. S,*
- *and the* grounded extension *$grd(F)$ is given by $grd(F) = \bigcup_{k \geq 1} \mathcal{F}_F^k(\emptyset)$.*

Example 2. For the EAF F from Figure 1 we have the following extensions: $adm(F) = \{\emptyset, \{e\}, \{b\}, \{b, d\}, \{b, e\}\}$, $stb(F) = com(F) = prf(F) = \{\{b, d\}, \{b, e\}\}$, and $\{b, d\}$ is the unique grounded extension. ◇

2.1 Answer Set Programming

In this section we recall the basics of logic programs under the answer set semantics [6,20].

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

$$a \leftarrow b_1, \ldots, b_k, \ not\, b_{k+1}, \ldots, \ not\, b_m.$$

with $m \geq k \geq 0$, where $a, b_1, \ldots, b_m$ are atoms, and "*not*" stands for *default negation*. The *head* of r is the set $H(r) = \{a\}$ and the *body* of r is $B(r) = \{b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m\}$. Furthermore, $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. A *constraint* is a rule with empty head. A rule r is *safe* if each variable in r occurs in $B^+(r)$. A rule r is *ground* if no variable occurs in r. A *fact* is a ground rule with empty body. An (input) database is a set of facts. A program is a finite set of rules. For a program π and an input database D, we often write $\pi(D)$ instead of $D \cup \pi$.

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

3 Reduction-Based Approaches to EAFs

Towards reductions to answer set programming encodings and propositional logic we first give an alternative characterization of acceptance.

3.1 An Alternative Characterization of Acceptance

Reinstatement sets are defined for a single attack in an EAF $F = (A, R, D)$ and a set of arguments $S \subseteq A$. Here we show that we just need to consider one reinstatement set for all attacks in R_S.

Lemma 1. *If $\mathcal{RS}$, $\mathcal{RS}'$ are reinstatement sets for $y \rightarrowtail^S z$ and $y' \rightarrowtail^S z'$ respectively then $\mathcal{RS} \cup \mathcal{RS}'$ is a reinstatement set for both $y \rightarrowtail^S z$ and $y' \rightarrowtail^S z'$.*

Proof. We have to verify conditions R1-R3 from Definition 3.
R1) We have $y \rightarrowtail^S z \in \mathcal{RS} \cup \mathcal{RS}'$ and $y' \rightarrowtail^S z' \in \mathcal{RS} \cup \mathcal{RS}'$ as the former is contained in $\mathcal{RS}$ and the latter is in $\mathcal{RS}'$.

R2) Consider $(y, z) \in \mathcal{RS} \cup \mathcal{RS}'$ and w.l.o.g. assume that $(y, z) \in \mathcal{RS}$. As $\mathcal{RS}$ is a reinstatement set for $y \rightarrowtail^S z$ we have $y \in S$.
R3) Consider $(y, z) \in \mathcal{RS} \cup \mathcal{RS}'$ with $(x, (y, z)) \in D$ and again w.l.o.g. assume that $(y, z) \in \mathcal{RS}$. As $\mathcal{RS}$ is a reinstatement set for $y \rightarrowtail^S z$ we have that there is a $(y', x) \in \mathcal{RS}$ and thus also $(y', x) \in \mathcal{RS} \cup \mathcal{RS}'$. □

As the union of two reinstatement sets for the same set S is again a reinstatement set there exists a unique maximal reinstatement set. This is by a standard argument: assume that there are two of them then the union of them would be a larger one contradicting the maximality of the original ones. This leads us to the definition of the *maximal reinstatement set* $\mathcal{RS}[S]$ of a set S.

Definition 5. *Given an EAF (A, R, D) and $S \subseteq A$. The (unique)* maximal reinstatement set $\mathcal{RS}[S]$ of S *is the maximal subset of R_S satisfying*
R2 For each $(y, z) \in \mathcal{RS}[S]$ it holds that $y \in S$.
R3 For every $(y, z) \in \mathcal{RS}[S]$ and every $(x, (y, z)) \in D$ there is a $(y', x) \in \mathcal{RS}[S]$.

We next show that when it comes to the verification of extensions S in EAFs we only have to consider the *maximal reinstatement set* $\mathcal{RS}[S]$ instead of all possible reinstatement sets for each attack $y \rightarrowtail^S z$.

Proposition 1. *Given an EAF $F = (A, R, D)$, $S \subseteq A$, and $y \rightarrowtail^S z$. There exists a reinstatement set for $y \rightarrowtail^S z$ iff $\mathcal{RS}[S]$ is a reinstatement set for $y \rightarrowtail^S z$.*

Proof. $\Rightarrow$: Towards a contradiction assume that there is a reinstatement set $\mathcal{RS}$ for $y \rightarrowtail^S z$ but $y \rightarrowtail^S z \notin \mathcal{RS}[S]$. Then by Lemma 1 the set $\mathcal{RS} \cup \mathcal{RS}[S]$ would be a reinstatement set for $y \rightarrowtail^S z$. Thus $\mathcal{RS}[S] \subset \mathcal{RS} \cup \mathcal{RS}[S]$ and $\mathcal{RS} \cup \mathcal{RS}[S]$ satisfying R2 and R3 contradicting the maximality of $\mathcal{RS}[S]$.
$\Leftarrow$: By assumption $\mathcal{RS}[S]$ is a reinstatement set for $y \rightarrowtail^S z$. □

Next we reformulate the condition for an argument to be acceptable.

Corollary 1. *Given an EAF $F = (A, R, D)$, an argument $a \in A$ is acceptable w.r.t. $S \subseteq A$ if whenever $z \rightarrowtail^S a$ then there is some $y \in S$ with $(y, z) \in \mathcal{RS}[S]$.*

Given S, the reinstatement set $\mathcal{RS}[S]$ can be computed in polynomial time.

Proposition 2. *Given an EAF (A, R, D) and $S \subseteq A$. $\mathcal{RS}[S]$ can be computed in polynomial time.*

Proof. The proof proceeds as follows. We first present a procedure to compute $\mathcal{RS}[S]$ and then show correctness and that it terminates in polynomial time.

Procedure:

- Start with $U = R_S \cap (S \times A)$.
- Repeat until fixed-point is reached:
  - For each $y \rightarrowtail^S z \in U$: if there is $(x, (y, z)) \in D$ such that there is no $(y', x) \in U$ then remove $y \rightarrowtail^S z$ from U.
- return $\mathcal{RS}[S] = U$

Correctness: To prove correctness we show (1) that the fixed-point satisfies R2 and R3. and (2) that in each iteration only attacks which are not in $\mathcal{RS}[S]$ are removed, i.e. $\mathcal{RS}[S] \subseteq U$ holds during the whole procedure.

(1) The property R2 is ensured by the initialization $U = R_S \cap (S \times A)$, that is at each time the set U only contains (y, z) with $y \in S$. Now consider property R3. As the algorithm terminated we have that for every $y \rightarrowtail^S z \in U$, if there is a $(x, (y, z)) \in D$ then there is also a $(y', x) \in U$. That is R3 holds.

(2) We prove this by induction on the number of iterations n. As base case we consider $n = 1$ meaning that the algorithm returns $U = R_S \cap (S \times A)$. As by definition $\mathcal{RS}[S] \subseteq R_S \cap (S \times A)$ we are fine. Now let U_i be the set after the i-th iteration. For the induction step we assume that $\mathcal{RS}[S] \subseteq U_{n-1}$ and show that then also $\mathcal{RS}[S] \subseteq U_n$. To this end consider a $(y, z) \in U_{n-1} \setminus U_n$. Then there is an $(x, (y, z)) \in D$ such that there is no $(y', x) \in U_{n-1}$. But this implies that there is an $(x, (y, z)) \in D$ such that there is no $(y', x) \in \mathcal{RS}[S]$ and thus, because of property R3, $(y, z) \notin \mathcal{RS}[S]$. Hence $\mathcal{RS}[S] \subseteq U_n$.

Polynomial-Time: For the initialization step notice that R_S can be computed in polynomial time and also checking whether an attack has its source in S is easy. As in each iteration of the loop, except the last one, at least one attack is removed from the set, there are at most as many iterations as attacks. Finally the condition in the loop can be tested in polynomial time. □

3.2 Answer Set Programming Encodings

In this section we present ASP encodings based on our characterization of EAF acceptance of arguments. In our encodings we will use atoms $\mathbf{in}(a)$ to represent that an argument a is in an extension. The answer-sets of the combination of an encoding for a semantics σ with an ASP representation of an EAF F are in a 1-to-1 correspondence to $\sigma(F)$. More formally we have the following correspondence.

Definition 6. *Let I be an interpretation, $\mathcal{I}$ a set of interpretations, S a set and $\mathcal{S}$ a set of sets. We define $I \cong S$ iff $\{a \mid \mathbf{in}(a) \in I\} = S$. Further, $\mathcal{I} \cong \mathcal{S}$ iff there is a bijective function $f : \mathcal{I} \to \mathcal{S}$ such that for each $I \in \mathcal{I}$ we have $I \cong f(I)$.*

For readability we partition the encodings into several modules. We begin with the input database for a given EAF $F = (A, R, D)$, i.e. the facts representing the EAF.

$$\begin{aligned}\hat{F} :=& \{\mathbf{arg}(x). \mid x \in A\} \cup \\ & \{\mathbf{att}(x, y). \mid (x, y) \in R\} \cup \\ & \{\mathbf{d}(x, y, z). \mid (x, (y, z)) \in D\}\end{aligned}$$

That is, $\mathbf{arg}(x)$ is a fact that represents that x is an argument in F. The binary predicate $\mathbf{att}(x, y)$ indicates that there is an attack from x to y and $\mathbf{d}(x, y, z)$ signifies that there is an attack from x to the attack from y to z.

Listing 1.1. Module π_{cf}

```
% guess a set S
in(X) ← arg(X), not out(X).
out(X) ← arg(X), not in(X).

% mutually attacking arguments are forbidden in a cf set
← att(X,Y), att(Y,X), in(X), in(Y).

% canceled attacks via D
cancel(X,Y) ← att(X,Y), in(Z), d(Z,X,Y).
succeed(X,Y) ← att(X,Y), not cancel(X,Y).
← in(X), in(Y), succeed(X,Y).
```

The first basic module π_{cf} is shown in Listing 1.1. Comments can be distinguished from rules by the preceding '%' symbol. The first two lines encode a typical ASP guess. The **in** and **out** predicates identify a subset S of the arguments in the given EAF. If $\mathbf{in}(x)$ is present in an answer-set then $x \in S$ and otherwise we have $\mathbf{out}(x)$ in the answer-set and $x \notin S$. The first constraint encodes that mutually attacking arguments cannot be in a conflict-free set of F. Using the predicates **succeed** and **cancel** we can derive all attacks (x, y) which are canceled by a $(z, (x, y)) \in D$, s.t. $z \in S$ for the guessed S. The last line encodes that no two conflicting arguments can be in S, if an attack in either direction succeeds.

Next we look at module π_{rs} in Listing 1.2, which computes $\mathcal{RS}[S]$ in the predicate **rs**. Intuitively the "procedure" is as in the proof of Proposition 2. We collect with **rsinit** all successful attacks coming from an argument in S. If for such an attack (y, z) there is an $x \in A$ s.t. $(x, (y, z)) \in D$, then we need to check if the attack (y, z) is reinstated by $\mathcal{RS}[S]$, in particular we need to check if there is an attack $(y', x) \in \mathcal{RS}[S]$. We mark such a case with $\mathbf{todef}(x, y, z)$. The procedure for computing the maximal reinstatement set now starts with the initial set of attacks and iteratively removes attacks until a fixed-point is reached. We remove (y, z) if there is an $(x, (y, z)) \in D$, s.t. in the set of the current iteration there is no (y', x).

The fixed-point computation is simulated by the predicate **unattacked upto** and **remove**. The latter predicate marks attacks to be removed from **rsinit** in order to compute the unique maximal reinstatement set in **rs**. We iterate for each removal candidate marked by $\mathbf{todef}(x, y, z)$ over each argument n in the EAF. If $\mathbf{rsinit}(n, x)$ is not derivable or $\mathbf{remove}(n, x)$ was derived then (n, x) is not in the maximal reinstatement set and thus does not defend the attack (y, z) from $(x, (y, z))$. If this holds for all arguments in the EAF, then (y, z) is not defended and we mark it for removal by $\mathbf{remove}(y, z)$. For achieving this we use the module π_{order} to impose an order on the arguments. This is a standard module used in several ASP encodings of AF semantics, e.g. in [15]. We present here only the main predicates defined in this module. The predicate $\mathbf{lt}(x, y)$

Listing 1.2. Module π_{rs}

```
% rsinit represents all succeeding attacks coming from S
rsinit(Y,Z) ← in(Y), succeed(Y,Z).

% removal candidates
todef(X,Y,Z) ← rsinit(Y,Z), d(X,Y,Z).

% remove attacks
unattacked upto(X,Y,Z,N) ← inf(N), todef(X,Y,Z),
                           not rsinit(N,X).
unattacked upto(X,Y,Z,N) ← inf(N), todef(X,Y,Z), remove(N,X).
unattacked upto(X,Y,Z,N) ← succ(M,N),
                           unattacked upto(X,Y,Z,M),
                           not rsinit(N,X).
unattacked upto(X,Y,Z,N) ← succ(M,N),
                           unattacked upto(X,Y,Z,M),
                           remove(N,X).
unattacked(X,Y,Z) ← sup(N), unattacked upto(X,Y,Z,N).
remove(Y,Z) ← unattacked(X,Y,Z).

% rs represents RS[S]
rs(X,Y) ← rsinit(X,Y), not remove(X,Y).
```

is used to relate x and y, s.t. x is ordered lower than y. Using $\mathbf{succ}(x, y)$ we derive that y is the immediate successor of x in this ordering and lastly **inf** and **sup** are the infimum and supremum elements. Now, we start with the infimum argument and go through the successor predicate **succ** to the next argument. If (y, z) is undefended up to the supremum then we have to remove it. Intuitively $\mathbf{unattacked\ upto}(x, y, z, n)$ states that $(x, (y, z))$ is not successfully attacked by an attack in $\mathcal{RS}[S]$ up to the argument n in the ordering. Lastly, in **rs** we simply derive all attacks from **rsinit**, for which we cannot derive that the attack should be removed. The attacks derived via **rs** correspond to $\mathcal{RS}[S]$.

In $\pi_{defense}$ (Listing 1.3) we simply state that each y is defeated if there is an attack in our reinstatement set given by **rs**. Note that we still refer to a guessed set S. Using this we derive which arguments are undefended. Now we present our ASP encoding for admissible sets. We combine the modules for the conflict-free property, reinstatement sets, order and defense and add an intuitive constraint ensuring that if an argument is in, then it has to be defended.

$$\pi_{adm} := \pi_{cf} \cup \pi_{rs} \cup \pi_{order} \cup \pi_{defense} \cup \{\leftarrow \mathbf{in}(X), \mathbf{undefended}(X).\}$$

It is straightforward to extend this encoding to complete semantics as follows.

$$\pi_{com} := \pi_{adm} \cup \{\leftarrow \mathbf{out}(X), \mathit{not}\ \mathbf{undefended}(X).\}$$

Listing 1.3. Module $\pi_{defense}$

```
% arguments which are defeated by RS[S]
defeated(X) ← rs(X,Y).

% undefended arguments
undefended(A) ← arg(A), succeed(Z,A), not defeated(Z).
```

Listing 1.4. Module π_{range}

```
in range(Z) ← in(Y), succeed(Y,Z).
```

For the stable semantics we compute for $S \subseteq A$ the set $\{a \mid b \rightarrowtail^S a, b \in S\}$. This is encoded in π_{range} in Listing 1.4. Stable semantics can be computed via

$$\pi_{stb} := \pi_{cf} \cup \pi_{range} \cup \{\leftarrow \mathbf{out}(Z), \mathit{not}\ \mathbf{in\ range}(Z).\}$$

The 1-to-1 correspondence between the answer-sets of our encodings and the σ-extensions is summarized in the following proposition.

Proposition 3. *For any EAF F: (i) $\mathcal{AS}(\pi_{cf}(\hat{F})) \cong cf(F)$; (ii) $\mathcal{AS}(\pi_{adm}(\hat{F})) \cong adm(F)$; (iii) $\mathcal{AS}(\pi_{com}(\hat{F})) \cong com(F)$; and (iv) $\mathcal{AS}(\pi_{stb}(\hat{F})) \cong stb(F)$.*

Encodings for grounded semantics of EAFs are straightforward to achieve via techniques used in [15]. Essentially by starting with the empty set we derive the grounded extension of a given EAF, by iteratively applying the characteristic function of EAFs [18]. The ASP encoding of the characteristic function is based on the module π_{rs}.

In spirit of promising approaches for computing reasoning tasks under preferred semantics [8,13] in AFs we can compute preferred extensions in EAFs by iteratively using simple adaptations of encodings for admissible semantics. The basic idea is to traverse the search space of admissible (or complete) extensions and iteratively compute larger admissible sets until we reach a maximal set. By restricting the future search space to admissible sets not contained in previously found preferred extensions, we can compute all preferred extensions in this way.

We implemented reasoning for EAFs under conflict-free, admissible, complete, grounded, preferred and stable semantics in the tool "GERD" available online[2]. Except for preferred semantics, we provide a single ASP encoding for download which computes all extensions of the desired semantics if augmented with an input database representing the given EAF. For solving one can use modern ASP solvers, like clingo [17]. For preferred semantics we provide a UNIX bash script, which calls clingo repeatedly to compute preferred extensions in the manner described above. In Fig. 2 one can see a screenshot of the web-interface.

[2] See `http://gerd.dbai.tuwien.ac.at`

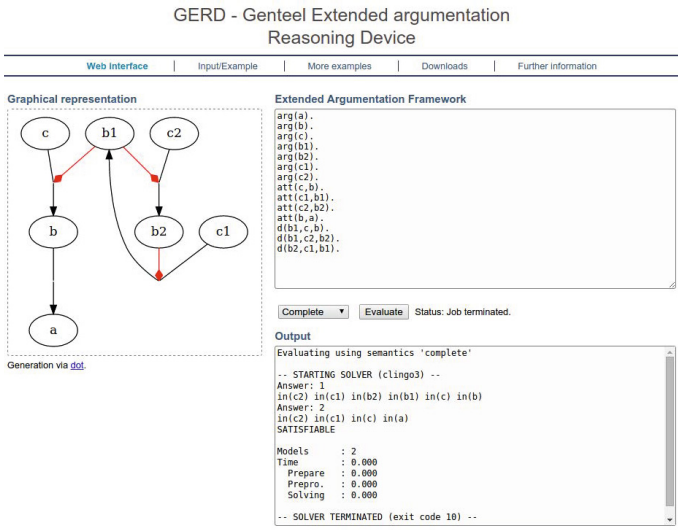


Fig. 2. Web-interface for ASP encodings of EAF semantics

3.3 Propositional Encoding

Our alternative characterization is not only useful in the context of ASP. To exemplify this we encode admissible semantics in terms of propositional logic. Notice that such encodings are the basis to generalize several (implementation) approaches studied for abstract argumentation, like for using SAT and QBF-solvers [4,16], monadic second order logic encodings [14], and approaches using iterative SAT-calls [8,13].

The idea of propositional logic encodings is to give a formula such that the models of the formula correspond to the extensions of the EAF. Given an EAF $F = (A, R, D)$ for each $x \in A$ we introduce a variable a_x encoding that x is in the extension S, i.e. x is in the extension iff a_x is true in the corresponding model. Then for each pair $(y, z) \in R$ we introduce variables $r_{y,z}$ encoding that $y \rightarrowtail^S z$. The truth-values of $r_{y,z}$ can be defined in terms of a_x.

$$\varphi_r = \bigwedge_{(x,(y,z))\in D} (\neg a_x \vee \neg r_{y,z}) \wedge \bigwedge_{(y,z)\in R} (r_{y,z} \vee (\bigvee_{(x,(y,z))\in D} a_x))$$

The first part saying that for each attack $(x, (y, z))$ either $x \notin S$ or $y \not\rightarrowtail^S z$. The second part is the reverse direction saying that either $y \rightarrowtail^S z$ or there is an attack $(x, (y, z))$ with $x \in S$. We are now ready to encode conflict-freeness.

$$\varphi_{cf} = \bigwedge_{(x,y)\in R} (\neg a_x \vee \neg a_y \vee \neg r_{x,y}) \wedge \bigwedge_{(x,y),(y,x)\in R} (\neg a_x \vee \neg a_y)$$

The first part says that for each $(x, y) \in R$ either $x \notin S$ or $y \notin S$ or the attack must be canceled by S. The second part encodes the condition that mutually conflicting arguments cannot be in the same conflict-free set.

To test admissibility we need a reinstatement set $\mathcal{RS}$ which is encoded by variables $rs_{y,z}$, i.e. the attack $(y,z) \in R$ is in the reinstatement set $\mathcal{RS}$ iff $rs_{y,z}$ is true in the corresponding model.

$$\varphi_{\mathcal{RS}} = \bigwedge_{(y,z)\in R} ((\neg rs_{y,z} \vee a_y) \wedge (\neg rs_{y,z} \vee r_{y,z})) \wedge \bigwedge_{(x,(y,z))\in D} (\neg rs_{y,z} \vee \bigvee_{(z',x)\in R} rs_{z',x})$$

The first part stating that if an attack (y,z) is in $\mathcal{RS}$ then $y \in S$ and $y \rightarrowtail^S z$. The second one says that for each $(x,(y,z))$ either there is an attack (z',x) in $\mathcal{RS}$ or (y,z) cannot be in $\mathcal{RS}$.

Finally we can encode the condition for a set S defending its arguments.

$$\varphi_{def} = \bigwedge_{(y,z)\in R} (\neg a_z \vee \neg r_{y,z} \vee \bigvee_{(x,y)\in R} rs_{x,y})$$

So for each attack (y,z) either $z \notin S$, the attack is canceled by S or y is counter attacked by an attack in $\mathcal{RS}$.

Now it is straight forward to show the following proposition.

Proposition 4. *Consider the function* $Ext(M) = \{x \in A \mid a_x \in M\}$ *mapping models to extensions. For any EAF* F *we have* $adm(F) = \{Ext(M) \mid M \text{ is model of } \varphi_r \wedge \varphi_{cf} \wedge \varphi_{\mathcal{RS}} \wedge \varphi_{def}\}$.

4 Complexity and Expressiveness of EAFs

In this section we first use our characterization of acceptance to answer an open complexity-question from [12]. Second, given that the complexity of the main reasoning tasks in EAFs and AFs coincide and complexity is often considered as an indicator for expressiveness one might expect that they have the same expressiveness. We answer this negatively by showing that for each semantics considered in this paper, except grounded, EAFs are more expressive than AFs.

4.1 Complexity of Grounded-Scepticism

Modgil [18] observed that in EAFs the grounded extension is not always contained in all the preferred extensions. This is in contrast to Dung's AFs where this is always the case and grounded semantics can be seen as strictly more skeptical than skeptical preferred reasoning, i.e. than considering the arguments that are contained in all preferred extensions. Dunne et al. [12] introduced the computational problem of Grounded-Scepticism, i.e. deciding whether the grounded extension is contained in all the preferred extensions, and gave a coNP lower bound but left the exact complexity open.

Theorem 1. Grounded-Scepticism *is* Π_2^{P}*-complete.*

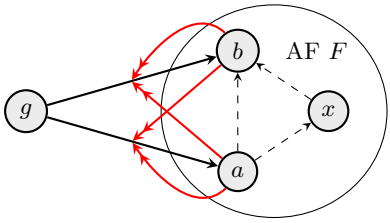


Fig. 3. The AF F' from the proof of Theorem 1, for $A = \{a, b, x\}$

Proof. We first show *membership in* Π_2^{P}. This is by a Σ_2^{P} algorithm for disproving that the grounded extension is contained in each preferred extension. This algorithm first computes the grounded extension G which is in P [12] and then guesses a preferred extension E. Then the NP-oracle is used to verify that E is a preferred extension and finally $G \subseteq E$ is tested.

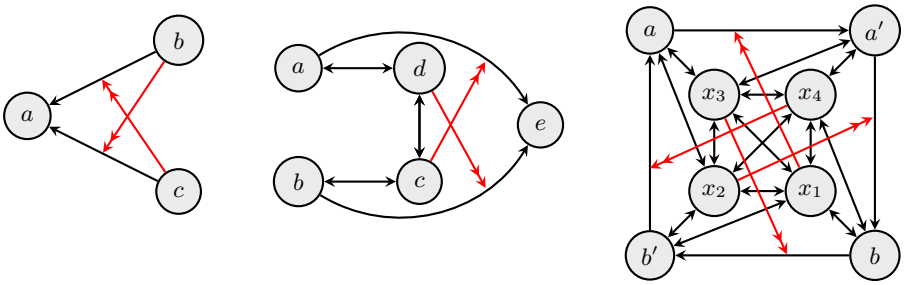
To obtain *hardness* we give a reduction from the Π_2^{P}-hard problem $\mathsf{Skept}_{prf}^{AF}$, that is deciding whether an argument $x \in A$ is skeptically accepted w.r.t. *prf* in Dung AFs [10]. To this end consider an instance $F = (A, R), x \in A$ of $\mathsf{Skept}_{prf}^{AF}$. W.l.o.g. we can assume that $(x, x) \notin R$. We construct an EAF $F' = (A', R', D')$ with $A' = A \cup \{g\}$, $R' = R \cup \{(g, a) \mid a \in A \setminus \{x\}\}$ and $D' = \{(b, (g, a)) \mid a, b \in A \setminus \{x\}\}$ (see also Figure 3). Clearly F' can be constructed in polynomial time.

To complete the proof we next show that x is skeptically accepted in F iff $grd(F') \subseteq E$ for each $E \in prf(F')$. To this end we show that $com(F') = \{\{g, x\}\} \cup \{E \cup \{g\} \mid E \in com(F)\}$. First as g is not attacked at all it has to be contained in each complete extension. Considering $S = \{g\}$ we have that $\mathcal{RS}[S] = \{(g, a) \mid a \in A \setminus \{x\}\}$ and thus that g defends x and thus x must be in the grounded extension. Now consider $S = \{g, x\}$. Still $\mathcal{RS}[S] = \{(g, a) \mid a \in A \setminus \{x\}\}$ and none of the $a \in A \setminus \{x\}$ is acceptable as a is not defended against (g, a). Hence, $\{g, x\}$ is the grounded extension. Next consider an S with $S \cap (A \setminus x) \neq \emptyset$. Then $\rightarrowtail^S$ corresponds to R. As no attack in R is attacked by D' we have that $E \cup \{g\}$ is complete iff $E \in com(F)$.

By the above we have that either (i) $prf(F') = \{E \cup \{g\} \mid E \in prf(F)\}$ if there is an $E \in prf(F)$ with $x \in E$, or (ii) $prf(F') = \{\{g, x\}\} \cup \{E \cup \{g\} \mid E \in prf(F)\}$ otherwise. In the former $\{g, x\}$ is contained in all preferred extensions of F' iff x was skeptically accepted in F and in the latter $\{g, x\}$ is not contained in all preferred extensions but also x was not skeptically accepted in F. Hence, $\{g, x\}$ is contained in all preferred extensions of F' iff x is skeptically accepted in F. □

4.2 Expressiveness of EAFs

Recently the expressiveness of the most prominent semantics of AFs was studied in terms of realizability [11]. A collection of sets of arguments $\mathbb{S}$, frequently called extension-set in the remainder of this section, is said to be realizable under a semantics σ, if there exists some AF F such that the σ-extensions of F coincide with $\mathbb{S}$, i.e. $\sigma(F) = \mathbb{S}$. In the following we show that the additional modelling



(a) Conflict-free. (b) Stable. (c) Admissible, preferred, and complete.

Fig. 4. EAFs witnessing the increased expressive power compared to AFs

power of EAFs also gives rise to increased expressiveness. This means that for every semantics under consideration, except grounded, there are extension-sets obtained by some EAF which do not have an AF as syntactic counterpart. We show EAFs with sets of extensions which cannot be realized under the corresponding AF-semantics in the following example (see also Figure 4).

Conflict-Free Sets: Given an arbitrary AF F, it holds that $cf(F)$ is downward closed, that is for every $E \in cf(F)$ also $E' \in cf(F)$ for each $E' \subseteq E$. This is, as already pointed out in [12], not necessarily true in EAFs. For example, the conflict-free sets of the EAF F_1 in Figure 4a coincide with $\{\emptyset, \{a\}, \{b\}, \{c\}, \{b, c\}, \{a, b, c\}\}$, which cannot be the collection of conflict-free sets of any AF. Observe that for $E = \{a, b, c\}$ both $\{a, b\} \subseteq E$ and $\{a, c\} \subseteq E$, but neither one of those sets is a conflict-free set of F_1. This comes by the fact that the success of attacks can be conditioned by the presence of arguments.

Stable Semantics: It was shown in [11] that for every AF $F = (A, R)$, the stable extensions of F, denoted by $\mathbb{S}$, form a tight set, i.e. the following holds: $\mathbb{S} \subseteq max_{\subseteq}\{S \subseteq A \mid \forall a, b \in S\ \exists T \in \mathbb{S} : \{a, b\} \subseteq T\}$. One can check that this condition does not hold for the extension-set $\mathbb{T} = \{\{a, b\}, \{a, c, e\}, \{b, d, e\}\}$. Hence there is no AF F with $stb(F) = \mathbb{T}$. On the other hand, the EAF F_2 depicted in Figure 4b has exactly $\mathbb{T}$ as stable extensions.

Preferred Semantics: The preferred semantics is among the most expressive semantics in AFs. For a collection of sets of arguments $\mathbb{S}$, the property called adm-closed is decisive for *prf*-realizability [11]: For each $A, B \in \mathbb{S}$ such that $A \cup B \notin \mathbb{S}$ (for *prf* just $A \neq B$) there have to be some $a, b \in (A \cup B)$ with $\nexists C \in \mathbb{S} : \{a, b\} \subseteq C$. Now consider the extension-set $\mathbb{U} = \{\{a, b\}, \{a', b'\}, \{a, a', x_1\}, \{a', b, x_2\}, \{b, b', x_3\}, \{a, b', x_4\}\}$ and observe that $A = \{a, b\}$ and $B = \{a', b'\}$ violate the condition. Each pair of arguments in $(A \cup B)$ occurs together in some element of $\mathbb{U}$ and is therefore necessarily without conflict in every AF trying to realize $\mathbb{U}$. On the other hand, we can again find an EAF realizing $\mathbb{U}$ under the preferred semantics, namely F_3 shown in Figure 4c, where conflicts are resolved by attacks from the x_i-arguments.

Admissible and Complete Semantics: Finally one can also show that $adm(F_3)$ (resp. $com(F_3)$) are not realizable by AFs under the admissible (resp. complete) semantics, indicating the increase in expressiveness for admissible and complete semantics. Towards a contradiction assume that $adm(F_3)$ (resp. $com(F_3)$) could be realized by an AF F under admissible (resp. complete) semantics. Then the preferred extensions $prf(F)$ of F are just the $\subseteq$-maximal sets in $adm(F_3)$ (resp. $com(F_3)$) and thus $prf(F) = \mathbb{U}$. However, this contradicts the observation from above that $\mathbb{U}$ is not *prf*-realizable in AFs.

5 Conclusion

In this work we revisited Modgil's extended argumentation frameworks [18], an appealing approach to incorporate preferences in abstract argumentation formalisms. We provided a different, yet equivalent, characterization of acceptance in EAFs, of which we made use of in reductions to two well-established formalisms. First we presented ASP encodings for all semantics together with an implementation in the online tool GERD. Second we encoded admissible semantics in terms of propositional logic as a basis for implementation approaches such as SAT- and QBF-solving. Moreover, we addressed a problem which was left open in the complexity analysis of EAFs [12] by showing that deciding whether the grounded extension is contained in all preferred extensions is Π_2^P-complete for EAFs. Finally we showed that the additional modelling capabilities within EAFs give rise to higher expressiveness for all but the grounded semantics.

Making use of the propositional encoding of admissible semantics in an (iterative) SAT-based implementation of EAF reasoning tasks is an obvious direction of future work. Moreover, the performance of our ASP-based implementation could be compared to labeling-based algorithms [21] in an empirical evaluation. Finally, the connection of EAFs to ADFs [7], a very recent and general argumentation formalism, should be explored, particularly by providing an efficient translation from EAFs to ADFs.

Acknowledgements. We express our gratitude to Gerd Brewka, to whom this Festschrift is dedicated. Each of the authors visited Gerd's group in Leipzig in the course of their work, which has led to many ongoing and fruitful collaborations and discussions. Insights gained through these visits have been, and continue to be, influential for our works.

We further thank Günther Charwat and Andreas Pfandler for their support for developing the web front-end GERD, Gerald Weidinger for his contributions to earlier versions of the ASP encodings and Pietro Baroni for his helpful comments on an earlier version of this paper.

This research has been supported by the Austrian Science Fund (FWF). Thomas Linsbichler's work has been funded by FWF project I1102 and Johannes Wallner's work has been funded by FWF project P25521.

References

1. Amgoud, L., Cayrol, C.: A reasoning model based on the production of acceptable arguments. Ann. Math. Artif. Intell. 34(1-3), 197–215 (2002)
2. Baroni, P., Caminada, M.W.A., Giacomin, M.: An introduction to argumentation semantics. Knowledge Eng. Review 26(4), 365–410 (2011)
3. Bench-Capon, T.J.M.: Persuasion in practical argument using value-based argumentation frameworks. J. Log. Comput. 13(3), 429–448 (2003)
4. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: Proc. NMR, pp. 59–64 (2004)
5. Brewka, G., Woltran, S.: Abstract Dialectical Frameworks. In: Proc. KR 2010, pp. 102–111. AAAI Press (2010)
6. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (2011)
7. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract Dialectical Frameworks Revisited. In: Proc. IJCAI, pp. 803–809. AAAI Press / IJCAI (2013)
8. Cerutti, F., Dunne, P.E., Giacomin, M., Vallati, M.: Computing preferred extensions in abstract argumentation: A SAT-based approach. In: Black, E., Modgil, S., Oren, N. (eds.) TAFA 2013. LNCS, vol. 8306, pp. 176–193. Springer, Heidelberg (2014)
9. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77(2), 321–358 (1995)
10. Dunne, P.E., Bench-Capon, T.J.M.: Coherence in finite argument systems. Artif. Intell. 141(1/2), 187–203 (2002)
11. Dunne, P.E., Dvořák, W., Linsbichler, T., Woltran, S.: Characteristics of multiple viewpoints in abstract argumentation. In: Proc. KR, pp. 72–81. AAAI Press (2014)
12. Dunne, P.E., Modgil, S., Bench-Capon, T.J.M.: Computation in extended argumentation frameworks. In: Proc. ECAI, pp. 119–124. IOS Press (2010)
13. Dvořák, W., Järvisalo, M., Wallner, J.P., Woltran, S.: Complexity-sensitive decision procedures for abstract argumentation. Artif. Intell. 206, 53–78 (2014)
14. Dvořák, W., Szeider, S., Woltran, S.: Abstract argumentation via monadic second order logic. In: Hüllermeier, E., Link, S., Fober, T., Seeger, B. (eds.) SUM 2012. LNCS, vol. 7520, pp. 85–98. Springer, Heidelberg (2012)
15. Egly, U., Gaggl, S.A., Woltran, S.: Answer-Set Programming Encodings for Argumentation Frameworks. Argument and Computation 1(2), 147–177 (2010)
16. Egly, U., Woltran, S.: Reasoning in argumentation frameworks using Quantified Boolean Formulas. In: Proc. COMMA, pp. 133–144. IOS Press (2006)
17. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam Answer Set Solving Collection. AI Communications 24(2), 105–124 (2011)
18. Modgil, S.: Reasoning about preferences in argumentation frameworks. Artif. Intell. 173(9-10), 901–934 (2009)
19. Modgil, S., Prakken, H.: A general account of argumentation with preferences. Artif. Intell. 195, 361–397 (2013)
20. Niemelä, I.: Logic Programming with Stable Model Semantics as a Constraint Programming Paradigm. Ann. Math. Artif. Intell. 25(3-4), 241–273 (1999)
21. Nofal, S., Dunne, P.E., Atkinson, K.: Towards experimental algorithms for abstract argumentation. In: Proc. COMMA, pp. 217–228. IOS Press (2012)