Deletion-Backdoors for Argumentation Frameworks with Collective Attacks

Wolfgang Dvořák[1], Matthias König[1,*] and Stefan Woltran[1]

[1]TU Wien, Institute of Logic and Computation

Abstract

Analyzing computational aspects of argumentation has the ultimate goal to find efficient tools to reason in an argumentative setting. In particular, exploiting islands of tractability leads to enhancements of our ability to create such tools. In this work, we identify as such an island of tractability *deletion-backdoors* for argumentation frameworks with collective attacks (SETAFs). A backdoor is the part of a problem instance, the removal of which leads to a simple structure. If we can find such a backdoor and guess the solution on this part (in the context of argumentation: extensions), the rest of the solution follows almost effortlessly. In terms of complexity analysis, this means for constant backdoor sizes, general argumentation tasks become efficiently solvable—they are fixed-parameter tractable. In this work, we generalize the respective techniques that are known for the special case of Dung-style argumentation frameworks (AFs) and show that they also apply to the more general case of SETAFs. In addition, we can show an improvement in the asymptotic runtime compared to earlier approaches for AFs. Along the way, we point out similarities and interesting situations arising from the more general setting.

Keywords

Abstract Argumentation, Backdoors, FPT, SETAF, Complexity

1. Introduction

In the last decades of argumentation research, several formalizations have been considered for suitable abstractions of argumentation processes—Dung's approach of *abstract argumentation frameworks* (AFs) [1] is nowadays one of the most widely used such formalisms. In AFs, arguments are abstract entities represented by nodes in a directed graph, the edges form the attack relation. Several generalizations have been proposed in order to achieve more expressiveness or to be able to instantiate knowledge bases more easily or intuitively. One such generalization is the addition of *collective attacks* [2], the resulting class of frameworks is referred to as SETAFs. Reasoning in AFs is computationally expensive—in order to still obtain efficient runtimes in special cases restrictions of the graph structure have been investigated [3, 4, 5]. While in the general case the same (mostly intractable) complexity upper bounds hold for

SAFA'22: Fourth International Workshop on Systems and Algorithms for Formal Argumentation 2022, September 13, 2022, Cardiff, Wales, United Kingdom

*Corresponding author.

wolfgang.dvorak@tuwien.ac.at (W. Dvořák); matthias.koenig@tuwien.ac.at (M. König); stefan.woltran@tuwien.ac.at (S. Woltran)
https://dbai.tuwien.ac.at/staff/dvorak/ (W. Dvořák); https://dbai.tuwien.ac.at/user/mkoenig/ (M. König); https://dbai.tuwien.ac.at/staff/woltran/ (S. Woltran)
0000-0002-2269-8193 (W. Dvořák); 0000-0003-0205-0039 (M. König); 0000-0003-1594-8972 (S. Woltran)

SETAFs [6], tractable graph classes have only recently been brought to the attention of the research community [7, 8]. The analysis of computational aspects of SETAFs by parameterized means just started by investigations of the parameter treewidth [9]. A particular challenge hereby is to extend structural parameters (that work well for the simple structure of AFs), to the hypergraph structure SETAFs provide. In fact, it turned out that while for certain generalizations the parameterized tractability results of AFs carry over to SETAFs, other scenarios give rise to interesting behavior that leads to most problems remaining intractable.

In this paper, we use the tools of parameterized complexity analysis and focus on the parameter of minimum-size deletion-backdoors. The backdoor-concept is used in different contexts such as constraint satisfaction problems (CSP), satisfiability checking (SAT), answer set programming (ASP) (see e.g. [10, 11, 12, 13] for recent work), and has also been investigated in the context of AFs [14]. Intuitively, a backdoor is one part of an instance that "makes it hard to solve". In this work, we focus on *deletion-backdoors*, i.e. if this backdoor is removed from the instance, the remaining (sub-)instance is computationally easy. In the context of formal argumentation this means that a framework belongs to a tractable class (such as acyclicity) after removing certain arguments. This means we can utilize the tractability results of these easy classes for general frameworks without restrictions, as arbitrary distances to the tractable fragments are allowed. If this distance is small (constant), we can reason in polynomial time [14].

Our main contributions are as follows: (i) we introduce the concept of deletion-backdoors for SETAFs utilizing the concept of primal-graphs for SETAFs [7]; (ii) we present an algorithm to reason w.r.t. the common admissibility based semantics stable, preferred, complete, and semi-stable, to exploit given backdoors for the fragments acyclicity and even-cycle-freeness (as introduced for SETAFs in [7]), thereby improving the runtime of the best known backdoor-approach for AFs [14]; and (iii) show that our introduced algorithm is optimal unless the Strong Exponential Time Hypothesis is false.

2. Background

We start by recalling the definition of argumentation frameworks with collective attacks (SETAFs) [2] and identify argumentation frameworks (AFs) [1] as a special case.

Definition 1. *A SETAF is a pair $SF = (A, R)$ where A is a finite set of* arguments*, and $R \subseteq (2^A \setminus \{\emptyset\}) \times A$ is the* attack relation*. For an attack $(T,h) \in R$ we call T the* tail *and h the* head *of the attack. SETAFs (A,R), where for all $(T,h) \in R$ it holds that $|T| = 1$, amount to (standard Dung) AFs. In that case, we usually write (t,h) to denote the set-attack $(\{t\},h)$. For $S \subseteq A$, we say S* attacks *an argument $a \in A$ if there is an attack $(T,a) \in R$ with $T \subseteq S$. We use S^+_R to denote the set $\{a \mid S \text{ attacks } a\}$ and define the* range *of S (w.r.t. R), denoted $S^\oplus_R$, as the set $S \cup S^+_R$.*

The well-known notions of conflict and defense from classical Dung-style AFs naturally generalize to SETAFs.

Definition 2. *Let $SF = (A,R)$ be a SETAF. A set $S \subseteq A$ is* conflicting *in SF if S attacks a for some $a \in S$. $S \subseteq A$ is* conflict-free *in SF, if S is not conflicting in SF, i.e. if $T \cup \{h\} \not\subseteq S$ for each $(T,h) \in R$. $cf(SF)$ denotes the set of all conflict-free sets in SF.*

Table 1
Complexity for AFs and SETAFs (C-c denotes completeness for class C).

	adm	*com*	*pref*	*stb*	*sem*
$Cred_\sigma$	NP-c	NP-c	NP-c	NP-c	Σ_2^{P}-c
$Skept_\sigma$	trivial	P-c	Π_2^{P}-c	coNP-c	Π_2^{P}-c

Definition 3. *Let $SF = (A,R)$ be a SETAF. An argument $a \in A$ is* defended *(in SF) by a set $S \subseteq A$ if for each $B \subseteq A$, such that B attacks a, also S attacks B in SF. A set $T \subseteq A$ is defended (in SF) by S if each $a \in T$ is defended by S (in SF).*

The semantics we study in this work are the admissible, complete, preferred, stable, and semi-stable semantics, which we will abbreviate by *adm*, *com*, *pref*, and *stb*, and *sem*, respectively [2, 6, 15]. $\sigma(SF)$ denotes the set of *extensions* of SF under semantics σ.

Definition 4. *Given a SETAF $SF = (A,R)$ and a conflict-free set $S \in cf(SF)$. Then,*

- *$S \in adm(SF)$, if S defends itself in SF,*
- *$S \in com(SF)$, if $S \in adm(SF)$ and $a \in S$ for all $a \in A$ defended by S,*
- *$S \in pref(SF)$, if $S \in adm(SF)$ and there is no $T \in adm(SF)$ s.t. $T \supset S$,*
- *$S \in stb(SF)$, if $S_R^{\oplus} = A$,*
- *$S \in sem(SF)$, if $S \in adm(SF)$ and $\nexists T \in adm(SF)$ s.t. $T_R^{\oplus} \supset S_R^{\oplus}$.*

The relationship between the semantics has been clarified in [6, 2, 15] and matches with the relations between the semantics for Dung AFs, i.e. for any SETAF SF:

$$stb(SF) \subseteq sem(SF) \subseteq pref(SF) \subseteq com(SF) \subseteq adm(SF) \subseteq cf(SF)$$

We assume the reader to have basic knowledge in computational complexity theory[1], in particular we make use of the complexity classes P (polynomial time), NP (non-deterministic polynomial time), coNP, Σ_2^{P}, and Π_2^{P}. For a SETAF $SF = (A,R)$ and an argument $a \in A$, we consider the standard reasoning problems (under semantics σ):

- Credulous acceptance $Cred_\sigma$: Is a contained in at least one σ extension of SF?
- Skeptical acceptance $Skept_\sigma$: Is a contained in all σ extensions of SF?

The complexity landscape of SETAFs coincides with that of Dung AFs and is depicted in Table 1. As SETAFs generalize Dung AFs the hardness results for Dung AFs [16] carry over to SETAFs, also the same upper bounds hold for SETAFs [6].

For a more fine-grained complexity analysis we also make use of the complexity class FPT (fixed-parameter tractability): a problem is fixed-parameter tractable w.r.t. a parameter if there is an algorithm with runtime $O(f(p) \cdot \mathsf{poly}(n))$, where n is the size of the input, p is an integer describing the instance called the *parameter value*, and $f(\cdot)$ is an arbitrary computable function independent of n (typically at least exponential in p). We also make use of the class XP. A problem is in XP w.r.t. a parameter if there is an algorithm with runtime $O(n^{O(p)})$, where n is the size of the input, and p is the *parameter value*. We have that $\mathsf{P} \subseteq \mathsf{FPT} \subseteq \mathsf{XP}$.

[1]For a gentle introduction to complexity theory in the context of formal argumentation, see [16].

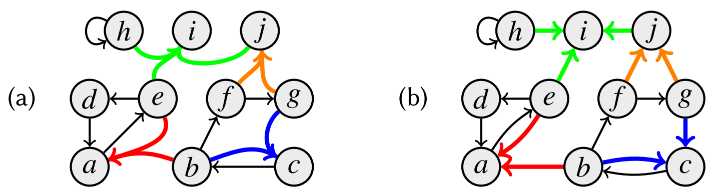

Figure 1: (a) The SETAF SF and (b) its primal graph $\texttt{Primal}(SF)$. Collective attacks and their corresponding edges in the primal graph are highlighted.

3. Graph Properties and Backdoors for SETAFs

The underlying structure of SETAFs is a *directed hypergraph*, which makes it hard to apply certain notions of graph properties. We can avoid these issues by utilizing "standard" directed graphs to describe SETAFs, and analyze graph properties (such as backdoors). There are different such directed graph-representations [7, 8]; we focus on the *primal graph*, as it arguably is the most intuitive way to embed SETAFs into directed graphs and fits our purposes best. An example for a primal graph is given in Figure 1; the corresponding SETAF will serve as a running example throughout the paper. We utilize the primal graph to define *primal-backdoors*.

Definition 5 ([7]). *Let $SF = (A,R)$ be a SETAF. The* primal graph *of SF is defined as* $\texttt{Primal}(SF) = (A,R')$, *where* $R' = \{(t,h) \mid (T,h) \in R, t \in T\}$. *A* (primal-)cycle *of length n in SF is a non-repeating sequence $(a_1,a_2,\dots,a_n)$ of arguments such that for each $1 \leq i \leq n-1$ there is $(A_i,a_{i+1}) \in R$ such that $a_i \in A_i$, and there is $(A_n,a_1) \in R$ with $a_n \in A_n$. Equivalently, a primal cycle corresponds to a directed cycle in* $\texttt{Primal}(SF)$. *SETAFs SF with*

- *no (primal-)cycles are called* (primal-)acyclic*,*
- *no (primal-)cycles of even length are called* even-(primal-)cycle-free*,*

It is easy to see that several SETAFs can map to the same primal graph. However, restrictions on the primal graph are often useful to obtain computational speedups for otherwise hard problems [7, 8, 9]. We denote the classes of acyclic, even-cycle-free SETAFs by $\mathsf{ACYC}, \mathsf{NOEVEN}$, respectively. Note that in the special case of AFs these classes coincide with the standard definitions, i.e., they properly generalize the standard case. For AFs, also the classes of bipartite and symmetric frameworks have been investigated. However, while finding suitable backdoors to these classes was shown to be parameterized-tractable, reasoning in AFs with constant backdoor-size to these frameworks remains hard [14]. Even though there are generalizations of symmetry and bipartiteness for SETAFs where reasoning also becomes tractable [7], as these classes generalize the respective properties of AFs, the hardness-results for backdoor evaluation carry over to the general SETAFs. Hence, we focus on the fragments ACYC and NOEVEN. On AFs, all of these classes are considered as the *tractable fragments of argumentation*, as for many semantics there are efficient algorithms to reason in AFs that belong to one of these classes [3, 4, 5]. However, these fragments restrict the possible structure of an AF. In order to exploit the speedup while still keeping the full expressiveness, the requirements have been weakened to allow for arbitrary *distance* to these fragments. We generalize this so-called *backdoor*-approach by Dvořák et al. [14].

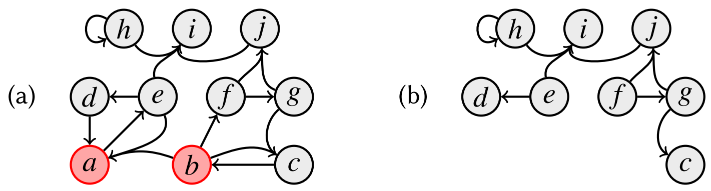

Figure 2: (a) The SETAF $SF = (A,R)$ with the highlighted NOEVEN-backdoor $B = \{a,b\}$ and (b) the "remaining" framework $SF{\downarrow_{A\setminus B}}$ after the deletion of B.

For this, we need the notion of the *projection*. Intuitively, a SETAF $SF = (A,R)$ projected to a set of arguments $S \subseteq A$ (we write $SF{\downarrow_S}$) can be seen as the result of removing the arguments outside of S while retaining as much of the original structure as possible.

Definition 6. *Let $SF = (A,R)$ be a SETAF and let C be a class of SETAFs. We call a set $B \subseteq A$ a C-backdoor if $SF{\downarrow_{A\setminus B}}$ belongs to C, where*

$$SF{\downarrow_{A\setminus B}} = (A \setminus B, \{(T \setminus B, h) \mid (T,h) \in R, T \setminus B \neq \emptyset, h \in A \setminus B\}).$$

We denote the size of a smallest C-backdoor by $\mathrm{bd}_{\mathsf{C}}(SF)$.

Figure 2 illustrates the idea of backdoors: the set $B = \{a,b\}$ is a NOEVEN-backdoor for the SETAF $SF = (A,R)$ of size 2. It is easy to see in Figure 2(b) that in the SETAF SF projected to $A \setminus B$, i.e., the remaining framework $SF{\downarrow_{A\setminus B}}$ after removing the backdoor, has no directed cycles of even length in its primal graph. We want to highlight that in $SF{\downarrow_{A\setminus B}}$ the attack $(\{b,g\},c)$ from SF is not deleted as a whole, but "reduced" to the attack (g,c). It can easily be verified that indeed, B is a smallest NOEVEN-backdoor. For any ACYC-backdoor, furthermore the argument h has to be included (in general, every ACYC-backdoor is a NOEVEN-backdoor but not vice versa). Hence, for our example we have $\mathrm{bd}_{\mathsf{ACYC}}(SF) = 3$ and $\mathrm{bd}_{\mathsf{NOEVEN}}(SF) = 2$.

It was shown that reasoning in AFs w.r.t. stable, complete, and preferred semantics is tractable for the fragments $\mathsf{C} \in \{\mathsf{ACYC}, \mathsf{NOEVEN}\}$ if $\mathrm{bd}_{\mathsf{C}}(SF)$ is fixed [14], we generalize these results in Section 4. We can efficiently *recognize* these classes for AFs [14] and *find* backdoors of bounded size. As we defined the fragments for SETAFs on the primal graph, the respective results from AFs immediately carry over to our setting.

Corollary 7. *Let SF be a SETAF.*

1. *We can recognize whether SF belongs to ACYC or NOEVEN in polynomial time.*
2. *We can find a NOEVEN-backdoor of size at most p in time $|SF|^{O(p)}$ (or if no such backdoor exists correctly detect so).*
3. *We can find an ACYC-backdoor of size at most p in time $f(p) \cdot \mathrm{poly}|SF|$ (or if no such backdoor exists correctly detect so).*

This means, that the fragment ACYC is in principle suitable for FPT algorithms, while for NOEVEN we aim for XP algorithms (however, it is principally possible that faster algorithms for finding NOEVEN-backdoors exist).

4. Backdoor Evaluation

In this section we tackle the evaluation of backdoors of constant size, i.e., assume we are *given* a backdoor, how can we efficiently compute the extensions. We adapt the results from [14] and generalize the therein mentioned notions such as partial labelings and propagation algorithms to SETAFs. Our approach however differs from the one in [14] as it is tailored to the fragments ACYC and NOEVEN. This allows us to give a tighter upper bound for the runtime: $O(2^p \cdot \mathsf{poly}(n))$ instead of $O(3^p \cdot \mathsf{poly}(n))$ (p is the size of the given backdoor and n is the size of the instance). Our improved algorithm is applicable to SETAFs as well as AFs, as the latter is just a special case of the former.

The basic idea of the backdoor evaluation algorithm is to guess parts of an extension on the arguments in the backdoor, and then cleverly propagate this guess to the rest of the instance. As the remaining instance is even-cycle-free, there are no supportive cycles to consider in the propagation step. Algorithm 1 captures the whole process; in the following we will verify its correctness and give an intuition for each step. If we provide for a SETAF *SF* a NOEVEN-backdoor *B*, Algorithm 1 returns a set of admissible candidates $pref^*(SF,B)$ for preferred extensions.

Algorithm 1: Computation of $pref^*(SF,B)$

```
1  $pref^*(SF,B) \leftarrow \emptyset$;
2  foreach $I \subseteq B$ do
3      let $\lambda$ be a partial labeling on $B$;
4      set $\lambda(a) = in$ for $a \in I$;
5      set $\lambda(a) = out$ for $a \in B \setminus I$;
6      $\lambda^* \leftarrow \mathsf{propagate}_{\mathsf{IO}}(SF,\lambda)$;
7      set $\lambda^*(a) = und$ for $a \in A \setminus DEF(\lambda^*)$;
8      $\lambda^\dagger \leftarrow \mathsf{propagate}_{\mathsf{U}}(SF,\lambda^*)$;
9      if $IN(\lambda^\dagger) \cap B = I$ then
10         $pref^*(SF,B) \leftarrow pref^*(SF,B) \cup \{IN(\lambda^\dagger)\}$;
```

We capture the concept of the partial guess by partial labelings (cf. [14]).

Definition 8. *Let $SF = (A,R)$ be a SETAF and let $X \subseteq A$ be a set of arguments. A* partial labeling *is a function $\lambda : X \to \{in, out, und\}$. By $IN(\lambda)$ we denote the set $\{a \in X \mid \lambda(a) = in\}$ (similarly, $OUT(\lambda), UND(\lambda)$). We write $DEF(\lambda)$ to identify the set X. For a set $S \subseteq X$ we fix the* corresponding partial labeling *on X as*

$$\lambda_S(a) = \begin{cases} in & \text{if } a \in S, \\ out & \text{if } a \in S_R^+, \\ und & \text{else, i.e. } a \in X \setminus S_R^\oplus. \end{cases}$$

A labeling λ is compatible *with a set $S \subseteq A$ if $\forall a \in DEF(\lambda)$ it holds $\lambda(a) = \lambda_S(a)$.*

We consider two "phases" of propagation. In the first, we propagate the *in*/*out* labels of the guess as far as possible. The second phase fixes incorrectly labeled arguments and assured admissibility in the generated labeling. We will show that with this approach we can capture all preferred extensions.

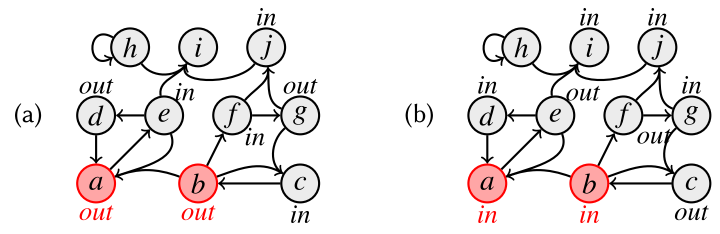

Figure 3: The first propagation propagate$_{\mathrm{IO}}$ (a) λ_1^*, (b) λ_2^*, on the guessed labels for the backdoor arguments a and b (highlighted).

Definition 9. *Let $SF = (A, R)$ be a SETAF and λ be a partial labeling on $X \subseteq A$. Consider the following propagation rules for λ:*

1. *set $\lambda^*(a) = out$ if $\exists (T, a) \in R$ with $T \subseteq IN(\lambda^*)$,*
2. *set $\lambda^*(a) = in$ if $\forall (T, a) \in R$ there is a $t \in T$ with $\lambda^*(t) = out$.*

We define propagate$_{IO}(SF, \lambda)$ as the result of initializing λ^* with λ on $DEF(\lambda)$, and then exhaustively applying the rules 1. and 2. to each argument $a \in A \setminus DEF(\lambda)$.

3. *set $\lambda^\dagger(a) = und$ if $\lambda^\dagger(a) = in$ and there is $(T, a) \in R$ s.t. $\nexists t \in T : \lambda^\dagger(t) = out$,*
4. *set $\lambda^\dagger(a) = und$ if $\lambda^\dagger(a) = out$ and there is no $(T, a) \in R$ s.t. $T \subseteq IN(\lambda^\dagger)$.*

We define propagate$_U(SF, \lambda)$ as the result of initializing $\lambda^\dagger$ with λ on $DEF(\lambda)$, and then exhaustively applying the rules 3. and 4. to each argument $a \in IN(\lambda) \cup OUT(\lambda)$.

In the following Lemma 10 we formalize the intuition of propagate$_{\mathrm{IO}}$. We propagate the labels we guessed on B and treat them at this stage as if they are "confirmed", i.e., whenever we have a reason to assume an argument is defended, we add it (by labeling it *in*), whenever we have a reason to assume an argument is defeated, we keep track of this fact (by labeling it *out*). In this stage, no argument will be labeled *und*. We revisit our running example from Figure 2. Assume we guess $\lambda_1(a) = \lambda_1(b) = out$ for the backdoor $\{a, b\}$. The labeling λ_1^* after exhaustively applying rules 1. and 2. is depicted in Figure 3(a). In general, we will incorrectly label arguments: as can be seen, the argument a is labeled *out*, but there is no attack towards a that verifies this label. Now assume we guess $\lambda_2(a) = \lambda_2(b) = in$. One can see in Figure 3(b) that there is a problem with the propagated labeling λ_2^*: the arguments d and a are both labeled *in*, effectively causing a conflict. Though we will correct this problem in the second step of the propagation algorithm, we will have to change the label of a from *in* to *und*. We will see that we can actually already stop at this point, as we will then compute (a subset of) the extension we get from the guess $\lambda_3(a) = out, \lambda_3(b) = in$. In the following lemma we establish that we label all arguments $a \in E_R^\oplus$ correctly (i.e., *in* and *out*, resp.).

Lemma 10. *Let $SF = (A, R)$ be a SETAF, let $E \in pref(SF)$, and $B \subseteq A$ a NOEVEN-backdoor for SF. For the input (SF, B) to Algorithm 1, assume in step 2 we choose $I = E \cap B$, let λ be the corresponding partial labeling from steps 4 and 5. Set $\lambda^* =$ propagate$_{IO}(SF, \lambda)$. Then for each $a \in A$:*

(a) if $\lambda^*(a) = in$ then $a \notin E_R^+$,
(b) if $\lambda^*(a) = out$ then $a \notin E$,
(c) if $a \notin DEF(\lambda^*)$ then $a \notin E_R^\oplus$,
(d) $E \subseteq IN(\lambda^*)$, and
(e) $E_R^+ \subseteq OUT(\lambda^*)$.

Proof. We show (a) and (b) by induction on the number of labeled arguments in the construction of λ^*. For the base case $\lambda^* = \lambda$ it is easy to see that all conditions (a) and (b) hold (by assumption we have $OUT(\lambda) = B \setminus IN(\lambda) = (A \cap B) \setminus E$). For the step we consider the rules 1. and 2.:

Assume a is labeled via rule 1., i.e. we set $\lambda^*(a) = out$ for some $a \in A \setminus DEF(\lambda)$. Clearly, (a) is not violated by labeling a as *out*, for (b) we show $a \notin E$. Since we invoked rule 1., there is an attack $(T,a) \in R$ with $T \subseteq IN(\lambda^*)$. By our induction hypothesis we know that (a) holds for each $t \in T$, i.e. $T \cap E_R^+ = \emptyset$. This means E does not defend a against the attack (T,a), and since $E \in pref(SF)$, we get $a \notin E$.

Now assume a is labeled via rule 2., i.e. we set $\lambda^*(a) = in$ for some $a \in A \setminus DEF(\lambda)$. Clearly, (b) is not violated by labeling a as *in*, for (a) we show $a \notin E_R^+$. Since we invoked rule 2., for all attacks $(T,a) \in R$ there is some $t \in T$ with $\lambda^*(t) = out$. By our induction hypothesis we know that (b) holds for each such t, i.e. $t \notin E$. Hence, there can be no attack (T,a) with $T \subseteq E$, i.e., $a \notin E_R^+$.

For (c) assume towards contradiction there is an argument $a_1 \in E$ such that $a_1 \notin DEF(\lambda^*)$. If there is no attack $(T,a_1) \in R$ towards a_1, we would have $\lambda^*(a_1) = in$, hence, there is such an attack. Moreover, there is a $(T,a_1) \in R$ s.t. for no $t \in T$ we have $\lambda^*(t) = out$, otherwise we would have $\lambda^*(a_1) = in$. However, by admissibility of E there is at least one $t_1 \in T \cap E_R^+$. For this t_1 we have $t_1 \in A \setminus DEF(\lambda^*)$, i.e. t_1 is unlabeled (the only other option, the label *in*, violates (b)). Since $t_1 \in E_R^+$, there is an attack $(S,t_1) \in R$, but since t_1 is unlabeled, $S \not\subseteq IN(\lambda^*)$, i.e., there is an $a_2 \in S$ such that a_2 is unlabeled. We have $a_1 \neq a_2$, as this would imply an even-length cycle (a_2,t_1). As for a_2 we can reason in the same way as for a_1, we obtain another unlabeled argument $t_2 \in E_R^+$, and eventually a sequence $a_1,t_1,a_2,t_2,\dots$ of arguments. However, as SF is finite and all a_i are different, eventually there is either (i) an unlabeled argument a_k where no attack (T_k,a_k) towards a_k has an unlabeled $t_k \in T_k$, but then $\lambda^*(a_k) = in$, a contradiction, or (ii) an unlabeled argument t_k where there is no counter-attack (S_k,t_k) with an unlabeled $a_{k+1} \in S_k$, but then $\lambda^*(t_k) = out$, a contradiction. Hence, no such a_1 can exist. Assuming there is an unlabeled argument $t_1 \in E_R^+$ analogously leads to a contradiction.

Finally, from (a) and (c) follows (d) and from (b) and (c) follows (e). □

Lemma 11 shows that the procedure $\text{propagate}_U(SF,\lambda)$ is sound. In particular, it is easy to see that any partial labeling λ^* with $DEF(\lambda^*) = A$ that we give as input to propagate_U will give us an admissible set (rule 3. removes conflicts and both rules 3. and 4. resolve undefended arguments). An argument is incorrectly labeled *in* if it is not defended against each attack; it is incorrectly labeled *out* if it is not attacked from within the set $IN(\lambda)$. By exhaustively applying the propagation rules 3. and 4. we fix these incorrect labels. To illustrate this, we revisit our running example. In Figure 3 we see the resulting labelings λ_1^*, λ_2^* for the backdoor $B = \{a,b\}$ for the guesses (a) $\lambda_1(a) = \lambda_1(b) = out$, and (b) $\lambda_2(a) = \lambda_2(b) = in$. Following Algorithm 1, in step 7 we set the un-labeled argument to *und* and compute propagate_U for both labelings. In

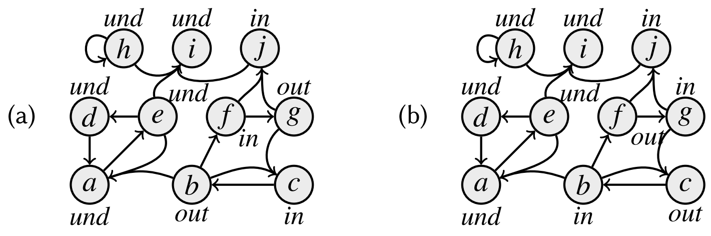

Figure 4: The second propagation propagate$_U$ (a) $\lambda_1^\dagger$, (b) $\lambda_2^\dagger$, computed on the labels from Figure 3(a) and (b), respectively.

Figure 4 we see the resulting labelings $\lambda_1^\dagger, \lambda_2^\dagger$. Our algorithm outputs the sets $\{c, f, j\}, \{b, g, j\}$ respectively, and it can be indeed verified that these sets are preferred in SF. Now consider the guess $\lambda_3(a) = out, \lambda_3(b) = in$. It turns out that $\lambda_3^\dagger = \lambda_2^\dagger$: the fact that both labelings result in the same propagation means in particular that there is no preferred extension E where $\{a, b\} \subseteq E$, as by trying to construct such an extension in (b) we had to correct the label of a. In general, whenever we re-label one of the guessed *in*-labels, we can abort the run of the algorithm on this guess and proceed with a different guess, as we will always compute (a superset of) the thereby "missed" set when we start with the corresponding "correct" labeling on the backdoor arguments.

We show in the following Lemma 11 that for every argument a where we fix the label *und*, a is indeed neither in the corresponding extension nor attacked by it, i.e. $a \notin E_R^\oplus$. This together with the results of Lemma 10 suffices to show that if we guess correctly, the algorithm computes every preferred extension.

Lemma 11. *Let $SF = (A, R)$ be a SETAF, let $E \in pref(SF)$, and $B \subseteq A$ a NOEVEN-backdoor for SF. For the input (SF, B) to Algorithm 1, assume in step 2 we choose $I = E \cap B$, let λ^* be the corresponding propagated partial labeling from step 7 with und labels. Set $\lambda^\dagger = propagate_U(SF, \lambda^*)$. Then $E = IN(\lambda^\dagger)$.*

Proof. We first show by induction on the number of re-labeled arguments (i.e., arguments that are labeled *und* during the construction of $\lambda^\dagger$) that for each $a \in A$ it holds if $\lambda^\dagger(a) = und$ then $a \in A \setminus E_R^\oplus$. The base case where $\lambda^\dagger = \lambda^*$ is covered by Lemma 10(c). For the inductive step consider the rules 3. and 4.:

Assume a is re-labeled by rule 3., i.e. we set $\lambda^\dagger(a) = und$ for some $a \in IN(\lambda^*)$. By Lemma 10(a) we know $a \notin E_R^+$, we show $a \notin E$. Since we invoked rule 3., there is $(T, a) \in R$ s.t. $\nexists t \in T$ with $\lambda^\dagger(t) = out$. By induction hypothesis and Lemma 10(a) and (c), this means $E_R^+ \cap T = \emptyset$, i.e. a is not defended by E against (T, a). By admissibility this means $a \notin E$.

Assume a is re-labeled by rule 4., i.e. we set $\lambda^\dagger(a) = und$ for some $a \in OUT(\lambda^*)$. By Lemma 10(b) we know $a \notin E$, we show $a \notin E_R^+$. Since we invoked rule 4., there is no $(T, a) \in R$ s.t. $T \subseteq IN(\lambda^*)$. By induction hypothesis and Lemma 10(c), this means $T \not\subseteq E$, i.e. a is not attacked by E.

Next, we show $IN(\lambda^\dagger) \in adm(SF)$. By E' we identify the set $IN(\lambda^\dagger)$. Assume towards contradiction there is a conflicting attack $(T, h) \in R$ with $T \cup \{h\} \subseteq E'$. However, this means we would re-label h by rule 3., as there is no $t \in T$ with $\lambda^\dagger(t) = out$, a contradiction. Hence,

$E' \in cf(SF)$. Now assume towards contradiction there is an undefended argument $a \in E'$, i.e., there is an attack $(T,a) \in R$ s.t. for no $t \in T$ there is an attack $(S,t) \in R$ with $S \subseteq E'$. As $a \in E'$, there is some $t \in T$ where either (i) we set $\lambda^*(t) = out$ during the computation of λ^* and did not change the label later, in which case we did not invoke propagation rule 4., and there is indeed an attack $(S,t) \in R$ towards t with $S \subseteq E'$ and a is defended, or (ii) we set $\lambda^*(t) = out$ during the computation of λ^* and during the computation of $\lambda^\dagger$ update it to $\lambda^\dagger(t) = und$, but then if no $t' \in T$ with $\lambda^\dagger(t') = out$ is left, we would have invoked propagation rule 3. for a and set it to *und*, and if such a t' exists where we did not change the label, then a is also defended as in case (i). In all cases we see that indeed a is defended by E' and can conclude $E' \in adm(SF)$.

Finally, by Lemma 10(d) and the formerly established fact that for each $a \in A$ it holds if $\lambda^\dagger(a) = und$ then $a \in A \setminus E_R^\oplus$ we know also $E \subseteq E'$. Since $E' \in adm(SF)$ and we assumed E is preferred, we get $E = E'$. □

These correctness results for the propagation procedures are the basis for the main result of this section, namely the efficient computation of preferred extensions if a suitable backdoor is given.

Theorem 12. *Let $C \in \{NOEVEN, ACYC\}$ be a SETAF class, let $SF = (A,R)$ be a SETAF, and $B \subseteq A$ a C-backdoor for SF with $|B| \leq p$. With Algorithm 1 we can enumerate all σ-extensions in time $2^p \cdot \mathsf{poly}(|SF|)$ for $\sigma \in \{pref, stb, sem\}$ on input (SF,B).*

Proof. Let $E \in pref(SF)$, we show that E is in the output of Algorithm 1, i.e., $E \in pref^*(SF,B)$. Since in step 2 we try all subsets of B, we will try $I = E \cap B$. Lemma 11 ensures $E \in pref^*(SF,B)$. It remains to show that all steps 3-10 can be done in polynomial time w.r.t. $|SF|$. It is easy to see that (assuming we use reasonable data structures) this is the case for steps 3-5, 7, 9, and 10. For step 6 and 8 note that each argument is (re)-labeled at most once, and the check for each propagation rule can clearly be carried out in polynomial time. Hence, the overall runtime is $2^p \cdot \mathsf{poly}(|SF|)$. Finally, we can then check whether the extensions are (range)-subset-maximal or attack every argument not in them, as all candidates are available. Hence, we can remove those sets that are not preferred/stable/semi-stable. □

As we can efficiently enumerate all extensions, also reasoning becomes efficient in this case. This result also carries over to admissible and complete semantics, as the respective credulous acceptance problems coincide with preferred semantics, $Skept_{com}$ is already possible in polynomial time and $Skept_{adm}$ is trivial. We want to highlight that even if we have a NOEVEN-backdoor of size p, there can be up to $O(3^p)$ complete extensions. Clearly we cannot enumerate them with our approach, which is why we capture the preferred extensions (in contrast to the AF approach [14]).

Corollary 13. *Let $C \in \{NOEVEN, ACYC\}$ be a SETAF class, let $SF = (A,R)$ be a SETAF, and $B \subseteq A$ a C-backdoor for SF with $|B| \leq p$. Then we can decide the problems $Cred_\sigma$ and $Skept_\sigma$ in time $2^p \cdot \mathsf{poly}(|SF|)$ for $\sigma \in \{adm, com, pref, stb, sem\}$.*

Combining Corollary 7 and Corollary 13, we immediately obtain the following main result that captures finding and exploiting minimal backdoors for SETAFs.

Theorem 14. *Let SF be a SETAF and let $\sigma \in \{adm, com, pref, stb, sem\}$ be a semantics.*

1. *If $bd_{ACYC}(SF) \leq p$, we can solve $Cred_\sigma$ and $Skept_\sigma$ in FPT w.r.t. parameter p.*
2. *If $bd_{NOEVEN}(SF) \leq p$, we can solve $Cred_\sigma$ and $Skept_\sigma$ in XP w.r.t. parameter p.*

In fact, our algorithm for exploiting a small backdoor that is already given is only in 2^p for the parameter value p, which is an improvement over the existing 3^p approach [14]. This also has implications in practice, as it turns out that finding the minimal backdoor-size often comes with high computational cost: for example, one known FPT algorithm for the computation of minimal NOEVEN-backdoors of size p for directed graphs G has runtime $4^p \cdot p! \cdot \mathsf{poly}(|G|)$ [17], which is often impractical even for small parameter values. Instead, recent implementations focus on finding backdoors *heuristically*, cf. [18]. Hence, our results are also relevant for the development of fast algorithms for AFs, since Theorem 12 & 14 and Corollary 13 also hold in the special case of AFs.

5. Conditional Lower Bounds for Backdoor Evaluation

In this section we show a so-called conditional lower bound [19] for NOEVEN/ACYC-backdoor evaluation, i.e. we show that our algorithm is basically optimal based on some well-known conjecture. The conjecture we are going to use is the *Strong exponential-time hypothesis (SETH)* [20, 21]. We show this for AFs; the result carries over to SETAFs.

Conjecture 1 (Strong Exponential Time Hypothesis (SETH)). *For each $\varepsilon > 0$ there is a k such that k-CNF-SAT on n variables and m clauses cannot be solved in $O(2^{(1-\varepsilon)n} \cdot \mathsf{poly}(n+m))$ time.*

Let p be the parameter for the backdoor size. We will show that any NOEVEN-backdoor evaluation that runs in time $O(2^{(1-\varepsilon)p} \cdot \mathsf{poly}(|F|))$ for AFs F would violate SETH and thus imply a major break-through in the development of SAT algorithms.

Theorem 15. *Let $F = (A,R)$ be an AF and let $C \in \{NOEVEN, ACYC\}$ and let p the size of the given backdoor. There is no $O(2^{(1-\varepsilon)p} \cdot \mathsf{poly}(|A|))$ C-backdoor evaluation algorithm for $Cred_\sigma$ for $\sigma \in \{adm, com, pref, stb, sem\}$ unless SETH is false.*

Proof. Given an instance from SETH, i.e. a CNF formula φ with n variables and m clauses. Let $X = \{x_1, \ldots, x_n\}$ be the set of variables and $C = \{c_1, \ldots, c_m\}$ be the set of clauses appearing in φ. Consider the standard translation [16] from a CNF formula φ to an AF $F_\varphi = (A,R)$ (cf. Figure 5) with $A = \{\varphi\} \cup C \cup X \cup \bar{X}$ and $R = \{(c,\varphi) \mid c \in C\} \cup \{(x,\bar{x}),(\bar{x},x) \mid x \in X\} \cup \{(x,c) \mid x \in c, c \in C\} \cup \{(\bar{x},c) \mid \bar{x} \in c, c \in C\}$. We know that the formula φ is satisfiable iff the argument φ is credulously accepted w.r.t. σ [16]. Moreover, we have that the set X is a C-backdoor of F_φ and has size n.

Towards, a contradiction let us assume we have a $O(2^{(1-\varepsilon)p} \cdot \mathsf{poly}(|A|))$ C-backdoor evaluation algorithm. Then we can decide whether a CNF formula φ is satisfiable as follows: We first construct the AF F_φ (which is polynomial in $n+m$) and then run the C-backdoor evaluation with backdoor X and return the answer for the credulous decision problem as answer to the satisfiability problem. By assumption the latter step has a running time of $O(2^{(1-\varepsilon)n} \cdot \mathsf{poly}(n+m))$. That is, we have a $O(2^{(1-\varepsilon)n} \cdot \mathsf{poly}(n+m))$ algorithm for k-CNF-SAT for arbitrary $k > 1$, which contradicts SETH. □

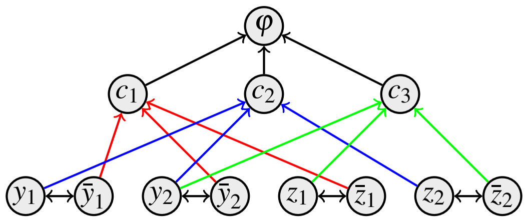

Figure 5: Illustration of the standard translation F_φ for $\varphi = \{\{\bar{y}_1, \bar{y}_2, \bar{z}_1\}, \{y_1, y_2, z_2)\}, \{y_2, z_1, \bar{z}_2\}\}$

6. Conclusion

In this paper, we have applied the well known concept of backdoors for the first time to argumentation frameworks with collective attacks. Instead of simply adapting previous work in this direction, we came up with a genuinely new approach for computing extensions utilizing backdoors to acyclicity and no-even graphs that shows improved runtime bounds compared to the known algorithms for standard AFs. Moreover, we proved that under some complexity-theoretic assumptions further such improvements are not possible. We focused on graph restrictions on the primal graph rather than the SETAF's hypergraph structure, as this allows us to exploit established ideas for finding backdoors—the situation for directed hypergraphs is less diverse. Future work includes the implementation of these techniques and the optimization for heuristics to find suitable backdoors.

Acknowledgments

This research has been supported by the Vienna Science and Technology Fund (WWTF) through project ICT19-065, and by the Austrian Science Fund (FWF) through projects P32830 and Y698.

References

[1] 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.
[2] S. H. Nielsen, S. Parsons, A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments, in: Proceedings of ArgMAS 2006, Springer, 2006, pp. 54–73. doi:10.1007/978-3-540-75526-5_4.
[3] S. Coste-Marquis, C. Devred, P. Marquis, Symmetric argumentation frameworks, in: Proceedings of ECSQARU 2005, volume 3571 of *LNCS*, Springer, 2005, pp. 317–328.
[4] P. E. Dunne, Computational properties of argument systems satisfying graph-theoretic constraints, Artif. Intell. 171 (2007) 701–729.
[5] P. E. Dunne, T. J. M. Bench-Capon, Complexity and Combinatorial Properties of Argument Systems, Technical Report, Dept. of Computer Science, University of Liverpool, 2001.
[6] W. Dvořák, A. Greßler, S. Woltran, Evaluating SETAFs via answer-set programming, in:

Proceedings of SAFA 2018, volume 2171 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 10–21.

[7] W. Dvořák, M. König, S. Woltran, Graph-classes of argumentation frameworks with collective attacks, in: Proceedings of JELIA 2021, volume 12678 of *LNCS*, Springer, 2021, pp. 3–17. doi:10.1007/978-3-030-75775-5_1.

[8] W. Dvořák, M. König, S. Woltran, On the complexity of preferred semantics in argumentation frameworks with bounded cycle length, in: Proceedings of KR 2021, 2021, pp. 671–675. doi:10.24963/kr.2021/67.

[9] W. Dvořák, M. König, S. Woltran, Treewidth for argumentation frameworks with collective attacks, in: Proceedings of COMMA, 2022. To appear.

[10] S. Gaspers, S. Ordyniak, S. Szeider, Backdoor sets for CSP, in: The Constraint Satisfaction Problem: Complexity and Approximability, volume 7 of *Dagstuhl Follow-Ups*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 137–157. URL: https://doi.org/10.4230/DFU.Vol7.15301.5. doi:10.4230/DFU.Vol7.15301.5.

[11] J. K. Fichte, S. Szeider, Backdoors to tractable answer set programming, Artif. Intell. 220 (2015) 64–103. URL: https://doi.org/10.1016/j.artint.2014.12.001. doi:10.1016/j.artint.2014.12.001.

[12] S. Ordyniak, A. Schidler, S. Szeider, Backdoor dnfs, in: Z. Zhou (Ed.), Proceedings of IJCAI 2021, ijcai.org, 2021, pp. 1403–1409. URL: https://doi.org/10.24963/ijcai.2021/194. doi:10.24963/ijcai.2021/194.

[13] S. Gaspers, N. Misra, S. Ordyniak, S. Szeider, S. Zivný, Backdoors into heterogeneous classes of SAT and CSP, in: C. E. Brodley, P. Stone (Eds.), Proceedings of AAAI, AAAI Press, 2014, pp. 2652–2658. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8177.

[14] W. Dvořák, S. Ordyniak, S. Szeider, Augmenting tractable fragments of abstract argumentation, Artificial Intelligence 186 (2012) 157–173. URL: http://www.sciencedirect.com/science/article/pii/S0004370212000239. doi:10.1016/j.artint.2012.03.002.

[15] G. Flouris, A. Bikakis, A comprehensive study of argumentation frameworks with sets of attacking arguments, Int. J. Approx. Reason. 109 (2019) 55–86. doi:10.1016/j.ijar.2019.03.006.

[16] W. Dvořák, P. E. Dunne, Computational problems in formal argumentation and their complexity, in: Handbook of Formal Argumentation, College Publications, 2018, pp. 631–687.

[17] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem, J. ACM 55 (2008) 21:1–21:19. URL: https://doi.org/10.1145/1411509.1411511. doi:10.1145/1411509.1411511.

[18] W. Dvořák, M. Hecher, M. König, A. Schidler, S. Szeider, S. Woltran, Tractable abstract argumentation via backdoor-treewidth, in: Proceedings of AAAI, 2022, pp. 5608–5615.

[19] A. Abboud, V. V. Williams, Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems, in: Proceedings of FOCS, 2014, pp. 434–443.

[20] R. Impagliazzo, R. Paturi, Complexity of k-SAT, in: Proceedings of CCC, 1999, pp. 237–240.

[21] R. Impagliazzo, R. Paturi, F. Zane, Which Problems Have Strongly Exponential Complexity?, in: Proceedings of FOCS, 1998, pp. 653–662.