Computational Models of Argument
F. Toni et al. (Eds.)

doi:10.3233/FAIA220148

Treewidth for Argumentation Frameworks with Collective Attacks

Wolfgang DVOŘÁK [a], Matthias KÖNIG [a] and Stefan WOLTRAN [a]
[a] *TU Wien, Institute of Logic and Computation*

Abstract. Abstract Argumentation is a key formalism to resolve conflicts in incomplete or inconsistent knowledge bases. Argumentation Frameworks (AFs) and extended versions thereof turned out to be a fruitful approach to reason in a flexible and intuitive setting. The addition of collective attacks, we refer to this class of frameworks as SETAFs, enriches the expressiveness and allows for compacter instantiations from knowledge bases, while maintaining the computational complexity of standard argumentation frameworks. This means, however, that standard reasoning tasks are intractable and worst-case runtimes for known standard algorithms can be exponential. In order to still obtain manageable runtimes, we exploit graph properties of these frameworks. In this paper, we initiate a parameterized complexity analysis of SETAFs in terms of the popular graph parameter treewidth. While treewidth is well studied in the context of AFs with their graph structure, it cannot be directly applied to the (directed) hypergraphs representing SETAFs. We thus introduce two generalizations of treewidth based on different graphs that can be associated with SETAFs, i.e., the primal graph and the incidence graph. We show that while some of these notions allow for parameterized tractability results, reasoning remains intractable for other notions, even if we fix the parameter to a small constant.

Keywords. Abstract Argumentation, Collective Attacks, SETAF, Treewidth

1. Introduction

Argumentation is a key area in Artificial Intelligence. Abstract Argumentation as introduced by Dung [1] serves as a unifying framework to capture argumentation processes in a formal yet intuitive setting. In standard argumentation frameworks (AFs), discussions are formalized as a directed graph where the nodes represent abstract arguments (independent of their internal structure), and the edges represent the attack relation. It turned out that the binary attack relation of AFs occasionally limits the expressiveness of frameworks, in particular if one is not willing to introduce artificial arguments to model technicalities. To avoid this issue, Nielsen and Parsons proposed a relaxation of this restriction: collective attacks [2]. If a set S collectively attacks an argument, said argument is only effectively defeated by S if all arguments in S are accepted by an agent. The resulting class of frameworks is referred to as SETAFs. It was shown that SETAFs are indeed more expressive than AFs [3].

Due to SETAFs being highly expressive yet intuitive, there is now an increased interest in this formalism within the research community. While in the general case the

same (mostly intractable) complexity upper bounds hold [4], tractable graph classes have only recently been investigated [5,6]. We add to this by starting the analysis of computational properties of SETAFs in the context of *parameterized complexity*. A problem is *fixed-parameter tractable* (FPT), if we can find a numerical parameter p describing the instances such that for constant values of p the runtime is polynomial in the instance size (and the degree of the polynomial is independent of p). Implementations of these parameterized algorithms often work well in practice, if instances are not randomly generated but admit an exploitable structure. One prominent such parameter is *treewidth*. A low treewidth indicates a certain "tree-likeness" of a graph, and as problems often become easy on trees, adapted versions of these easy algorithms can often be applied to instancees with low treewidth. In AFs, it has been shown that reasoning is indeed fixed-parameter tractable w.r.t. treewidth [7]. We investigate how this notion of treewidth is applicable to the *directed hypergraph*-structure of SETAFs and show that certain generalizations admit FPT algorithms, while other reasonable attempts do not. In particular, our contributions can be summarized as follows. After recalling the necessary formal background in Section 2, we discuss the challenges of defining treewidth for SETAFs in Section 3 and present two different notions of treewidth: primal-treewidth and incidence-treewidth. In Section 4 we present negative results regarding primal-treewidth, namely that reasoning remains intractable even for small parameter values. Section 5 establishes FPT results for incidence-treewidth via a generic argument utilizing Monadic Second Order logic; this theoretical result gets refined and improved in Section 6 where we discuss a dynamic programming algorithm tailored to SETAFs. Finally, in Section 7 we conclude and give pointers to possibly interesting directions for future research.

2. Background

We start with the definition of an Argumentation Framework with Collective Attacks (SETAF) [2] as a generalization of (standard) Argumentation Frameworks (AFs) [1].

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* $(\{t\},h)$*. We write* $S \mapsto_R a$ *if there is a set* $T \subseteq S$ *with* $(T,a) \in R$*. Moreover, we write* $S' \mapsto_R S$ *if* $S' \mapsto_R a$ *for some* $a \in S$*. We drop subscript* R *in* $\mapsto_R$ *if there is no ambiguity. For* $S \subseteq A$*, we use* S_R^+ *to denote the set* $\{a \mid S \mapsto_R a\}$ *and define the* range *of* S *(w.r.t.* R*), denoted* $S_R^\oplus$*, 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* $S \mapsto_R a$ *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*.*

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* $B \mapsto_R a$*, also* $S \mapsto_R B$*. A set* $T \subseteq A$ *is defended (in* SF*) by* S *if each* $a \in T$ *is defended by* S *(in* SF*).*

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

	grd	*adm*	*com*	*pref*	*stb*
$Cred_\sigma$	P-c	NP-c	NP-c	NP-c	NP-c
$Skept_\sigma$	P-c	trivial	P-c	Π_2^P-c	coNP-c

The semantics we study in this work are the grounded, admissible, complete, preferred, and stable, semantics, which we will abbreviate by *grd*, *adm*, *com*, *pref*, and *stb*, respectively [2,4,8]. Acceptable sets of arguments w.r.t. a semantics are called *extensions*.

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,*
- *$S \in grd(SF)$, if $S = \bigcap_{T \in com(SF)} T$,*
- *$S \in pref(SF)$, if $S \in adm(SF)$ and there is no $T \in adm(SF)$ s.t. $T \supset S$,*
- *$S \in stb(SF)$, if $S \mapsto a$ for all $a \in A \setminus S$,*

The relationship between the semantics has been clarified in [2,4,8] and matches with the relations between the semantics for Dung AFs, i.e. for any SETAF SF:

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

Complexity. 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, 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 [9] carry over to SETAFs, also the same upper bounds hold for SETAFs [4].

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 $O(f(p) \cdot n^k)$, where n is the size of the input, k is an integer constant, 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). For fixed (i.e., constant) parameter values p, FPT-runtime is polynomial (and the degree of the polynomial does not depend on p).

3. Graph Notions and Tree Decompositions of SETAFs

In this section we discuss approaches to apply the notion of *treewidth* [10] to SETAFs.

Definition 5 (Treewidth)**.** *Let $G = (V,E)$ be an undirected graph. A* tree decomposition (TD) *of G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a tree and $\mathcal{X} = (X_n)_{n \in V_\mathcal{T}}$ is a set of bags (a bag is a subset of V) such that*

[1] For a gentle introduction to complexity theory in the context of formal argumentation, see [9].

1. $\bigcup_{n \in V_{\mathscr{T}}} X_n = V$;
2. *for each* $v \in V$, *the subgraph induced by* v *in* $\mathscr{T}$ *is connected; and*
3. *for each* $\{v,w\} \in E$, $\{v,w\} \subseteq X_n$ *for some* $n \in V_{\mathscr{T}}$.

The width *of a TD is* $\max\{|X_n| \mid n \in V_{\mathscr{T}}\} - 1$, *the* treewidth *of* G *is the minimum width of all TDs for* G.

For fixed k it can be decided in linear time whether a graph has treewidth at most k; moreover an according tree decomposition can be computed in linear time [11]. For practical applications there are heuristic approaches available that will return decompositions of reasonable width very efficiently [12]. However, as the underlying structure of SETAFs is a *directed hypergraph*, this notion is not directly applicable in our context. We can use "standard" directed graphs to describe SETAFs, and apply treewidth by simply ignoring the direction of the involved arcs. For SETAFs there is the *primal graph* [5] and the *incidence graph* [6] as such notions, each of which leads to its own treewidth notion for SETAFs. First, we utilize the primal graph to define *primal-treewidth*.

Definition 6 (Primal Graph)**.** *Let* $SF = (A,R)$ *be a SETAF. The* primal graph *of* SF *is defined as* $\texttt{Primal}(SF) = (A,R')$, *where* $R' = \{(t,h) \mid (T,h) \in R, t \in T\}$. *The* primal-treewidth $\texttt{ptw}(SF)$ *is defined as the treewidth of* $\texttt{Primal}(SF)$.

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 [5,6]. In contrast, the *incidence graph* uniquely describes a SETAF, as attacks are explicitly modeled in this notion. Again, we utilize the incidence graph to define *incidence-treewidth*.

Definition 7 (Incidence Graph)**.** *Let* $SF = (A,R)$ *be a SETAF and let* $tails(SF) = \{T \mid (T,h) \in R\}$. *We define the bipartite* incidence graph *of* SF *as* $\texttt{Inc}(SF) = (V,E)$ *with* $V = A \cup tails(SF)$ *and* $E = \{(t,T),(T,h) \mid (T,h) \in R, t \in T\}$. *The* incidence-treewidth $\texttt{itw}(SF)$ *is defined as the treewidth of* $\texttt{Inc}(SF)$.

We want to highlight that (a) both of these notions properly generalize the classical notion of treewidth commonly applied to Dung-style AFs, and (b) these measures coincide on AFs. Formally:

Proposition 8. *The "standard" treewidth of AFs* F *coincides with* $\texttt{ptw}(F)$ *and* $\texttt{itw}(F)$.

Proof. The case of primal-treewidth is immediate. For incidence-treewidth note that we can construct $\texttt{Inc}(F)$ from F by substituting each edge $r = (a,b) \in R$ by a fresh vertex r and two edges (a,r), (r,b). It is well known that this operation preserves treewidth. □

We will first show that reasoning on SETAFs with fixed primal-treewidth remains hard (Section 4). Incidence-treewidth on the other hand admits FPT algorithms—we will initially establish this by characterizing the SETAF semantics via Monadic Second Order logic (MSO) [13,14] (Section 5). We utilize this characterization to obtain the desired upper bounds, as in this context we can apply a meta-theorem due to Courcelle [15,16]. In a nutshell, it states that every graph property that can be characterized in MSO can be checked in polynomial time. However, this generic method typically produces infeasible constants in practice, which is why in Section 6 we will refine these results and provide a prototypical algorithm with feasible constants for stable semantics (cf. [7]).

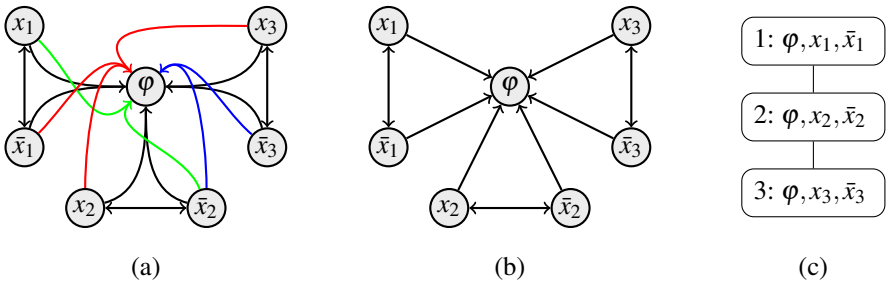


Figure 1. (a) The framework SF_φ from the proof of Theorem 9 for $\varphi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2) \wedge (x_2 \vee x_3)$, (b) $\texttt{Primal}(SF_\varphi)$, and (c) a tree-decomposition of $\texttt{Primal}(SF_\varphi)$ of width 2.

4. Decomposing the Primal Graph

We start with an investigation of the treewidth of the primal graph. It has been shown that various restrictions on the primal graph can lead to computational ease [5,6]. However, we will see that reasoning remains hard for SETAFs with constant primal-treewidth (in contrast to the AF-case, where we observe FPT results). We establish this via reductions from (QBF-)SAT, illustrated in Figures 1 and 2. Intuitively, the attacks between the dual literals x and $\bar{x}$ represent the choice between assigning x to *true* or *false*. The collective attack $(\{x, \bar{x}\}, \varphi)$ ensures that we take at least one of x and $\bar{x}$ into any extension in order to defend φ, making sure we only construct proper truth assignments. Finally, the remaining attacks towards φ correspond to the clauses and make sure that we cannot defend φ if for any clause we set all duals of its literals *true*—as this means the clause is not satisfied.

Theorem 9. *The problems* $Cred_\sigma$ *for* $\sigma \in \{adm, com, stb, pref\}$ *are* NP*-complete, and* $Skept_{stb}$ *is* coNP*-complete for SETAFs SF with* $\texttt{ptw}(SF) \geq 2$.

Proof. The membership coincides with the general case. For the respective hardness results, consider the following reduction from SAT (see Figure 1). Let X be the set of atoms and C be the set of clauses of a boolean formula φ (given in CNF). We denote a clause $c \in C$ as the set of literals in the clause, e.g. the clause $x_1 \vee \bar{x}_2 \vee \bar{x}_3$ correspond to the set of literals (arguments, resp.) $\{x_1, \bar{x}_2, \bar{x}_3\}$. By x^d we denote the dual of a literal (e.g. $x^d = \bar{x}$ and $\bar{x}^d = x$). Let $SF_\varphi = (A, R)$, where $A = \{x, \bar{x} \mid x \in X\} \cup \{\varphi\}$, and

$$R = \{(\{x^d \mid x \in c\}, \varphi) \mid c \in C\} \cup \{(\{x, \bar{x}\}, \varphi), (x, \bar{x}), (\bar{x}, x) \mid x \in X\}$$

Now it holds that φ is in at least one σ extension for $\sigma \in \{adm, com, stb, pref\}$ if and only if φ is satisfiable.

($\Rightarrow$) An admissible set E containing φ contains exactly one of each pair $x, \bar{x}$, as otherwise φ would not be defended against the attack $(\{x, \bar{x}\}, \varphi)$. Moreover, as $\bar{c} \not\subseteq E$ for each attack $(\bar{c}, \varphi)$—$\bar{c}$ consists of the *duals* of the literals in c—this means at least one argument corresponding to a literal of each clause $c \in C$ is in E. Hence, E corresponds to a satisfying assignment of φ.

($\Leftarrow$) Every satisfying assignment of φ corresponds to a stable extension of SF_φ: let I be the interpretation satisfying φ, the corresponding set $E = \{\varphi\} \cup \{x \mid I(x) = true\} \cup \{\bar{x} \mid I(x) = false\}$ is then stable (admissible, complete, preferred): As I satisfies φ, for each attack corresponding to a clause not all tail-arguments are in E, and hence φ is defended.

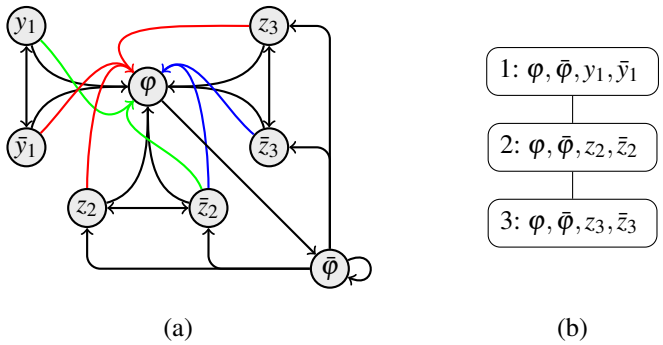


Figure 2. (a) SF_Φ from the proof of Theorem 10 for $\Phi = \forall\{y_1\}\exists\{z_2, z_3\}(y_1 \vee \bar{z}_2 \vee \bar{z}_3) \wedge (\bar{y}_1 \vee z_2) \wedge (z_2 \vee z_3)$, and (b) a tree-decomposition of $\texttt{Primal}(SF_\Phi)$ of width 3.

For the coNP completeness result, we add an argument $\bar{\varphi}$ and an attack $(\varphi, \bar{\varphi})$. If φ is unsatisfiable, φ will be attacked and $\bar{\varphi}$ will be contained in every stable extension. Finally note that stable extensions are admissible, complete, and preferred. The constant primal-treewidth is immediate, as illustrated in the example of Figure 1(c). □

Also for preferred semantics reasoning remains intractable for SETAFs with fixed primal-treewidth. For this result, we extend the construction from Theorem 9 by an additional argument $\bar{\varphi}$ that attacks the existentially quantified variables (see Figure 2).

Theorem 10. *$Skept_{pref}$ is Π_2^P-complete for SETAFs SF with $\texttt{ptw}(SF) \geq 3$.*

Proof. We show this by a reduction from the Π_2^P-complete $QBF_\forall^2$ problem. Let $\Phi = \forall Y \exists Z \varphi$ be a $QBF_\forall^2$-formula with φ in CNF. We construct the SETAF SF_Φ by extending F_φ from Theorem 9 in the following way (for an example see Figure 2): First, we set $X = Y \cup Z$ and add all arguments and attacks according to the construction of F_φ. Moreover we add an argument $\bar{\varphi}$ and attacks $(\bar{\varphi}, \bar{\varphi}), (\varphi, \bar{\varphi})$. The last step is to add attacks $(\bar{\varphi}, z), (\bar{\varphi}, \bar{z})$ for each $z \in Z$. Now φ is in every preferred extension of SF_Φ if and only if Φ is valid. We start with some general observations: $\bar{\varphi}$ cannot be in any admissible set, and φ can only be in an admissible set S if for each $x \in Y \cup Z$ exactly one of x and $\bar{x}$ is in S. Moreover, in order to have $S \cap (Z \cup \bar{Z}) \neq \emptyset$, the argument $\bar{\varphi}$ has to be attacked by S, and consequently $\varphi \in S$. This means for every admissible set S that $S \cap (Y \cup \bar{Y} \cup Z \cup \bar{Z})$ corresponds to a satisfying assignment of the formula φ. In summary, every assignment on the variables Y corresponds to an admissible set, and every other admissible set in SF_Φ contains φ and represents a satisfying assignment for φ.

($\Rightarrow$) Assume φ is in every preferred extension. Since every set $S \subseteq 2^Y$ is admissible and in order to accept φ for each $x \in Y \cup Z$ either x or $\bar{x}$ have to be accepted, we know that for every assignment of Y variables there is an assignment satisfying φ.

($\Leftarrow$) Now assume Φ is valid, i.e. for each assignment I_Y on the variables Y, there is an assignment I_Z on Z such that $I_Y \cup I_Z$ satisfies φ. From this and our observations above it follows that φ is in every preferred extensions.

It is easy to see that the primal-treewidth of F_Φ is bounded by 3 (see Figure 2(b)). □

Hence, under standard complexity-theoretical assumptions, these problems do not become tractable when parameterized by the primal-treewidth.

5. Parameterized Tractability via Incidence-Treewidth

In this section, we establish tractability for reasoning in SETAFs with constant incidence treewidth by utilizing a meta-theorem due to Courcelle [15,16]. In particular, we use the tools of Monadic Second Order logic (MSO) to characterize the semantics of SETAFs (similarly, this has been done for AFs [13,14]). MSO generalizes first-order logic in the sense that it is also allowed to quantify over sets. Domain elements in our settings are vertices of an (incidence)-graph, i.e., arguments or attacks. Hence, MSO in our context consists of *variables* corresponding to domain elements (indicated by lower case letters), and *set-variables* corresponding to sets of domain elements (indicated by uppercase letters). Moreover, we use the standard logical connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, as well as quantifiers $\exists, \forall$ for both types of variables. We use the unary predicates $A(\cdot)$ and $R(\cdot)$ to indicate an element being an argument or an attack of our SETAF, respectively. Moreover, we use the binary predicate $E(x,y)$ to indicate an edge in the incidence graph between incidence-vertices x and y. Alternatively, we write $a \in A$, $r \in R$, $(x,y) \in E$ for $A(a), R(r), E(x,y)$, respectively. Based on these basic definitions, we define notational shortcuts to conveniently characterize SETAF-properties. Let $SF{=}(A,R)$ be a SETAF and $\mathtt{Inc}(SF){=}(V,E)$ its incidence graph. We define the following notion for $T \subseteq V$ and $h \in V$: let $(T,h) \in R$ be short-hand notation for $\exists r\,(r \in R \wedge (r,h) \in E \wedge \forall t(t \in T \leftrightarrow (t,r) \in E))$. This notion consists of four parts: (1) vertex r corresponds to an attack, (2) h is the head of the attack r, (3) the arguments in T constitute the tail of r. We utilize this to avoid dealing with the attack-vertices of the incidence graph in our semantics characterizations. We borrow the following "building blocks" from [14] (slightly adapted for our setting).

$$
\begin{array}{ll}
X \subseteq Y = \forall x\,(x \in X \rightarrow x \in Y) & X \not\subset Y = \neg(X \subset Y) \\
X \subset Y = X \subseteq Y \wedge \neg(Y \subseteq X) & x \notin X = \neg(x \in X) \\
X \not\subseteq Y = \neg(X \subseteq Y) & x \in X_R^{\oplus} = x \in X \vee \exists Y(Y \subseteq X \wedge (Y,x) \in R)
\end{array}
$$

We can express (subset-)maximality: $\max_{A,P(.),\subseteq}(X) = P(X) \wedge \neg\exists Y(Y \subseteq A \wedge P(Y) \wedge X \subset Y)$, and analogously (subset-)minimality: $\min_{A,P(.),\subseteq}(X) = \max_{A,P(.),\supseteq}(X)$ [14] for any expressible property $P(\cdot)$. Having these tools at hand, we can characterize the SETAF semantics in an intuitive way. It is easy to verify that these exactly correspond to the respective notions from Definition 4. Utilizing these building blocks, we can encode the semantics $cf, adm, com, grd, stb, pref$ exactly as in AFs [13,14].

Definition 11. *Let $SF = (A,R)$ be a SETAF and let $\mathtt{Inc}(SF) = (V,E)$ be its incidence graph. For a set $X \subseteq V$ where $\forall x \in X(x \in A)$:*

$$
\begin{aligned}
cf(X) &= \forall T,h\,((T,h) \in R \rightarrow (T \not\subseteq X \vee h \notin X)) \\
adm(X) &= cf(X) \wedge \forall T,h(((T,h) \in R \wedge h \in X) \rightarrow \exists S,t(S \subseteq X \wedge t \in T \wedge (S,t) \in R)) \\
com(X) &= adm(X) \wedge \forall x((x \in A \wedge x \notin X) \rightarrow \\
&\qquad \exists S((S,x) \in R \wedge \neg\exists T(T \subseteq X \wedge (X,s) \in R \wedge s \in S))) \\
grd(X) &= \min_{A,com(\cdot),\subseteq}(X) \\
stb(X) &= cf(X) \wedge \forall x(x \in A \rightarrow x \in X_R^{\oplus}) \\
pref(X) &= \max_{A,adm(\cdot),\subseteq}(X)
\end{aligned}
$$

We can immediately apply Courcelle's theorem [15,16] to obtain the desired result.

Theorem 12. *Let SF be a SETAF. For the semantics under our consideration, reasoning is fixed-parameter tractable w.r.t.* $\mathtt{itw}(SF)$.

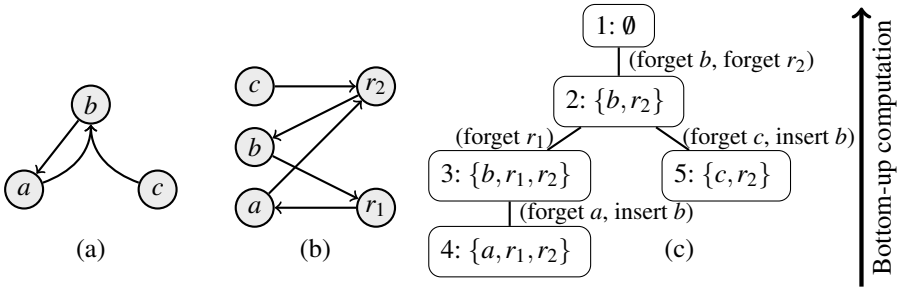


Figure 3. Running example for Section 6: (a) SETAF SF; (b) $\texttt{Inc}(SF)$; (c) tree decomposition of $\texttt{Inc}(SF)$. The edge labels indicate how (c) can be transformed into a nice tree decomposition.

6. Dynamic Programming on SETAFs

In the following, we specify a dynamic programming algorithm utilizing incidence-treewidth to reason in stable semantics. Ultimately, we will show that this algorithm allows us to reason efficiently in SETAFs with fixed incidence-treewidth.

Node Types. To illustrate the idea of this algorithm, we restrict the tree-decompositions of the incidence graph to *nice tree-decompositions*: a tree-decomposition $(\mathscr{T}, \mathscr{X})$ is called *nice* if $\mathscr{T} = (V_\mathscr{T}, E_\mathscr{T})$ is a rooted tree with an empty bag in the root node, and if each node $t \in \mathscr{T}$ (shorthand notion for $n \in V_\mathscr{T}$) is of one of the following types:

1. *Leaf*: n has no children in $\mathscr{T}$,
2. *Forget*: n has one child n', and $X_n = X_{n'} \setminus \{v\}$ for some $v \in X_{n'}$,
3. *Insert*: n has one child n', and $X_n = X_{n'} \cup \{v\}$ for some $v \notin X_{n'}$,
4. *Join*: n has two children n', n'', with $X_n = X_{n'} = X_{n''}$.

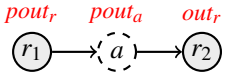
Any tree-decomposition can be transformed into a nice tree decomposition with the same width in linear time [17]. Let $SF = (A, R)$ be a SETAF and $\texttt{Inc}(SF) = (V, E)$. For sets $U \subseteq V$, by U^A, U^R we identify the sets $(U \cap A)$, $(U \cap R)$, respectively. By $X_{\geq n}$ we denote the union of all bags X_m where $m \in V_\mathscr{T}$ appears in the subtree of $\mathscr{T}$ rooted in n.

Colorings. We use *colors* to keep track of the arguments and attacks that appear in the bag X_n of node $n \in V_\mathscr{T}$. These colorings characterize extension candidates that are consistent with the framework rooted in the node in question. For an argument a in relation to an extension E, we use the color in_a to indicate $a \in E$. The color out_a indicates there is an attack $r = (T, a)$ with $a \notin T, T \subseteq E$, and $r \in X^R_{\geq n}$, i.e., a is attacked by E and a "responsible" attack appears in the subtree of $\mathscr{T}$ rooted in n. Finally, the color $pout_a$ (*provisionally* out) indicates there is an attack $r = (T, a)$ with $a \notin T, T \subseteq E$, and $r \notin X^R_{\geq n}$, i.e., a is attacked by E but the "responsible" attack appears "above" the node n in $\mathscr{T}$. Similarly, for attacks (T, h) we use the color in_r to indicate $T \subseteq E$. The color out_r means that there is an argument $a \in T$ (i.e., in the tail) that is attacked by E, and the "responsible" attack appears in the subtree of $\mathscr{T}$ rooted in n. Finally, $pout_r$ means that an argument $a \in T$ is attacked by E, but the "responsible" attack appears "above" the node n in $\mathscr{T}$. Formally, a *coloring* for a node $n \in V_\mathscr{T}$ is a function $C : X_n \to \{in_a, out_a, pout_a, in_r, out_r, pout_r\}$. By $[C]$ we denote the set $\{a \mid C(a) = in_a\}$. Colorings in a node $t \in \mathscr{T}$ characterize extension candidates—partial evaluations of the framework rooted in t. Colorings are generated in the leaves, and unsuitable extension candidates are successively eliminated when traversing the tree in a bottom-up manner. For an example see Figure 3.

(a)

r_1	a	r_2
in_r	out_a	out_r
$pout_r$	in_a	in_r
$pout_r$	in_a	$pout_r$
$pout_r$	$pout_a$	out_r

(b)



(c)

r_1	r_2
in_r	out_r
$pout_r$	in_r
$pout_r$	$pout_r$

Figure 4. Example for valid colorings for (a) the leaf node 4 from Figure 3 and (c) the preceding forget node for argument a. Subfigure (b) illustrates the subgraph of the incidence graph that corresponds to the leaf node together with the coloring that is discarded by the forget node.

Leaf Nodes. Intuitively, in leaf nodes we guess one of two possibilities: *in* or *out*/*pout* for each argument and each attack, and keep every "consistent" coloring. Whether an argument/attack is colored *out* or *pout* depends only on whether the attack in this coloring is already present in the current leaf node. Formally, a *valid coloring* for a leaf n is each coloring that satisfies the conditions in the box below. The valid colorings for leaf node 4 of our running example (Figure 3) are depicted in Figure 4(a).

$$\text{For each argument } a \in X_n^A\text{:}$$
$$C(a) = in_a \Rightarrow \forall r = (T,a) \in X_n^R\colon C(r) \in \{pout_r, out_r\}$$
$$C(a) = out_a \Leftrightarrow \exists r = (T,a) \in X_n^R\colon C(r) = in_r$$

$$\text{For every attack } r = (T,h) \in X_n^R\text{:}$$
$$C(r) = in_r \Rightarrow C(h) \neq in_a \wedge \forall t \in T \cap X_n^A\colon C(t) = in_a$$
$$C(r) = out_r \Leftrightarrow \exists t \in T \cap X_n^A\colon C(t) \in \{pout_a, out_a\}$$

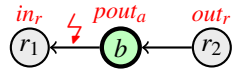
Forget Nodes. We examine forget-argument nodes and forget-attack nodes separately. Let n be a *forget-argument node* with child n' such that $X_n^A = X_{n'}^A \setminus \{a\}$. We have to discard all colorings C where $C(a) = pout_a$, as in these colorings a is supposed to be attacked by $[C]$. As we forget a in the current node and by the definition of a tree-decomposition, this cannot happen: consider again the running example from Figure 3. a is forgotten between bag 4 and 3; i.e., in the "upper" part of the tree decomposition, no attacks towards a can be added. Hence, there cannot be an attack colored in_r towards a that confirms a being attacked, and the provisional color $pout_a$ cannot be updated to out_a. Formally, if C is a valid coloring for n' and $C(a) \neq pout_a$, then $C - a$ is a valid coloring for n, where $(C-a)(b) = C(b)$ for each $b \in X_n$. We handle *forget-attack nodes* in the same way: if n is a forget node with child n' such that $X_n^R = X_{n'}^R \setminus \{r\}$, and if $C(r) \neq pout_r$, then $(C-r)$ is a valid coloring for n, where $(C-r)(b) = C(b)$ for each $b \in X_n$.

Insert Nodes. We distinguish the two cases where we insert an argument and insert an attack. Whenever we insert an argument a, we have to consider up to two different scenarios: $(C+a)$: the added argument is attacked by the extension. In this case the added argument is colored $pout_a$ or out_a, depending on whether the "responsible" attack is already in the current bag. In case a is in the tail of an attack, we can color this attack out_r. $(C\dot{+}a)$: the added argument is in the extension. In both cases we have to check whether the result will be consistent with the existing colors, i.e., for $(C+a)$ the added argument must not be in the tail of an attack that is colored in_r, and for $(C\dot{+}a)$ there must not be an attack colored in_r towards the added argument. Assume we would color the inserted argument b as out_a/$pout_a$ while b is in the tail of attack r, which we already colored in_r in a previous step. Of course, this is not consistent with our intended meaning of the attack color in_r (see (Figure 5(b)). On the other hand, assume we color b as in_a while it is attacked by r which we already colored in_r in a previous step. This would

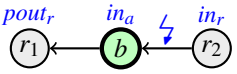
(a)

r_1	r_2
in_r	out_r
$pout_r$	in_r
$pout_r$	$pout_r$

(b)



(c)



(d)

r_1	b	r_2
in_r	in_a	out_r
out_r	out_a	in_r
out_r	$pout_a$	$pout_r$
$pout_r$	in_a	$pout_r$

Figure 5. "Insert b" node between node 4 and 3 (in the running example from Figure 3 after "Forget a" from Figure 4). Subfigures (b) and (c) show inconsistent colorings, (d) shows the resulting valid colorings.

introduce a conflict in the constructed extension (see (Figure 5(c)). The operations $C+a$ and $C\dot{+}a$ are defined the box below. Formally: Let n be an *insert-argument node* with child n' s.t. $X_n^A = X_{n'}^A \cup \{a\}$. If C is a valid coloring for n',

- if $\nexists r=(T,h) \in X_n^R$: $(C(r) = in_r \wedge a \in T)$, then $C+a$ is a valid coloring for n;
- if $\nexists r=(T,a) \in X_n^R$: $(C(r) = in_r)$, then $C\dot{+}a$ is a valid coloring for n.

$$(C+a)(b) = \begin{cases} out_a & \text{if } b = a \wedge \exists r = (T,a) \in X_n^R\colon (C(r) = in_r \wedge a \notin T) \\ pout_a & \text{if } b = a \wedge \nexists r = (T,a) \in X_n^R\colon (C(r) = in_r \wedge a \notin T) \\ out_r & \text{if } b = (T,h) \wedge a \in T \wedge C(b) = pout_r \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\dot{+}a)(b) = \begin{cases} in_a & \text{if } b = a \\ C(b) & \text{otherwise} \end{cases}$$

For *insert-attack nodes*, we also have to consider two cases for an attack $r=(T,h)$: $(C+r)$: the extension attacks T, either in the current bag (in which case we color the attack out_r), or possibly in the "upper" parts of $\mathscr{T}$, then we color the attack $pout_r$. $(C\dot{+}r)$: this case indicates $T \subseteq E$ for the extension E. In this case the head of the attack can be set to out_a. Again, we can only apply this coloring if it is consistent with the previous colors. We will use the operations $C+r$ and $C\dot{+}r$ as defined below. Let n be an *insert-attack node* with child n' such that $X_n^R = X_{n'}^R \cup \{r=(T,h)\}$. If C is a valid coloring for n',

- then $C+r$ is a valid coloring for n;
- if $(h \notin X_n^A \vee C(h) \neq in_a) \wedge \forall t \in T \cap X_t^A\colon C(t) = in_a$, then $C\dot{+}r$ is a valid coloring for n.

$$(C+r)(b) = \begin{cases} out_r & \text{if } b = r \wedge \exists t \in T \cap X_n^A\colon C(t) \in \{pout_a, out_a\} \\ pout_r & \text{if } b = r \wedge \nexists t \in T \cap X_n^A\colon C(t) \in \{pout_a, out_a\} \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\dot{+}r)(b) = \begin{cases} in_r & \text{if } b = r \\ out_a & \text{if } r = (T,h) \wedge b = h \\ C(b) & \text{otherwise} \end{cases}$$

Join Nodes. In these nodes we combine the colorings of immediate child nodes. Let n be a join node with children n', n''. If C is a valid coloring for n' and D is a valid coloring for n'' with $[C] = [D]$ and $\{r \mid C(r) = in_r\} = \{r \mid D(r) = in_r\}$, then $C \bowtie D$ is a valid coloring for n (see the box below).

$$(C \bowtie D)(b) = \begin{cases} in_a & \text{if } C(b) = D(b) = in_a \\ out_a & \text{if } C(b) = out_a \vee D(b) = out_a \\ pout_a & \text{otherwise (if } b \in X^A) \\ in_r & \text{if } C(b) = D(b) = in_r \\ out_r & \text{if } C(b) = out_r \vee D(b) = out_r \\ pout_r & \text{otherwise (if } b \in X^R) \end{cases}$$

Theorem 13. *With the presented algorithm, $Cred_{stb}$, $Skept_{stb}$ as well as counting the number of stable extensions can be done in time $O(5^k \cdot k \cdot (|A| + |R|))$. Moreover, we can enumerate all stable extensions with linear delay.*

Proof. We can assume the number of nodes to be bounded by $O(|A| + |R|)$. For each node, the number of (valid) colorings (i.e., rows in our tables of colorings) is bounded by 3^k and that we can find and access rows in linear time w.r.t. k. In leaf nodes, we can check the colorings in time $O(k^2)$ for each of the $O(3^k)$ possible colorings, resulting in $O(3^k \cdot k^2)$. In forget nodes, we can check the conditions and compute eventually resulting colorings in time $O(k)$ for each of the $O(3^k)$ colorings of the child node, resulting in $O(3^k \cdot k)$. In insert nodes, we can check the conditions and compute eventually resulting colorings in time $O(k^2)$ for each of the $O(3^k)$ colorings of the child node, resulting in $O(3^k \cdot k^2)$. Finally, for join nodes we have to consider $3^k \cdot 3^k = 9^k$ pairs. However, we only need to consider 5^k pairs if we assume the data structure to be properly sorted, e.g. lexicographically by treating the colors in_a/in_r as 0 and $pout_a/out_a/pout_r/out_r$ as 1. As each table has $O(3^k)$ rows, sorting is in $O(3^k \cdot k)$. Let C be a coloring such that $m \leq k$ arguments/attacks are colored as in_a/in_r. There exist at most 2^{k-m} distinct colorings C' with $\forall x : (C(x) \in \{in_a, in_r\} \Leftrightarrow C'(x) \in \{in_a, in_r\})$. There are $\binom{k}{m}$ possibilities resulting from the choice of m, resulting in $\sum_{m=0}^{k} \binom{k}{m} \cdot 2^{k-m} \cdot 2^{k-m} = 5^k$ join pairs. We can then compute $C \bowtie D$ in $O(k)$, resulting in $O(5^k \cdot k)$ for join nodes, dominating the runtime of the other node types. The resulting runtime for the algorithm is $O(5^k \cdot k \cdot (|A| + |R|))$.

We can decide $Cred_{stb}$/$Skept_{stb}$ for $a \in A$ by flagging colorings that contain/do not contain a. In each node we update the flag accordingly; the flag in the root node indicates credulous/skeptical acceptance. We can keep the count of extensions w.r.t. each coloring, and to enumerate the extensions once the dynamic programming algorithm is done we can traverse the tree top-down and output the extensions with linear delay (cf. [7]). □

Other Semantics. The core concepts to characterize stable extensions carry over to other admissibility-based semantics, where also undecidedness can occur. This can be handled in a similar manner as the $pout_r/out_r$ colors, where one indicates "confirmed undecidedness" and another color indicates "provisional undecidedness". The latter color can be "updated" to the former if a suitable witness is present (either in an insert- or join node). Again, colorings containing provisional colors have to be removed in forget nodes.

7. Discussion

In this paper, we investigated the treewidth parameter for reasoning tasks in SETAFs. We showed that reasoning with constant primal-treewidth remains hard (contrasting the results for the special case of AFs), while constant incidence-treewidth allows us to reason and count in polynomial time. Finally, we improved these generically obtained results by providing a dynamic programming algorithm tailored for SETAFs, highlighting interesting differences to the AF-case that arise from the generalization step. The underlying structure of SETAFs is a directed hypergraph. While there are measures available for general hypergraphs, the directed case is not as well explored—this work contributes to this, as we provide an alternative treewidth measure in this context. Moreover, while there are several systems available to compute the treewidth of undirected simple graphs efficiently—be it exactly or heuristically—the situation for implementations of hyper-

treewidth is less advanced. Finally, reasoning in frameworks with fixed *directed* graph parameters (e.g., cycle rank, directed path-width, etc.) already turned out to be intractable for AFs [7]; which carries over to SETAFs. Hence, we decided to focus on the treewidth-based measures, so that we can implement the presented algorithms in the future.

The results of this paper may serve as a starting point for further parameterized analysis of computational properties of SETAFs. Considering SETAFs in recent additions to the treewidth literature in the context of argumentation constitutes interesting topics for future research, see e.g. [18]. For example, recently treewidth has been investigated in conjunction with *backdoors* in [19], effectively decreasing the relevant parameter value.

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] Dung PM. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artif Intell. 1995;77(2):321-58.
[2] Nielsen SH, Parsons S. A Generalization of Dung's Abstract Framework for Argumentation: Arguing with Sets of Attacking Arguments. In: Proceedings of ArgMAS 2006. Springer; 2006. p. 54-73.
[3] Dvořák W, Fandinno J, Woltran S. On the expressive power of collective attacks. Argument Comput. 2019;10(2):191-230.
[4] Dvořák W, Greßler A, Woltran S. Evaluating SETAFs via Answer-Set Programming. In: Proceedings of SAFA 2018. vol. 2171 of CEUR Workshop Proceedings. CEUR-WS.org; 2018. p. 10-21.
[5] Dvořák W, König M, Woltran S. Graph-Classes of Argumentation Frameworks with Collective Attacks. In: Proceedings of JELIA 2021. vol. 12678 of LNCS. Springer; 2021. p. 3-17.
[6] Dvořák W, König M, Woltran S. On the Complexity of Preferred Semantics in Argumentation Frameworks with Bounded Cycle Length. In: Proceedings of KR 2021; 2021. p. 671-5.
[7] Dvořák W, Pichler R, Woltran S. Towards fixed-parameter tractable algorithms for abstract argumentation. Artif Intell. 2012;186:1-37.
[8] Flouris G, Bikakis A. A comprehensive study of argumentation frameworks with sets of attacking arguments. Int J Approx Reason. 2019;109:55-86.
[9] Dvořák W, Dunne PE. Computational Problems in Formal Argumentation and their Complexity. In: Handbook of Formal Argumentation. College Publications; 2018. p. 631-87.
[10] Robertson N, Seymour PD. Graph minors. II. Algorithmic aspects of tree-width. J Algorithms. 1986;7(3):309-22.
[11] Bodlaender HL. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. SIAM J Comput. 1996;25(6):1305-17.
[12] Abseher M, Musliu N, Woltran S. htd - A Free, Open-Source Framework for (Customized) Tree Decompositions and Beyond. In: Proceedings of CPAIOR 2017. Springer; 2017. p. 376-86.
[13] Dunne PE. Computational properties of argument systems satisfying graph-theoretic constraints. Artif Intell. 2007;171(10-15):701-29.
[14] Dvořák W, Szeider S, Woltran S. Abstract Argumentation via Monadic Second Order Logic. In: Proceedings of SUM 2012. vol. 7520 of LNCS. Springer; 2012. p. 85-98.
[15] Courcelle B. Recognizability and second-order definability for sets of finite graphs. Université de Bordeaux; 1987. I-8634.
[16] Courcelle B. Graph rewriting: an algebraic and logic approach. In: Handbook of theoretical computer science, Vol. B. Amsterdam: Elsevier; 1990. p. 193-242.
[17] Kloks T. Treewidth, Computations and Approximations. vol. 842 of LNCS. Springer; 1994.
[18] Fichte JK, Hecher M, Mahmood Y, Meier A. Decomposition-Guided Reductions for Argumentation and Treewidth. In: Zhou Z, editor. Proceedings of IJCAI 2021; 2021. p. 1880-6.
[19] Dvořák W, Hecher M, König M, Schidler A, Szeider S, Woltran S. Tractable Abstract Argumentation via Backdoor-Treewidth. In: Proceedings of AAAI 2022; 2022. p. 5608-15.