Counting Complexity for Reasoning in Abstract Argumentation*

Johannes K. Fichte
TU Dresden
01062 Dresden, Germany
johannes.fichte@tu-dresden.de

Markus Hecher
TU Wien
Institute of Logic and Computation
Favoritenstraße 9-11 / E192
1040 Vienna, Austria
hecher@dbai.tuwien.ac.at

Arne Meier
Leibniz Universität Hannover
Institut für Theoretische Informatik
Appelstraße 4
30167 Hannover, Germany
meier@thi.uni-hannover.de

Abstract

In this paper, we consider counting and projected model counting of extensions in abstract argumentation for various semantics. When asking for projected counts we are interested in counting the number of extensions of a given argumentation framework while multiple extensions that are identical when restricted to the projected arguments count as only one projected extension. We establish classical complexity results and parameterized complexity results when the problems are parameterized by treewidth of the undirected argumentation graph. To obtain upper bounds for counting projected extensions, we introduce novel algorithms that exploit small treewidth of the undirected argumentation graph of the input instance by dynamic programming (DP). Our algorithms run in time double or triple exponential in the treewidth depending on the considered semantics. Finally, we take the exponential time hypothesis (ETH) into account and establish lower bounds of bounded treewidth algorithms for counting extensions and projected extension.

Introduction

Abstract argumentation (Dung 1995; Rahwan 2007) is a central framework for modeling and the evaluation of arguments and its reasoning with applications to various areas in artificial intelligence (AI) (Amgoud and Prade 2009; Rago, Cocarascu, and Toni 2018). The semantics of argumentation is described in terms of arguments that are acceptable with respect to an abstract framework, such as stable or admissible. Such arguments are then called extensions of a framework. In argumentation, one is particularly interested in the credulous or skeptical reasoning problem, which asks, given an argumentation framework and an argument, whether the argument is contained in some or all extension(s) of the framework, respectively. A very interesting, but yet entirely unstudied question in abstract argumentation is the computation and the computational complexity of counting, which asks for outputting the number of extensions with respect to a certain semantics. By counting extensions, we can answer questions such as how many extensions are available containing certain arguments. An even more interesting question is how many extensions containing certain arguments exist when restricted to a certain subset of the arguments which asks for outputting the number of projected extensions.

Interestingly, the computational complexity of the decision problem is already quite hard. More precisely, the problem of credulous acceptance, which asks whether a given argument is contained in at least one extension, is NP-complete for the stable semantics and even Σ_2^p-complete for the semi-stable semantics (Dunne and Bench-Capon 2002; Dvořák and Woltran 2010; Dvořák 2012). The high worst-case complexity is often a major issue to establish algorithms for frameworks of abstract argumentation. A classical way in parameterized complexity and algorithmics is to identify structural properties of an instance and establish efficient algorithms under certain structural restrictions (Cygan et al. 2015). Usually, we aim for algorithms that run in time polynomial in the input size and exponential in a measure of the structure, so-called *fixed-parameter tractable* algorithms. Such runtime results require more fine-grained runtime analyses and more evolved reductions than in classical complexity theory where one considers only the size of the input. Here, we take a graph-theoretical measure of the undirected graph of the given argumentation framework into account. As measure we take treewidth, which is arguably the most prominent graph invariant in combinatorics of graph theory and renders various graph problems easier if the input graph is of bounded treewidth.

Our results are as follows:

- We establish the classical complexity of counting extensions and counting projected extensions for various semantics in abstract argumentation.
- We present an algorithm that solves counting projected extensions by exploiting treewidth in runtime double exponential in the treewidth or triple exponential in the treewidth depending on the considered semantics.
- Assuming the exponential time hypothesis (ETH), which states that there is some real $s > 0$ such that we cannot decide satisfiability of a given 3-CNF formula φ in time $2^{s\cdot|\varphi|} \cdot \|\varphi\|^{\mathcal{O}(1)}$, we show that one *cannot* count projected extensions double exponentially in the treewidth.

*Funded by Austrian Science Fund FWF grants I2854, Y698, and P30168-N31, as well as the German Research Fund DFG grants HO 1294/11-1 and ME 4279/1-2. The second author is also affiliated with the University of Potsdam, Germany.

Related work. Baroni, Dunne, and Giacomin (2010) considered general extension counting and show #P-completeness and identify tractable cases. We generalize these results to the reasoning problems. Lampis, Mengel, and Mitsou (2018) considered bounded treewidth algorithms and established lower bounds for the runtime of an algorithm that solves reasoning in abstract argumentation under the admissible and preferred semantics. These results do not trivially extend to counting and are based on reductions to QBF. They yield asymptotically tight bounds, but still involve a constant factor. Unfortunately, already a small increase even by one can amount to one order of magnitude in inference time with *dynamic programming (DP)* algorithms for QBF. As a result, a factor of just two can already render it impractical. Fichte et al. (2018) gave DP algorithms for projected #SAT and established that it cannot be solved in runtime double exponential in the treewidth under ETH using results by Lampis and Mitsou (2017), who established lower bounds for the problem $\exists\forall$-SAT. Dvořák, Pichler, and Woltran (2012) introduced DP algorithms that exploit treewidth to solve decision problems of various semantics in abstract argumentation. We employ these results and lift them to projected counting. Further, DP algorithms for projected counting in answer set programming (ASP) were recently presented (Fichte and Hecher 2018a).

Formal Background

We use graphs and digraphs as usually defined (Bondy and Murty 2008) and follow standard terminology in computational complexity (Papadimitriou 1994) and parameterized complexity (Cygan et al. 2015). Let Σ and Σ' be some finite alphabets and $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. For $(I, k) \in L$, we call $I \in \Sigma^*$ an *instance* and k the parameter. For a set X, let 2^X consist of all subsets of X. Later we use the generalized combinatorial inclusion-exclusion principle, which allows to compute the number of elements in the union over all subsets (Graham, Grötschel, and Lovász 1995).

Counting Complexity. We follow standard terminology in this area (Toda and Watanabe 1992; Hemaspaandra and Vollmer 1995; Durand, Hermann, and Kolaitis 2005). In particular, we will make use of complexity classes preceded with the sharp-dot operator '$\#\cdot$'. (Note the difference to Valiant's classes (Valiant 1979).) A *witness* function is a function $w\colon \Sigma^* \to \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are alphabets, mapping to a finite subset of Γ^*. Such functions associate with the counting problem "given $x \in \Sigma^*$, find $|w(x)|$". If $\mathcal{C}$ is a decision complexity class then $\# \cdot \mathcal{C}$ is the class of all counting problems whose witness function w satisfies (1.) $\exists$ polynomial p such that for all $y \in w(x)$, we have that $|y| \leqslant p(|x|)$, and (2.) the decision problem "given x and y, is $y \in w(x)$?" is in $\mathcal{C}$. A *parsimonious* reduction between two counting problems $\#A, \#B$ preserves the cardinality between the corresponding witness sets and is computable in polynomial time. A *subtractive* reduction between two counting problems $\#A$ and $\#B$ is composed of two functions f, g between the instances of A and B such that $B(f(x)) \subseteq B(g(x))$ and $|A(x)| = |B(g(x))| - |B(f(x))|$, where A and B are respective witness functions.

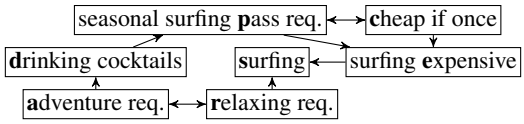

Figure 1: Argumentation framework F: surfing vs. cocktails.

Abstract Argumentation. We consider the Argumentation Framework by Dung (1995). An *argumentation framework (AF)*, or framework for short, is a directed graph $F = (A, R)$ where A is a non-empty and finite set of arguments, and $R \subseteq A \times A$ a pair arguments representing direct attacks[1] of arguments. In argumentation, we are interested in computing so-called *extensions*, which are subsets $S \subseteq A$ of the arguments that meet certain properties according to certain semantics as given below. An argument $s \in S$, is called *defended by S in F* if for every $(s', s) \in R$, there exists $s'' \in S$ such that $(s'', s') \in R$. The family $\mathrm{def}_F(S)$ is defined by $\mathrm{def}_F(S) := \{\, s \mid s \in A, s \text{ is defended by } S \text{ in } F \,\}$. We say $S \subseteq A$ is *conflict-free in S* if $(S \times S) \cap R = \emptyset$; S is *admissible in F* if (i) S is *conflict-free in F*, and (ii) every $s \in S$ is *defended by S in F*. Assume an admissible set S. Then, (iiia) S is *complete in F* if $\mathrm{def}_F(S) = S$; (iiib) S is *preferred in F*, if there is no $S' \supset S$ that is *admissible in F*; (iiic) S is *semi-stable in F* if there is no admissible set $S' \subseteq A$ in F with $S^+_R \subsetneq (S')^+_R$ where $S^+_R := S \cup \{\, a \mid (b, a) \in R, b \in S \,\}$; (iiid) S is *stable in F* if every $s \in A \setminus S$ is *attacked* by some $s' \in S$. A conflict-free set S is *stage in F* if there is no conflict-free set $S' \subseteq A$ in F with $S^+_R \subsetneq (S')^+_R$. Let ALL abbreviate the set {admissible, complete, preferred, semi-stable, stable, stage}. For a semantics $\mathcal{S} \in$ ALL, $\mathcal{S}(F)$ denotes the set of *all extensions* of semantics $\mathcal{S}$ in F. In general $\mathrm{stable}(F) \subseteq \text{semi-stable}(F) \subseteq \mathrm{preferred}(F) \subseteq \mathrm{complete}(F) \subseteq \mathrm{admissible}(F) \subseteq \text{conflict-free}(F)$ and $\mathrm{stable}(F) \subseteq \mathrm{stage}(F) \subseteq \text{conflict-free}(F)$.

Problems of Interest. In argumentation one is usually interested in credulous and skeptical reasoning problems. In this paper, we are in addition interested in counting versions of these problems. Therefore, let $\mathcal{S} \in$ ALL be an abstract argumentation semantic, $F = (A, R)$ be an argumentation framework, and $a \in A$ an argument. The *credulous reasoning problem* $\mathsf{Cred}_\mathcal{S}$ asks to decide whether there is an $\mathcal{S}$-extension of F that contains the (credulous) argument a. The *skeptical reasoning problem* $\mathsf{Skep}_\mathcal{S}$ asks to decide whether all $\mathcal{S}$-extensions of F contain the argument a. The *credulous counting problem* $\#\mathsf{Cred}_\mathcal{S}$ asks to output the number of $\mathcal{S}$-extensions of F that contain a, i.e., $|\{\, S \mid S \in \mathcal{S}(F), a \in S \,\}|$. The *projected credulous counting problem* ($\#\mathsf{PCred}_\mathcal{S}$) asks to output the number of $\mathcal{S}$-extensions restricted to the projection arguments P, i.e., $|\{\, S \cap P \mid S \in \mathcal{S}(F), a \in S \,\}|$.

Example 1. *Consider framework F from Figure 1, which depicts a framework for deciding between surfing and drinking cocktails. Framework F admits three stable*

[1] Given $S, S' \subseteq A$. Then, $S \rightarrowtail_R S'$ denotes $\{s \in S \mid (\{s\} \times S') \cap R \neq \emptyset\}$, and $S \leftarrowtail_R S' := \{s \in S \mid (S' \times \{s\}) \cap R \neq \emptyset\}$.

extensions $\mathrm{stable}(F) = \{\{d,r,c\},\{s,a,c\},\{s,a,p\}\}$. $\#\mathsf{Cred}_{\mathrm{stable}}$ *for argument* s *equals* 2*, whereas* $\#\mathsf{PCred}_{\mathrm{stable}}$ *for argument* s *restricted to* $P := \{a,r\}$ *equals* 1.

Tree Decompositions (TDs). For a tree $T = (N, A, n)$ with root n and a node $t \in N$, we let $\mathrm{children}(t, T)$ be the sequence of all nodes t' in arbitrarily but fixed order, which have an edge $(t, t') \in A$. Let $G = (V, E)$ be a graph. A *tree decomposition (TD)* of graph G is a pair $\mathcal{T} = (T, \chi)$, where $T = (N, A, n)$ is a rooted tree, $n \in N$ the root, and χ a mapping that assigns to each node $t \in N$ a set $\chi(t) \subseteq V$, called a *bag*, such that the following conditions hold: (i) $V = \bigcup_{t\in N} \chi(t)$ and $E \subseteq \bigcup_{t\in N} \{\, \{u, v\} \mid u, v \in \chi(t) \,\}$; and (ii) for each r, s, t, such that s lies on the path from r to t, we have $\chi(r) \cap \chi(t) \subseteq \chi(s)$. Then, $\mathrm{width}(\mathcal{T}) := \max_{t\in N} |\chi(t)| - 1$. The *treewidth* $tw(G)$ of G is the minimum $\mathrm{width}(\mathcal{T})$ over all tree decompositions $\mathcal{T}$ of G. For arbitrary but fixed $w \geq 1$, it is feasible in linear time to decide if a graph has treewidth at most w and, if so, to compute a TD of width w (Bodlaender 1996). However, in practice, heuristics (Abseher, Musliu, and Woltran 2017; Fichte, Lodha, and Szeider 2017) to compute a tree decomposition are often sufficient. In order to simplify case distinctions in the algorithms, *we assume nice TDs*, which can be computed in linear time without increasing the width (Kloks 1994) and are defined as follows. For a node $t \in N$, we say that $\mathrm{type}(t)$ is *leaf* if $\mathrm{children}(t, T) = \langle\rangle$; *join* if $\mathrm{children}(t, T) = \langle t', t''\rangle$ where $\chi(t) = \chi(t') = \chi(t'') \neq \emptyset$; *int* ("introduce") if $\mathrm{children}(t, T) = \langle t'\rangle$, $\chi(t') \subseteq \chi(t)$ and $|\chi(t)| = |\chi(t')| + 1$; *rem* ("remove") if $\mathrm{children}(t, T) = \langle t'\rangle$, $\chi(t') \supseteq \chi(t)$ and $|\chi(t')| = |\chi(t)| + 1$. If for every node $t \in N$, $\mathrm{type}(t) \in \{$*leaf*, *join*, *int*, *rem*$\}$ and bags of leaf nodes and the root are empty, then the TD is called *nice*.

Classical Counting Complexity

In this section, we investigate the classical counting complexity of the credulous reasoning problem. Proofs of marked statements ("$\star$") are omitted or shortened. For full details, see the technical report (Fichte et al. 2018).

Lemma 2 ($\star$)**.** $\#\mathsf{Cred}_{\mathcal{S}}$ *is in* $\# \cdot \mathrm{P}$ *if* $\mathcal{S} \in \{$*conflict-free, stable, admissible, complete*$\}$*, and in* $\# \cdot \mathrm{coNP}$ *if* $\mathcal{S} \in \{$*preferred, semi-stable, stage*$\}$.

The next lemma does neither consider conflict-free nor preferred extensions.

Lemma 3 ($\star$)**.** $\#\mathsf{Cred}_{\mathcal{S}}$ *is* $\# \cdot \mathrm{P}$*-hard under parsi. reductions if* $\mathcal{S} \in \{$*stable, admissible, complete*$\}$*, and* $\# \cdot \mathrm{coNP}$*-hard under subtr. reductions if* $\mathcal{S} \in \{$*semi-stable, stage*$\}$.

Proof (Sketch). (1) Start with the case of stable or complete extensions. Following the construction of Dunne and Bench-Capon (2002), we parsimoniously reduce from $\#\mathsf{SAT}$. For the case of admissible extensions, to count correctly, it is crucial that for each x_i either argument x_i or $\bar{x}_i$ is part of the extension. To ensure this, we introduce arguments $s_1, \ldots, s_n$ attacking t that can only be defended by one of x_i or $\bar{x}_i$.

(2) The formalism of circumscription is well-established in the area of AI (McCarthy 1980). Formally, one considers assignments of Boolean formulas that are *minimal* regarding the *partially ordered set* of truth assignments: if $s = (s_1, \ldots, s_n), s' = (s'_1, \ldots, s'_n) \in \{0,1\}^n$, then write $s < s'$ if $s \neq s'$ and $s_i \leqslant s'_i$ for every $i \leqslant n$. Then, we define the problem $\#\mathsf{Circumscription}$ which asks given a Boolean formula φ in CNF to output the number of minimal models of φ. Durand, Hermann, and Kolaitis (2005) showed that $\#\mathsf{Circumscription}$ is $\# \cdot \mathrm{coNP}$-complete via subtractive reductions (a generalization of parsimonious reductions). The crux is, that choosing negative literals is more valuable than selecting positive ones. This is achieved by adding additionally attacked arguments to each negative literal. □

Lemma 2 and 3 together show the following theorem.

Theorem 4. $\#\mathsf{Cred}_{\mathcal{S}}$ *is* $\# \cdot \mathrm{P}$*-complete under parsimonious reductions if* $\mathcal{S} \in \{$*stable, admissible, complete*$\}$*, and* $\# \cdot \mathrm{coNP}$*-complete under subtractive reductions if* $\mathcal{S} \in \{$*semi-stable, stage*$\}$.

Now, consider the case of projected counting.

Lemma 5 ($\star$)**.** $\#\mathsf{PCred}_{\mathcal{S}}$ *is (1) in* $\# \cdot \mathrm{NP}$ *if* $\mathcal{S} \in \{$*stable, admissible, complete*$\}$*, and (2) in* $\# \cdot \Sigma_2^{\mathrm{P}}$ *if* $\mathcal{S} \in \{$*semi-stable, stage*$\}$.

Proof (Sketch). Given a framework, a projection set, and an argument a. We non-deterministically branch on a possible projected extension S. Accordingly, we have $S \subseteq P$. If $a \in S$ and S is of the respective semantics, then we accept. Otherwise we make a non-deterministic guess $S' \supseteq S$, verify if $P \cap S' = S$, $a \in S'$, and S' is of the desired semantics. Extension verification is for (1) in P, and for (2) in coNP. Concluding, we get an NP oracle call for the first case, and an $\mathrm{NP}^{\mathrm{coNP}} = \mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{P}}$ oracle call in the second case. □

Consider the problem $\#\Sigma_k\mathsf{SAT}$, which asks, given $\varphi(Y) = \exists x_1 \forall x_2 \cdots Q_k x_k \psi(X_1, \ldots, X_k, Y)$, where ψ is a propositional DNF if k is even (and CNF if k is odd), X_i, for each i, and Y are sets of variables, to output the number of truth assignments to the variables from Y that satisfy φ. Durand, Hermann, and Kolaitis (2005) have shown that the problem is $\# \cdot \Sigma_k^{\mathrm{P}}$-complete via parsimonious reductions.

Lemma 6 ($\star$)**.** $\#\mathsf{PCred}_{\mathcal{S}}$ *is (1)* $\# \cdot \Sigma_2^{\mathrm{P}}$*-hard w.r.t. parsimonious reductions if* $\mathcal{S} \in \{$*stage, semi-stable*$\}$*, and (2)* $\# \cdot \mathrm{NP}$*-hard w.r.t. parsimonious reductions if* $\mathcal{S} \in \{$*admissible, stable, complete*$\}$.

Proof (Sketch). (1) We state a parsimonious reduction from $\#\Sigma_2\mathsf{SAT}$ to $\#\mathsf{PCred}_{\mathcal{S}}$. We use an extended version of the construction of Dvořák and Woltran (2010). Given a formula $\varphi(X) = \exists Y \forall Z\ \psi(X, Y, Z)$, where X, Y, Z are sets of variables, and ψ is a DNF. Consider now the negation of $\varphi(X)$, i.e., $\varphi'(X) = \neg\varphi(X) \equiv \forall Y \exists Z\ \neg\psi(X, Y, Z)$. Let $\psi'(X, Y, Z)$ be $\neg\psi(X, Y, Z)$ in NNF. Accordingly, ψ' is a CNF, $\psi'(X, Y, Z) = \bigwedge_{i=1}^{m} C_i$ and C_i is a disjunction of literals for $1 \leqslant i \leqslant m$. Note that, the formula $\varphi'(X)$ is of the same kind as the formula in the construction of Dvořák and Woltran (2010). Now define an argumentation framework $AF = (A, R)$, where $A = \{\, x, \bar{x}, y, \bar{y}, y', \bar{y}', z, \bar{z} \mid x \in X, y \in Y, z \in Z \,\} \cup \{t, \bar{t}, b\}$ and

$$R = \{\, (y', y'), (\bar{y}', \bar{y}'), (y, y'), (\bar{y}, \bar{y}'), (y, \bar{y}), (\bar{y}, y) \mid y \in Y \}$$
$$\cup \{(b,b), (t,\bar{t}), (\bar{t},t), (t,b)\} \cup \{\, (C_i, t) \mid 1 \leqslant i \leqslant m \,\}$$
$$\cup \{\, (u, C_i), (\bar{u}, C_i) \mid u \in X \cup Y \cup Z, u \in C_i, 1 \leqslant i \leqslant m \,\}$$

Note that, by construction, the $y', \bar{y}'$ variables make the extensions with respect to the universally quantified variables y incomparable. Further observe that choosing t is superior to selecting $\bar{t}$, as t increases the range by one more. If for every assignment over the Y-variables there exists an assignment to the Z-variables, then, each time, when there is a possible solution to $\psi'(X,Y,Z)$, so semantically $\neg\psi(X,Y,Z)$, w.r.t. the free X-variables, the extension will contain t. As a result, the extensions containing t correspond to the unsatisfying assignments. Let $A(\varphi(X))$ be the set of assignments of a given $\#\Sigma_2\mathsf{SAT}$-formula, and $B(AF,P,a)$ be the set of stage/semi-stable extensions which contain a and are projected to P. Then, one can show that $|A(\varphi(X))| = |B(AF,X,\bar{t})|$ proving the desired reduction (as $\bar{t}$ together with the negation of $\varphi(X)$ in the beginning, intuitively, is a double negation yielding a reduction from $\#\Sigma_2\mathsf{SAT}$).

(2) Now turn to the case of admissible, stable, or complete extensions. Again, we provide a similar parsimonious reduction, but this time, from $\#\Sigma_1\mathsf{SAT}$ to $\#\mathsf{PCred}_{\mathcal{S}}$. □

Theorem 7. *$\#\mathsf{PCred}_{\mathcal{S}}$ is $\# \cdot \mathrm{NP}$-complete via parsi. reductions if $\mathcal{S} \in \{$stable, admissible, complete$\}$, and $\# \cdot \Sigma_2^{\mathrm{P}}$-complete via parsi. reductions if $\mathcal{S} \in \{$stage, semi-stable$\}$.*

Similarly, one can introduce problems of the form $\#\mathsf{Skep}_{\mathcal{S}}$ and $\#\mathsf{PSkep}_{\mathcal{S}}$ corresponding to the counting versions of the skeptical reasoning problem. As skeptical is dual to credulous reasoning, one easily obtains completeness results for the dual counting classes.

DP for Abstract Argumentation

In this section, we recall DP techniques from the literature to solve skeptical and credulous reasoning in abstract argumentation. Additionally, we establish lower bounds for exploiting treewidth in algorithms that solve these problems for the most common semantics. Therefore, let $F = (A,R)$ be a given argumentation framework and $\mathcal{S}$ be an argumentation semantics. While an abstract argumentation framework can already be seen as a digraph, treewidth is a measure for undirected graphs. Consequently, we consider for framework F the *underlying graph* G_F, where we simply drop the direction of every edge, i.e., $G_F = (A,R')$ where $R' := \{\{u,v\} \mid (u,v) \in R\}$. Let $\mathcal{T} = (T,\chi)$ be a TD of the underlying graph of F. Further, we need some auxiliary definitions. Let $T = (N,\cdot,n)$ and $t \in N$. Then, $\text{post-order}(T,n)$ defines a sequence of nodes for tree T rooted at n in *post-order* traversal. The *bag-framework* is defined as $F_t := (A_t,R_t)$, where $A_t := A \cap \chi(t)$ and $R_t := (A_t \times A_t) \cap R$, the *framework below t* as $F_{\leqslant t} := (A_{\leqslant t}, R_{\leqslant t})$, where $A_{\leqslant t} := \{a \mid a \in \chi(t), t' \in \text{post-order}(T,t)\}$, and $R_{\leqslant t} := (A_{\leqslant t} \times A_{\leqslant t}) \cap R$. It holds that $F_n = F_{\leqslant n} = F$.

A standard approach (Bodlaender and Kloks 1996) to benefit algorithmically from small treewidth is to design DP algorithms, which traverse a given TD and run at each node a so-called *local algorithm* $\mathbb{A}$. The local algorithm does a case distinctions based on the types $\text{type}(t)$ of a nice TD and stores information in a table, which is a set of rows where a *row* $\vec{u}$ is a sequence of fixed length (and the length is bounded by the treewidth). Later, we traverse the TD multiple times. We access also information in tables computed in previous traversals and formalize access to previously computed tables in *tabled tree decomposition (TTD)* by taking in addition to the TD $\mathcal{T} = (T,\chi)$ a mapping τ that assigns to a node t of T also a table. Then, the TTD is the triple $\mathcal{T} = (T,\chi,\tau)$. Later, for simple use in algorithms, we assume $\tau(t)$ is initialized by the empty set for every node t of T. To solve the considered problem, we run the following steps:

1. Compute a TD $\mathcal{T} = (T,\chi)$ of the underlying graph of F.
2. Run algorithm $\mathtt{DP}_{\mathbb{A}}$, which takes a TTD $\mathcal{T} = (T,\chi,\iota)$ with $T = (N,\cdot,n)$ and traverses T in post-order. At each node $t \in N$ it stores the result of algorithm $\mathbb{A}$ in table $o(t)$. Algorithm $\mathbb{A}$ can access only information that is restricted to the currently considered bag, namely, the type of the node t, the atoms in the bag $\chi(t)$, the bag-framework F_t, and every table $o(t')$ for any child t' of t.
3. Print the solution by interpreting table $o(n)$ for root n of the resulting TTD (T,χ,o).

Credulous Reasoning. DP algorithms for credulous reasoning of various semantics have already been established in the literature (Dvořák, Pichler, and Woltran 2012) and their implementations are also of practical interest (Dvořák et al. 2013). While a DP algorithm for semi-stable (Bliem, Hecher, and Woltran 2016) semantics was presented as well, stage semantics has been missing. This section fills the gap by introducing a local algorithm for this case. The worst case complexity of these algorithms depends on the semantics and ranges from single to double exponential in the treewidth. In the following, we take these algorithms from the literature, simplify them and adapt them to solve $\#\mathsf{PCred}$ for the various semantics. First, we present the algorithm $\mathtt{DP}_{\mathbb{ADM}}$ that uses the algorithm in Listing 1 as local algorithm to solve credulous reasoning for the admissible semantics. $\mathtt{DP}_{\mathbb{ADM}}$ outputs a new TTD that we use to solve our actual counting problem. At each node t, we store in table $o(t)$ rows of the form $\vec{u} = \langle I,O,D\rangle$ and construct parts of extensions. The first position of the rows consists of a set $I \subseteq \chi(t)$ of arguments that will be considered for a part of an extension; we write $E(\vec{u}) := I$ to address this extension part. The second

Listing 1: Local algorithm $\mathbb{ADM}(t,\chi_t,\cdot,(F_t,c,\cdot),\langle\tau_1,\tau_2\rangle)$, c.f., (Dvořák, Pichler, and Woltran 2012).

In: Node t, bag χ_t, bag-framework $F_t = (A_t,R_t)$, credulous argument c, and $\langle\tau_1,\tau_2\rangle$ is the sequence of tables of children of t.
Out: Table τ_t.

```
if type(t) = leaf then τ_t ← {⟨∅,∅,∅⟩}
else if type(t) = int and a ∈ χ_t is introduced argum. then
    τ_t ← {⟨J, O^⊎_{A_t ↛_{R_t} J}, D^⊎_{J ↚_{R_t} A_t}⟩ | ⟨I,O,D⟩ ∈ τ_1, J ∈
          {I, I_a^+}, J ↛_{R_t} J = ∅, J ∩ {c} = χ(t) ∩ {c}}
else if type(t) = rem and a ∉ χ_t is removed argum. then
    τ_t ← {⟨I_a^-, O_a^-, D_a^-⟩ | ⟨I,O,D⟩ ∈ τ_1, a ∉ O \ D}
else if type(t) = join then
    τ_t ← {⟨I, O_1 ⊎_{O_2}, D_1 ⊎_{D_2}⟩ | ⟨I,O_1,D_1⟩∈τ_1, ⟨I,O_2,D_2⟩∈τ_2}
return τ_t
```

$S^{\uplus}_{S'} := S \cup S'$, $S^+_e := S \cup \{e\}$, and $S^-_e := S \setminus \{e\}$.

position consists of a set $O \subseteq \chi(t) \setminus I$ that represents arguments that attack any other argument of the extension part. Finally, the third position is the set $D \subseteq \chi(t)$ of arguments in the current bag that are already defeated (counterattacked) by any argument in the extension, and therefore in a sense compensate the set O of attacking arguments. The idea of the algorithm is as follows. For nodes with $\mathrm{type}(t) = \textit{leaf}$, Line 1 initially sets the extension part I, set O of attackers, and set D of defeated arguments to the empty set. Intuitively, in Line 3 whenever we encounter an argument a for the first time while traversing the TD ($\mathrm{type}(t) = \textit{int}$), we guess whether $a \in I$ or $a \notin I$. Further, we ensure that I is conflict-free and that we construct only rows where $c \in I$ if $a = c$. Since ultimately every argument has to be defended by the extension, we keep track of attacking arguments in O and defeated arguments D. In Line 5, whenever we remove an argument a ($\mathrm{type}(t) = \textit{rem}$), we are not allowed to store a in the table any more as the length of a row $\vec{u}$ in the table $o(t)$ depends on the arguments that occur in the bag $\chi(t)$; otherwise we would exceed the length and loose the bound on the treewidth. However, we have to ensure that either a is not an attacking argument ($a \notin O$), or that a was defeated at some point ($a \in D$). In the end, Condition (ii) of a TD ensures that whenever an argument does not occur in the bag any more, we encountered its entire involvement in the attack relation. Finally, Line 7 ensures that we only combine rows that agree on the extension and combine information concerning attacks and defeats accordingly. This case can be seen as a combination of database joins ($\mathrm{type}(t) = \textit{join}$). $\mathbb{ADM}$ can vacuously be extended to an algorithm $\mathbb{STAB}$ for stable semantics. There one simply drops the set O and ensures in Line 5 that the removed atom a is either in the extension part I or defeated (in $a \in D$). An algorithm $\mathbb{COMP}$ for the complete semantics requires some additional technical effort. There one can distinguish five states, namely elements that are in the extension, defeated "candidates", already defeated, candidates for not being in the extension (unrelated), or actually proven to be unrelated.

In the following proposition, we give more precise runtime upper bounds for the algorithms presented in the literature (Dvořák, Pichler, and Woltran 2012) that can be obtained by employing sophisticated data structures, especially for handling nodes t with $\mathrm{type}(t) = \textit{join}$.

Proposition 8 ($\star$). *Algorithm* $\mathtt{DP}_{\mathbb{STAB}}$ *runs in time* $\mathcal{O}(3^k \cdot k \cdot g)$, $\mathtt{DP}_{\mathbb{ADM}}$ *in* $\mathcal{O}(4^k \cdot k \cdot g)$, *and* $\mathtt{DP}_{\mathbb{COMP}}$ *in* $\mathcal{O}(5^k \cdot k \cdot g)$ *where* k *is the width and* g *the number of nodes of the TD.*

The definitions of preferred, semi-stable, and stage semantics involve subset-maximization. Therefore, one often introduces a concept of witness (extension part) and counter-witness in the rows in DP, where the counter-witness tries to invalidate subset-maximality of the corresponding witness (Jakl, Pichler, and Woltran 2009). In the counter-witness one stores sets of arguments that are supersets of the considered extension, such that, in the end, at the root there was no superset of an extension in the counter-witness while traversing the TD. In other words, for a witness the counter-witness failed to invalidate maximality and accordingly the witness is subset-maximal. In the literature, algorithms that involving

Listing 2: Local algorithm $\mathbb{STAG}(t, \chi_t, \cdot, (F_t, c, \cdot), \langle \tau_1, \tau_2 \rangle)$.

In: Node t, bag χ_t, bag-framework $F_t = (A_t, R_t)$, credulous argument c, and $\langle \tau_1, \tau_2 \rangle$ is the sequence of tables of children of t.
Out: Table τ_t.

1 **if** $\mathrm{type}(t) = \textit{leaf}$ **then** $\tau_t \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$
2 **else if** $\mathrm{type}(t) = \textit{int}$ *and* $a \in \chi_t$ *is introduced argum.* **then**
3 $\quad \tau_t \leftarrow \{\langle J, \mathcal{A}^{\uplus}_{J \rightarrowtail_{R_t} A_t}, AC, \mathcal{C}^{\oplus}_{\langle J, \mathcal{A}, AC \rangle}(a) \rangle \mid \langle I, \mathcal{A}, \mathcal{AC}, \mathcal{C} \rangle \in \tau_1, (J, AC) \in \mathrm{States}_a(I, \mathcal{AC}), J \cap \{c\} = \chi(t) \cap \{c\}\}$
4 **else if** $\mathrm{type}(t) = \textit{rem}$ **and** $a \notin \chi_t$ *is removed argum.* **then**
5 $\quad \tau_t \leftarrow \{\langle I^-_a, \mathcal{A}^-_a, \mathcal{AC}^-_a, \mathcal{C}^{\sim}_a \rangle \mid \langle I, \mathcal{A}, \mathcal{AC}, \mathcal{C} \rangle \in \tau_1, a \in I \cup \mathcal{A}\}$
6 **else if** $\mathrm{type}(t) = \textit{join}$ **then**
7 $\quad \tau_t \leftarrow \{\langle I, \mathcal{A}_1 \uplus_{\mathcal{A}_2} \mathcal{A}_2, \mathcal{AC}, (\mathcal{C}_1 \bowtie \mathcal{C}_2) \cup (\mathcal{C}_1 \bowtie \{\langle u_2, \bot \rangle\}) \cup (\{\langle u, \bot \rangle\} \bowtie \mathcal{C}_2) \rangle \mid u_1 \in \tau_1, u_2 \in \tau_2, u_1 = \langle I, \mathcal{A}_1, \mathcal{AC}_1, \mathcal{C}_1 \rangle, u_2 = \langle I, \mathcal{A}_2, \mathcal{AC}_2, \mathcal{C}_2 \rangle\}$
8 **return** τ_t

$$\mathrm{States}_a(I, \mathcal{AC}) := \{\, (J, AC) \mid J \in \{I, I^+_a\}, AC \in \{\mathcal{AC}, \mathcal{AC}^+_a\}, J \cap AC = \emptyset, [J \rightarrowtail_{R_t} J] = \emptyset, [A_t \leftarrowtail_{R_t} J] \subseteq AC \,\}$$

$$\mathcal{C}^{\oplus}_{\langle J', \mathcal{A}', AC' \rangle}(a) := \{\, \langle \langle J, \mathcal{A}^{\uplus}_{J \rightarrowtail_{R_t} A_t}, AC \rangle, (J^{\uplus}_{AC} \subsetneq J'^{\uplus}_{AC'}) \vee s \rangle \mid \langle \langle I, \mathcal{A}, \mathcal{AC} \rangle, s \rangle \in \mathcal{C}^+_{\langle J, \mathcal{A}, AC \rangle, \bot}, (J, AC) \in \mathrm{States}_a(I, \mathcal{AC}), J \cap \{c\} = \chi(t) \cap \{c\} \,\}$$

$$\mathcal{C}^{\sim}_a := \{\, \langle \langle I^-_a, \mathcal{A}^-_a, \mathcal{AC}^-_a \rangle, \sigma \rangle \mid \langle \langle I, \mathcal{A}, \mathcal{AC} \rangle, \sigma \rangle \in \mathcal{C}, a \in I \cup \mathcal{A} \}$$

$$\mathcal{C}_1 \bowtie \mathcal{C}_2 := \{\, \langle \langle I, \mathcal{A}_1 \uplus_{\mathcal{A}_2} \mathcal{A}_2, \mathcal{AC} \rangle, \sigma_1 \vee \sigma_2 \rangle \mid \langle \langle I, \mathcal{A}_1, \mathcal{AC} \rangle, \sigma_1 \rangle \in \mathcal{C}_1, \langle \langle I, \mathcal{A}_2, \mathcal{AC} \rangle, \sigma_2 \rangle \in \mathcal{C}_2 \,\}.$$

such an interplay between witnesses and counter-witnesses have been defined for preferred and semi-stable semantics, we simply refer to them as $\mathtt{DP}_{\mathbb{PREF}}$ and $\mathtt{DP}_{\mathbb{SEMI}}$. For the stage semantics, we provide the algorithm in Listing 2. Intuitively, we compute conflict-free extensions during the TD traversal and additionally guess candidates $\mathcal{AC}$ that ultimately have to be attacked ($\mathcal{A}$) by the extension part J. This allows us then to subset-maximize upon the range part $J \cup \mathcal{AC}$, by trying to find counter-witnesses $\mathcal{C}$ to subset-maximality. Again a more detailed runtime analysis yields the following result.

Proposition 9 ($\star$). *Algorithms* $\mathtt{DP}_{\mathbb{PREF}}$, $\mathtt{DP}_{\mathbb{SEMI}}$, *and* $\mathtt{DP}_{\mathbb{STAG}}$ *run in time* $\mathcal{O}(2^{2^{4k+1}} \cdot g)$ *where* k *is the width and* g *the number of nodes of the TD.*

Lower Bounds. A natural question is whether we can significantly improve the algorithms stated in Propositions 8 and 9. In other words, we are interested in lower bounds on the runtime of an algorithm that exploits treewidth for credulous reasoning. A common method in complexity theory is to assume that the *exponential time hypothesis (ETH)* holds and establish reductions. The ETH states that there is some real $s > 0$ such that we cannot decide satisfiability of a given 3-CNF formula φ in time $2^{s \cdot |\varphi|} \cdot \|\varphi\|^{\mathcal{O}(1)}$ (Cygan et al. 2015, Ch.14). Subsequently, we establish lower bounds, assuming ETH employing known reductions from the literature, showing that there is no hope for a better algorithm.

Theorem 10 ($\star$). *Let* $\mathcal{S} \in \{$*admissible, complete, stable*$\}$, F *be a framework and* k *the treewidth of the underlying graph* G_F. *Unless ETH fails,* $\mathsf{Cred}_{\mathcal{S}}$ cannot *be solved in*

time $2^{o(k)} \cdot \|F\|^{o(k)}$ *and for* $\mathcal{S} =$ *semi-stable,* $\mathsf{Cred}_\mathcal{S}$ *and* $\mathsf{Skep}_\mathcal{S}$ cannot *be solved in time* $2^{2^{o(k)}} \cdot \|F\|^{o(k)}$.

Proof (Idea). The existing reductions by Dunne and Bench-Capon (2002) increase the treewidth only linearly and are hence sufficient. For semi-stable statements the reductions by Dvořák and Woltran (2010) can be applied, since preferred and semi-stable extensions of the constructed argumentation framework coincide. □

Algorithms for Projected Credulous Counting by Exploiting Bounded Treewidth

In the previous section, we presented algorithms to solve the reasoning problems for abstract argumentation. These algorithms can be extended relatively straightforward to also count extensions without projection by adding counters to each row at a quadratic runtime instead of linear in the size of the input instance. One can even reconstruct extensions (Pichler, Rümmele, and Woltran 2010). However, things are more complicated for projected credulous counting.

In this section, we present an algorithm $\texttt{PCNT}_\mathbb{S}$ that solves the projected credulous counting problem ($\#\mathsf{PCred}_\mathcal{S}$) for semantics $\mathcal{S} \in$ ALL. Our algorithm lifts results for projected model counting in the computationally and conceptually much easier setting of propositional satisfiability (Fichte et al. 2018) to abstract argumentation. Our algorithm is based on dynamic programming and traverses a TD three times. To this end, we employ algorithms $\mathbb{S} \in \{\mathbb{ADM}, \mathbb{COMP}, \mathbb{PREF}, \mathbb{STAG}, \mathbb{SEMI}, \mathbb{STAB}\}$ as presented in the previous section according to the considered semantics $\mathcal{S}$. The first traversal consists of $\texttt{DP}_\mathbb{S}$, where $\mathbb{S}$ is a local algorithm for credulous reasoning of the chosen semantics, which results in TTD $\mathcal{T}_{\mathbb{S}\text{-}Cred} = (T, \chi, \tau)$.

In the following, let again $F = (A, R)$ be the given framework, $a \in A$ an argument, (T, χ) a TD of G_F with $T = (N, \cdot, n)$ and the root n, and $\mathcal{T}_{\mathbb{S}\text{-}Cred} = (T, \chi, \tau)$ be the TTD that has been computed by the respective algorithms as described in the previous section. Then, we intermediately traverse $\mathcal{T}_{\mathbb{S}\text{-}Cred}$ in pre-order and *prune irrelevant rows*, thereby we remove all rows that cannot be extended to a credulous extension of the corresponding semantics $\mathcal{S}$. We call the resulting TTD $\mathcal{T}_{\mathbb{S}\text{-}Pruned} = (T, \chi, \nu)$. Note that pruning does not affect correctness, as only rows are removed where already the count without even considering projection is 0. However, pruning serves as a technical trick for the last traversal to avoid corner cases, which result in correcting counters and backtracking.

In the final traversal, we count the projected credulous extensions. Therefore, we compute a TTD $\mathcal{T}_{\mathbb{S}\text{-}Proj} = (T, \chi, \pi)$ using algorithm $\texttt{DP}_{\mathbb{PROJ}}$ using local algorithm $\mathbb{PROJ}$ as given in Listing 3. Algorithm $\mathbb{PROJ}$ stores for each node a pair $\langle \sigma, c \rangle \in \pi(t)$, where $\sigma \subseteq \nu(t)$ is a table $\nu(t)$ from the previous traversal and $c \geq 0$ is an integer representing what we call the *intersection projected count (*ipc*)*.

Before we start with explaining how to obtain these ipc values c, we require auxiliary notations from the literature. First, we require a notion to reconstruct extensions from $\mathcal{T}$, more precisely, for a given row to define its predecessor rows in the corresponding child tables. Therefore, let t be a node of T with children t_1 and t_2, if it exists. Since sequences used in the following depend on number of the children assume for simplicity of the presentation that sequences are implicitly of corresponding length even if they are given as of length 2. For sequence $\vec{s} = \langle s_1, s_2 \rangle$, let $\langle\{\vec{s}\}\rangle := \langle \{s_1\}, \{s_2\} \rangle$. For a given row $\vec{u} \in \tau(t)$, we define the originating rows of $\vec{u}$ in node t by $\mathrm{origins}(t, \vec{u}) := \{ \vec{s} \mid \vec{s} \in \tau(t_1) \times \tau(t_2), \vec{u} \in \mathbb{S}(t, \chi(t), \cdot, (F_t, \cdot), \langle\{\vec{s}\}\rangle) \}$ and for a table σ as the union over the origins for all rows $\vec{u} \in \sigma$. Next, let $\sigma \subseteq \nu(t)$. In order to combine rows and solve projection accordingly, we need equivalence classes of rows. Let therefore relation $=_\mathrm{P} \subseteq \sigma \times \sigma$ consider equivalent rows with respect to the projection of its extension part by $=_\mathrm{P} := \{ (\vec{u}, \vec{v}) \mid \vec{u}, \vec{v} \in \sigma, E(\vec{u}) \cap P = E(\vec{v}) \cap P \}$. Let $\mathrm{buckets}_P(\sigma)$ be the set of equivalence classes induced by $=_\mathrm{P}$ on σ, i.e., $\mathrm{buckets}_P(\sigma) := (\sigma / =_\mathrm{P}) = \{ [\vec{u}]_P \mid \vec{u} \in \sigma \}$, where $[\vec{u}]_P = \{ \vec{v} \mid \vec{v} =_\mathrm{P} \vec{u}, \vec{v} \in \sigma \}$ (Wilder 1965).

When computing the ipc values c stored in each row $\vec{u}$ of $\pi(t)$, we compute a so-called *projected count (*pc*)* as follows. First, we define the *stored* ipc of $\sigma \subseteq \nu(t)$ in table $\pi(t)$ by $\text{s-ipc}(\pi(t), \sigma) := \sum_{\langle \sigma, c \rangle \in \pi(t)} c$. We use the ipc value in the context of "accessing" ipc values in table $\pi(t_i)$ for a child t_i of t. This can be generalized to a sequence $s = \langle \pi(t_1), \pi(t_2) \rangle$ of tables and a set $O = \{\langle \sigma_1, \sigma_2 \rangle, \langle \sigma_1', \sigma_2' \rangle, \ldots\}$ of sequences of tables by $\text{s-ipc}(s, O) = \text{s-ipc}(s_{(1)}, O_{(1)}) \cdot \text{s-ipc}(s_{(2)}, O_{(2)})$. Then, the *projected count* pc of rows $\sigma \subseteq \nu(t)$ is the application of the inclusion-exclusion principle to the stored intersection projected counts, i.e., ipc values of children of t. Therefore, pc determines the origins of table σ, and uses the stored counts (s-ipc) in the $\mathbb{PROJ}$-tables of the children t_i of t for all subsets of these origins. Formally, we define

$$\mathrm{pc}(t, \sigma, \langle \pi(t_1), \pi(t_2) \rangle) := \sum_{\emptyset \subsetneq O \subseteq \mathrm{origins}(t,\sigma)} (-1)^{(|O|-1)} \cdot \text{s-ipc}(\langle \pi(t_1), \pi(t_2) \rangle, O).$$

Intuitively, pc defines the number of distinct projected extensions in framework $F_{\leqslant t}$ to which any row in σ can be extended. Finally, the *intersection projected count* ipc for σ is the result of another application of the inclusion-exclusion principle. It describes the number of common projected $\mathcal{S}$-extensions which the rows in σ have in common in framework $F_{\leqslant t}$. We define $\mathrm{ipc}(t, \sigma, s) := 1$ if $\mathrm{type}(t) =$ *leaf* and otherwise $\mathrm{ipc}(t, \sigma, s) := \big| \mathrm{pc}(t, \sigma, s) + \sum_{\emptyset \subsetneq \varphi \subsetneq \sigma} (-1)^{|\varphi|} \cdot \mathrm{ipc}(t, \varphi, s) \big|$, where $s = \langle \pi(t_1), \pi(t_2) \rangle$. In other words, if a node is of type *leaf*, ipc is one, since bags of leaf nodes are empty. Observe that since bags $\chi(n)$ for root node n are empty, there is only one entry in $\pi(n)$ and $\mathrm{pc}(n, \nu(n), s) = \mathrm{ipc}(n, \nu(n), s)$, which corresponds to the number of projected credulous extensions. In the end, we collect pc-values for all subsets of $\nu(t)$.

Theorem 11 (⋆). *Algorithm* $\texttt{PCNT}_\mathbb{S}$ *is correct and solves* $\#\mathsf{PCred}_\mathcal{S}$ *for local algorithms* $\mathbb{S} \in \{\mathbb{ADM}, \mathbb{COMP}, \mathbb{PREF}, \mathbb{STAG}, \mathbb{SEMI}, \mathbb{STAB}\}$*, i.e.,* $\text{s-ipc}(\pi(n), \emptyset)$ *returns the proj. credulous count at the root* n *for resp. semantics* $\mathcal{S}$.

Proof (Idea). We can establish an invariant for each row of each table. Then, we show this invariant by simultaneous

Listing 3: Local algorithm $\mathbb{PROJ}(t, \cdot, \nu_t, (\cdot, \cdot, P), \langle \pi_1, \pi_2 \rangle)$ for projected counting, c.f., (Fichte et al. 2018).

In: Node t, table ν_t after purging, set P of projection atoms, $\langle \pi_1, \pi_2 \rangle$ is the sequence of tables at the children of t.
Out: Table π_t of pairs $\langle \sigma, c \rangle$, where $\sigma \subseteq \nu_t$, and $c \in \mathbb{N}$.
$\pi_t \leftarrow \{ \langle \sigma, \mathrm{ipc}(t, \sigma, \langle \pi_1, \pi_2 \rangle) \rangle \mid \emptyset \subsetneq \sigma \subseteq \mathrm{buckets}_P(\nu_t) \}$
return π_t

structural induction on pc and ipc starting at the leaf nodes and stepping until the root. This yields that the intersection projected count for the empty root corresponds to $\#\mathsf{PCred}_\mathcal{S}$ for the semantics $\mathcal{S}$. For completeness, we demonstrate by induction from root to leaves that a well-defined row of one table, which can indeed be obtained by the corresponding table algorithm, always has some preceding row in the respective child nodes. □

Runtime Bounds (Upper and Lower). In the following, we present upper bounds on algorithm $\mathbb{PROJ}$ that immediately result in runtime results for $\mathtt{PCNT}_\mathbb{S}$. Let therefore $\gamma(n)$ be the number of operations required to multiply two n-bit integers. Note that $\gamma(n) \in O(n \cdot \log(n) \cdot \log(\log(n)))$ (Knuth 1998). Note that in the following proposition m depends on the treewidth k. However, the actual order depends on the semantics.

Proposition 12 (⋆, Fichte and Hecher, 2018a)**.** $\mathtt{DP}_{\mathbb{PROJ}}$ *runs in time* $\mathcal{O}(2^{4m} \cdot g \cdot \gamma(\|F\|))$*, where* g *is the number of nodes of the given TD of the underlying graph* G_F *of the considered framework* F *and* $m := \max\{|\nu(t)| \mid t \in N\}$ *for input TTD* $\mathcal{T}_{purged} = (T, \chi, \nu)$ *of* $\mathtt{DP}_{\mathbb{PROJ}}$.

Corollary 13. *For* $\mathbb{S} \in \{\mathbb{ADM}, \mathbb{COMP}, \mathbb{STAB}\}$, $\mathtt{PCNT}_\mathbb{S}$ *runs in time* $\mathcal{O}(2^{2^{4k}} \cdot g \cdot \gamma(\|F\|))$*. For* $\mathbb{S} \in \{\mathbb{PREF}, \mathbb{SEMI}, \mathbb{STAG}\}$*, runs in time* $\mathcal{O}(2^{2^{2^{4k}}} \cdot g \cdot \gamma(\|F\|))$ *where* k *is the treewidth of the underlying graph* G_F *of the given AF* F.

Next, we take again the exponential time hypothesis (ETH) into account to establish lower bounds for counting projected extensions. In particular, we obtain that under reasonable assumptions, we cannot expect to improve the presented algorithms significantly.

Theorem 14 (⋆)**.** *Let* $\mathcal{S} \in \{$*admissible, complete, stable*$\}$*. Unless ETH fails, we cannot solve the problem* $\#\mathsf{PCred}_\mathcal{S}$ *in time* $2^{2^{o(k)}} \cdot \|F\|^{o(k)}$ *where* k *is the treewidth of the underlying graph* G_F *of the considered framework* F.

Proof (Sketch). We establish the lower bound by reducing an instance of ∀∃-SAT to an instance of a version of $\mathsf{Cred}_\mathcal{S}$ where the extension is of size exactly ℓ. Note that under ETH the problem ∀∃-SAT cannot be solved (Lampis and Mitsou 2017) in time $2^{2^{o(k)}} \cdot \|F\|^{o(k)}$ in the worst case. We follow the reduction from the proof of Statement 2 in Lemma 6. Let $\ell = |X|$, and observe that we can compute reduction in polynomial-time and the treewidth of the projected credulous counting instance is increased only linearly. It is easy to see that the reduction is correct since $|B(AF, X, t)| = \ell = |X|$ if and only if $\varphi(X) = \exists Y \psi(X, Y)$ holds for all assignments using X. Consequently, the claim follows. □

For semi-stable, preferred and stage semantics, we believe that this lower bound is not tight. Hence, we apply a stronger version (3ETH) of the ETH for quantified Boolean formulas (QBF). However, it is open whether also ETH implies 3ETH.

Hypothesis 15 (3ETH)**.** *The problem* ∃∀∃-SAT *for a QBF* Φ *of treewidth* k *can not be decided in time* $2^{2^{2^{o(k)}}} \cdot \|\Phi\|^{o(k)}$.

Using this hypothesis, we establish the following result.

Theorem 16 (⋆)**.** *Let* $\mathcal{S} \in \{$*preferred, semi-stable, stage semantics*$\}$*. Unless 3ETH fails, we cannot solve the problem* $\#\mathsf{PCred}_\mathcal{S}$ *in time* $2^{2^{2^{o(k)}}} \cdot \|F\|^{o(k)}$ *where* k *is the treewidth of the underlying graph of* F.

Proof (Idea). Assuming Hypothesis 15 we cannot solve an instance of ∀∃∀-SAT in time $2^{2^{2^{o(k)}}} \cdot \|F\|^{o(k)}$, otherwise we could solve an instance Φ of ∃∀∃-SAT, using a decision procedure for ∀∃∀-SAT with the inverse of Φ and inverting the result, in time $2^{2^{2^{o(k)}}} \cdot \|F\|^{o(k)}$. Towards the lower bound, we finally establish a reduction from ∀∃∀-SAT to projected credulous count exactly ℓ (c.f., Theorem 14). Thereby, we apply the reduction provided in Statement 1 of Lemma 6, set $\ell := |X|$ and proceed analogously to Theorem 14. □

Conclusion and Outlook

We established the classical complexity of counting problems in abstract argumentation and by present an algorithm that solves counting projected credulous extensions when exploiting treewidth in runtime double exponential in the treewidth or triple exponential in the treewidth depending on the considered semantics. Further, assuming ETH or a version for 3QBF, we establish that the runtime of the algorithms are asymptotically tight. While the upper bounds in Lemma 2 can be easily transferred to counting the number of extensions of a specific kind, the corresponding lower bounds cannot be immediately adopted from Lemma 3.

An open question is to investigate whether $\# \cdot \mathrm{coNP}$-hardness also applies for the preferred semantics. An interesting further research direction is to study whether we can obtain better runtime results by designing algorithms that take in addition also the number (small or large) of projection arguments into account or to study whether an implementation of our approach can benefit from massive parallelization (Fichte et al. 2018b). Furthermore, our technique might also be applicable to problems such as circumscription (Durand, Hermann, and Kolaitis 2005), default logic (Fichte, Hecher, and Schindler 2018), or QBFs (Charwat and Woltran 2016). Considering the (parameterized) enumeration complexity (Johnson, Papadimitriou, and Yannakakis 1988; Creignou et al. 2017; 2015) of the studied problems is also planned as future work. Finally, other measures such as fractional hypertree width might be interesting to consider (Fichte et al. 2018a).

References

Abseher, M.; Musliu, N.; and Woltran, S. 2017. htd – a free, open-source framework for (customized) tree decompositions and beyond. In *CPAIOR'17*.

Amgoud, L., and Prade, H. 2009. Using arguments for making and explaining decisions. *AIJ* 173(3-4):413–436.

Baroni, P., Dunne P. E., Giacomin M. 2010. COMMA'10.

Bliem, B.; Hecher, M.; and Woltran, S. 2016. On efficiently enumerating semi-stable extensions via dynamic programming on tree decompositions. In *COMMA'16*.

Bodlaender, H. L., and Kloks, T. 1996. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms* 21(2):358–402.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Bondy, J. A., and Murty, U. S. R. 2008. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer.

Hemaspaandra, L. A., and Vollmer, H. 1995. The satanic notations: Counting classes beyond #P and other definitional adventures. *SIGACT News* 26(1):2–13.

Charwat, G., and Woltran, S. 2016. Dynamic programming-based QBF solving. QBF'16@SAT.

Creignou, N.; Ktari, R.; Meier, A.; Müller, J.; Olive, F.; and Vollmer, H. 2015. Parameterized enumeration for modification problems. *LATA 2015*, 524–536.

Creignou, N.; Meier, A.; Müller, J.; Schmidt, J.; and Vollmer, H. 2017. Paradigms for parameterized enumeration. *TCS.* 60(4):737–758.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Dániel Marx, M. P.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AIJ* 77(2):321–357.

Dunne, P. E., and Bench-Capon, T. J. M. 2002. Coherence in finite argument systems. *AIJ* 141(1/2):187–203.

Dunne, P. E., and Bench-Capon, T. J. M. 2005. Argumentation in ai and law: Editors' introduction. *AI Law* 13(1):1–8.

Durand, A.; Hermann, M.; and Kolaitis, P. G. 2005. Subtractive reductions and complete problems for counting complexity classes. *TCS* 340(3):496–513.

Dvořák, W.; Morak, M.; Nopp, C.; and Woltran, S. 2013. dynpartix - a dynamic programming reasoner for abstract argumentation. In *INAP and WLP*, 259–268.

Dvořák, W. 2012. *Computational aspects of abstract argumentation*. Ph.D. Dissertation, TU Wien.

Dvořák, W., and Woltran, S. 2010. Complexity of semi-stable and stage semantics in argumentation frameworks. *IPL* 110(11):425 – 430.

Dvořák, W.; Pichler, R.; and Woltran, S. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *AIJ* 186:1–37.

Knuth, D. E. 1998. How fast can we multiply? In *The Art of Computer Programming*, volume 2 of *Seminumerical Algorithms*. Addison-Wesley. chapter 4.3.3, 294–318.

Fichte, J. K., and Hecher, M. 2018a. Exploiting Treewidth for Counting Projected Answer Sets. Extended abstract. *KR'18*. Extended workshop version available at NMR'18.

Fichte, J. K.; Hecher, M.; Lodha, N.; and Szeider, S. 2018a. An smt approach to fractional hypertree width. In *CP'18*.

Fichte, J. K.; Hecher, M.; and Meier, A. 2018. Counting Complexity for Reasoning in Abstract Argumentation. CoRR, abs/1811.11501.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2018. Exploiting treewidth for projected model counting and its limits. In *SAT'18*.

Fichte, J. K.; Hecher, M.; and Schindler, I. 2018. Default Logic and Bounded Treewidth. In *LATA'18*.

Fichte, J. K.; Hecher, M.; Woltran, S.; and Zisser, M. 2018b. Weighted Model Counting on the GPU by Exploiting Small Treewidth. *ESA'18*.

Fichte, J. K.; Lodha, N.; and Szeider, S. 2017. SAT-based local improvement for finding tree decompositions of small width. In *SAT'17*.

Graham, R. L.; Grötschel, M.; and Lovász, L. 1995. *Handbook of combinatorics*, volume I. Elsevier.

Jakl, M.; Pichler, R.; and Woltran, S. 2009. Answer-set programming with bounded treewidth. In *IJCAI'09*.

Johnson, D. S.; Papadimitriou, C. H.; and Yannakakis, M. 1988. On generating all maximal independent sets. *IPL* 27(3):119–123.

Kloks, T. 1994. *Treewidth. Computations and Approximations*. Springer.

Lampis, M., and Mitsou, V. 2017. Treewidth with a quantifier alternation revisited. *IPEC'17*.

Lampis, M.; Mengel, S.; and Mitsou, V. 2018. QBF as an Alternative to Courcelle's Theorem. In *SAT'18*.

Maher, M. J. 2016. Resistance to corruption of strategic argumentation. In *AAAI'16*.

McBurney, P.; Parsons, S.; and Rahwan, I., eds. 2011. *Proceedings of ArgMAS'11*.

McCarthy, J. 1980. Circumscription - A form of non-monotonic reasoning. *AIJ* 13(1-2):27–39.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison Wesley.

Pichler, R.; Rümmele, S.; and Woltran, S. 2010. Counting and enumeration problems with bounded treewidth. In *LPAR'10*.

Rago, A.; Cocarascu, O.; and Toni, F. 2018. Argumentation-based recommendations: Fantastic explanations and how to find them. In *IJCAI'18*.

Rahwan, I. 2007. *Argumentation in Artificial Intelligence*, volume 171. Elsevier Science Publishers, North-Holland.

Toda, S., and Watanabe, O. 1992. Polynomial-time 1-Turing reductions from #PH to #P *TCS*, 100:205–221, Elsevier.

Valiant, L. 1979. The complexity of computing the permanent. *TCS*, 8:189–201, Elsevier.

Wilder, R. L. 1965. *Introduction to the Foundations of Mathematics*. John Wiley & Sons, 2nd edition.