Journal of Artificial Intelligence Research 80 (2024) 805–834 Submitted 03/2024; published 06/2024

Counting Complexity for Reasoning in Abstract Argumentation

Johannes K. Fichte JOHANNES.FICHTE@LIU.SE
Linköping University
58183 Linköping, Sweden

Markus Hecher HECHER@MIT.EDU
MIT Computer Science & Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar St
Cambridge, MA 02139, USA

Arne Meier MEIER@THI.UNI-HANNOVER.DE
Leibniz Universität Hannover
Institut für Theoretische Informatik
Appelstraße 4
30167 Hannover, Germany

Abstract

In this paper, we consider counting and projected model counting of extensions in abstract argumentation for various semantics, including credulous reasoning. 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 the 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. Our algorithms run in double or triple exponential time in the treewidth, depending on the semantics under consideration. Finally, we establish lower bounds of bounded treewidth algorithms for counting extensions and projected extension under the exponential time hypothesis (ETH).

1. Introduction

Abstract argumentation (Dung, 1995; Rahwan, 2007) is a central framework for modeling and evaluating arguments and their reasoning with applications to various areas in artificial intelligence (AI) (Amgoud & Prade, 2009; Rago, Cocarascu, & 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 sets of 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 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 that contain certain arguments. This even opens the door to computing conditional probabilities with respect to quantities of extensions for a given argument, which enables us to understand solutions and solution spaces. Counting problems related to credulous reasoning can emphasize a quantitative view of this type of reasoning. By this, the conditional probability can be expressed. In addition, counting allows for more fine-grained reasoning between the usually two extreme cases of skeptical and credulous reasoning, c.f. (Fichte, Hecher, & Nadeem, 2022; Fichte, Hecher, Mahmood, & Meier, 2023). If we restrict our solutions to a certain subset of the arguments, we ask for the number of projected extensions, which is sometimes also called forgetting. Projected counting is a most natural extension of counting, since we are often interested in the count with respect to a subset of the arguments where the auxiliary arguments are not in one-to-one correspondence with other extensions. In the end, the proposed advancements allow for developing probabilistic variants of classical reasoning problems (e.g., from "'is a given argument contained in an extension?" to "is a given argument contained in an extension with probability P?").

Interestingly, the computational complexity of the decision problem is already quite hard. More precisely, the problem of credulous acceptance is NP-complete for the stable semantics and even Σ_2^p-complete for the semi-stable semantics (Dunne & Bench-Capon, 2002; Dvořák & 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 (Downey & Fellows, 1999, 2013; Cygan, Fomin, Kowalik, Lokshtanov, Marx, Pilipczuk, Pilipczuk, & Saurabh, 2015). Usually, one aims 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. In this paper, we consider a graph-theoretic measure of the undirected graph of the given argumentation framework. As a measure, we take treewidth, which is arguably the most prominent graph invariant in combinatorics of graph theory, and makes various graph problems easier when the input graph is of bounded treewidth.

The exponential time hypothesis (ETH) (Impagliazzo, Paturi, & Zane, 2001) states that there is some real $s > 0$ such that we cannot decide satisfiability of a given 3-CNF formula φ in time $2^{s \cdot |\varphi|} \cdot \|\varphi\|^{\mathcal{O}(1)}$. It is a widely believed hypothesis and has been used to establish lower bounds for various problems, also in classical complexity theory.

Contributions. Our main contributions are as follows.

1. We establish the classical complexity of counting extensions and counting projected extensions for various semantics in abstract argumentation. Furthermore, we exhibit the counting complexity of (projected) credulous reasoning for the first time. By this, we fill a gap in the literature and provide a comprehensive overview of the counting complexity of reasoning in abstract argumentation. Table 1 shows an overview of the counting complexity results.

semantics $\mathcal{S}$	$\#\mathsf{Ext}_{\mathcal{S}}$	$\#\mathsf{PExt}_{\mathcal{S}}$	$\#\mathsf{Cred}_{\mathcal{S}}$	$\#\mathsf{PCred}_{\mathcal{S}}$
admissible	$\# \cdot \mathrm{P}$	$\# \cdot \mathrm{NP}$	$\# \cdot \mathrm{P}$	$\# \cdot \mathrm{NP}$
complete	$\# \cdot \mathrm{P}$	$\# \cdot \mathrm{NP}$	$\# \cdot \mathrm{P}$	$\# \cdot \mathrm{NP}$
stable	$\# \cdot \mathrm{P}$	$\# \cdot \mathrm{NP}$	$\# \cdot \mathrm{P}$	$\# \cdot \mathrm{NP}$
preferred	$\in \# \cdot \mathrm{coNP}$	$\# \cdot \Sigma_2^{\mathrm{P}}$	$\in \# \cdot \mathrm{coNP}$	$\# \cdot \Sigma_2^{\mathrm{P}}$
semi-stable	$\# \cdot \mathrm{coNP}$	$\# \cdot \Sigma_2^{\mathrm{P}}$	$\# \cdot \mathrm{coNP}$	$\# \cdot \Sigma_2^{\mathrm{P}}$
stage	$\# \cdot \mathrm{coNP}$	$\# \cdot \Sigma_2^{\mathrm{P}}$	$\# \cdot \mathrm{coNP}$	$\# \cdot \Sigma_2^{\mathrm{P}}$
	Corollary 8	Corollary 12	Theorem 7	Theorem 11

Table 1: Overview of counting complexity results. The $\# \cdot \mathrm{coNP}$ results are completeness results (except only membership for $\#\mathsf{Cred}_{\mathrm{preferred}}$) with respect to subtractive reductions, the remainder is complete under parsimonious reductions.

semantics	$\#\mathsf{Cred}_{\mathcal{S}}$	$\#\mathsf{PCred}_{\mathcal{S}}$
admissible	$2^{\Theta(k)} \cdot \mathrm{poly}(\lvert F\rvert)^*$	$2^{2^{\Theta(k)}} \cdot \mathrm{poly}(\lvert F\rvert)$
complete	$2^{\Theta(k)} \cdot \mathrm{poly}(\lvert F\rvert)^*$	$2^{2^{\Theta(k)}} \cdot \mathrm{poly}(\lvert F\rvert)$
stable	$2^{\Theta(k)} \cdot \mathrm{poly}(\lvert F\rvert)^*$	$2^{2^{\Theta(k)}} \cdot \mathrm{poly}(\lvert F\rvert)$
preferred	$2^{2^{\Theta(k)}} \cdot \mathrm{poly}(\lvert F\rvert)^*$	$2^{2^{2^{\Theta(k)}}} \cdot \mathrm{poly}(\lvert F\rvert)$
semi-stable	$2^{2^{\Theta(k)}} \cdot \mathrm{poly}(\lvert F\rvert)^*$	$2^{2^{2^{\Theta(k)}}} \cdot \mathrm{poly}(\lvert F\rvert)$
stage	$2^{2^{\Theta(k)}} \cdot \mathrm{poly}(\lvert F\rvert)$	$2^{2^{2^{\Theta(k)}}} \cdot \mathrm{poly}(\lvert F\rvert)$
	Theorem 17	Theorem 21/23

Table 2: Summary of results for upper and lower bounds (under ETH) for projected credulous counting. *: Previously known upper bound.

2. We present an algorithm that solves the problem of counting projected extensions by exploiting the treewidth in runtime, either double exponential in the treewidth or triple exponential in the treewidth, depending on the semantics under consideration. This is the first algorithm that solves counting projected extensions for argumentation in time double or triple exponential in the treewidth. From a more practical point of view, if the abstract argumentation frameworks of interest are of small treewidth, then our algorithms efficiently solve hard counting problems.

3. Assuming ETH, we show that one *cannot* count projected extensions single exponential in the treewidth. This shows that our algorithms are tight under the ETH. These results are summarized in Table 2.

Broader Relation to Argumentation and AI. Counting allows for more fine-grained reasoning between the two extremes of skeptical and gullible reasoning by enabling quantitative and probabilistic reasoning. More precise cases of reasoning have also been investigated in argumentation. For example, Konieczny, Marquis, and Vesic (2015a) introduced

inference policies relying on greatest supports among extensions. There, extensions are compared based on number of arguments in one extension not attacked by another extension. In addition, counting is a key ingredient for handling probabilistic reasoning more effectively than by enumeration alone (Fichte, Hecher, & Hamiti, 2021). By counting twice, taking the number of extensions under an assumption and taking the number of all, we obtain conditional probability. In this way, we can also move reasoning from a given argument contained in an extension to arguments in an extension with a certain probability. Imposing certain constraints on the joint probability distribution of the argument sets can also be used to define advanced probabilistic semantics (Käfer, Baier, Diller, Dubslaff, Gaggl, & Hermanns, 2022). Counting can also be helpful for understanding cases where a large number of extensions arise (Dachselt, Gaggl, Krötzsch, Méndez, Rusovac, & Yang, 2022; Fichte, Gaggl, & Rusovac, 2022b) or when divergent solutions are sought (Böhl, Gaggl, & Rusovac, 2023). Interestingly, weighted argument systems have been introduced, where attacks are assigned a weight indicating the relative strength of the attack (Dunne, Hunter, McBurney, Parsons, & Wooldridge, 2011). Then, computing extensions turns into optimization, which slightly increases the computational complexity. Such extensions could also be considered by extending our results to weighted counting similar to sum-of-products or weighted model counting (Fichte et al., 2021) or probabilistic logic programming (De Raedt & Kimmig, 2015). Finally, there exist recent work that studies quantitative aspects of claim-centric reasoning in logic-based argumentation (Hecher, Mahmood, Meier, & Schmidt, 2024).

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 problems of creadulous reasoning as well as to further argumentation semantics. Lampis, Mengel, and Mitsou (2018) considered bounded treewidth algorithms and established lower bounds on the runtime of an algorithm that solves reasoning in abstract argumentation under the admissible and preferred semantics. These results do not extend trivially to counting and are based on reductions to QBF. They give asymptotically tight bounds, but still yield a constant factor. Unfortunately, even a small increase of one can add an 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, Hecher, Morak, Thier, and Woltran (2023) gave dynamic programming 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 dynamic programming algorithms that exploit treewidth to solve decision problems of various semantics in abstract argumentation. We employ these results and lift them to projected counting. Dynamic programming algorithms for projected counting in answer set programming (ASP) are known (Fichte & Hecher, 2018), but existing reductions from argumentation to ASP are not treewidth preserving. The results in this paper could already be fruitfully used in the context of quantitative reasoning in abstract argumentation quite recently (Fichte et al., 2023). The abstract argumentation track of the 4th International Competition on Computational Models of Argumentation (ICCMA) had as sub-track extension counting as a reasoning problem (CE) (Lagniez, Lonca, Mailly, & Rossit, 2020, 2021). Projected counting is a crucial reasoning task and of central interest in reasoning (Darwiche & Marquis, 2002; Aziz, Chu, Muise, & Stuckey, 2015; Fichte

et al., 2021, 2022b; Audemard, Lagniez, & Miceli, 2022; Yang, Chakraborty, & Meel, 2022; Fichte et al., 2023; Vigouroux, Bozga, Ene, & Mounier, 2024). In fact, today most solvers that participate in the Model Counting Competition also support projected model counting (Hecher & Fichte, 2023). There exists plenty applications on projected model-counting such as software reliability (Dueñas-Osorio, Meel, Paredes, & Vardi, 2017) and reliability of power grids (Dueñas-Osorio et al., 2017). Recent results considered counting in epistemic logic programs where one can reason inside a program about all or some solutions (Eiter, Fichte, Hecher, & Woltran, 2024).

Prior Work and Differences. A preliminary version of this article entitled "Counting complexity for reasoning in abstract argumentation" (Fichte, Hecher, & Meier, 2019) has been published in the proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019. The present version clarifies all implications on extension counting, projected counting, and (projected) counting on credulous extensions (Corollaries 8 and 12), while the preliminary version focused on credulous counting only. Our results are more comprehensive, only one hardness case remains open, in particular we include results for the preferred cases in our classifications. Results in Theorem 23 all hold directly under ETH, now. In addition, we provide all proof details, more examples, and a more thorough outlook.

2. Formal Background

We assume graphs to be undirected and use digraphs for directed graphs with their usually definitions (Bondy & Murty, 2008). Furthermore, we 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, & Lovász, 1995).

2.1 Counting Complexity

We follow standard terminology in this area (Toda & Watanabe, 1992; Hemaspaandra & Vollmer, 1995; Durand, Hermann, & 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 (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

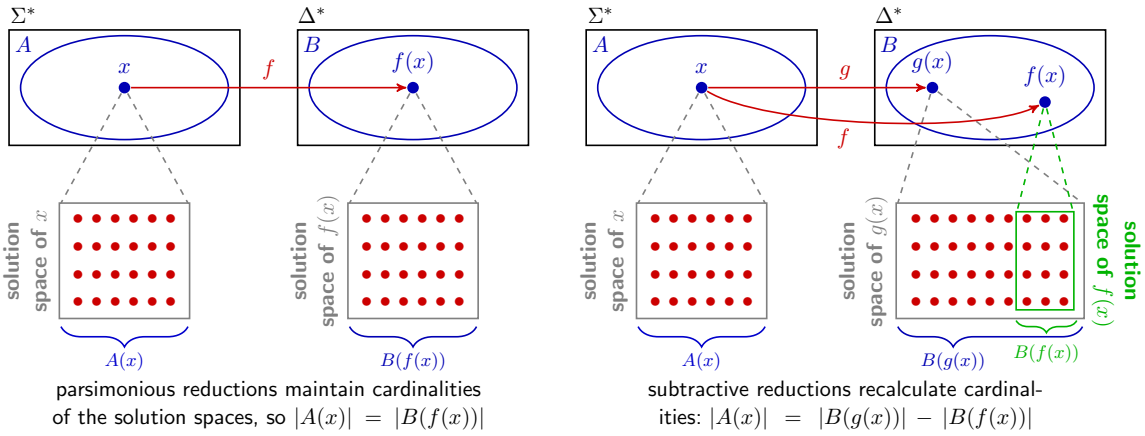

Figure 1: Difference between parsimonious and subtractive reductions.

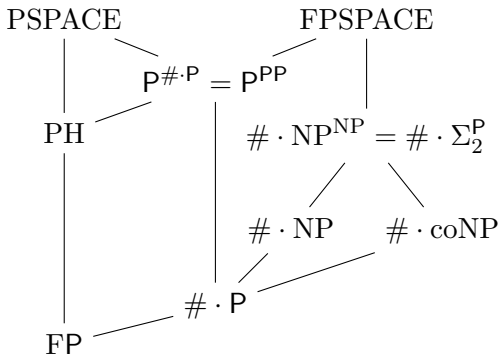

Figure 2: Counting complexity landscape with relevant classes in this paper.

witness functions. See Figure 1 for some intuition on the difference of the two considered reduction types. Figure 2 shows the relevant counting complexity classes in our settings, and relates them to classical decision complexity classes.

2.2 Abstract Argumentation

We make use of the formal argumentation theory developed by Dung (1995). An *argumentation framework (AF)*, or framework for short, is a digraph $F = (A, R)$ where A is a non-empty and finite set of arguments, and $R \subseteq A \times A$ is a set of pairs of arguments representing a direct attack relationship between arguments. More formally, given $S, S' \subseteq A$, we define $S \rightarrowtail_R S'$, which denotes $\{s \in S \mid (\{s\} \times S') \cap R \neq \emptyset\}$, and $S \leftarrowtail_R S' \coloneqq \{s \in S \mid (S' \times \{s\}) \cap R \neq \emptyset\}$. 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 $\mathrm{def}_F(S) \coloneqq \{\, s \mid s \in A, s \text{ is defended by } S \text{ in } F \,\}$. Now, we turn towards the definitions of the semantics. We say $S \subseteq A$ is

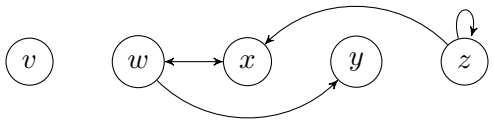

Figure 3: Argumentation framework F_{Ex}.

(i) *conflict-free in* S if $(S \times S) \cap R = \emptyset$;

(ii) *admissible in* F if S is *conflict-free in* F, and every $s \in S$ is *defended by* S *in* F.

(iii) 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 $S_R^+ \coloneqq S \cup \{\, a \mid (b,a) \in R, b \in S \,\}$;

  (iiid) S is *stable in* F if every $s \in A \setminus S$ is *attacked* by some $s' \in S$.

(iv) 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 stable$(F) \subseteq$ semi-stable$(F) \subseteq$ preferred$(F) \subseteq$ complete$(F) \subseteq$ admissible$(F) \subseteq$ conflict-free(F) and stable$(F) \subseteq$ stage$(F) \subseteq$ conflict-free(F).

Example 1. *Figure 3 illustrates an AF where* $F_1 = (A_1, R_1)$ *with* $A_1 = \{v, w, x, y, z\}$ *and* $R_1 = \{(w,x), (x,w), (w,y), (z,z), (z,x)\}$*, c.f. (Bliem, Hecher, & Woltran, 2016, Ex. 2.7). Observe that* $\emptyset \in$ conflict-free(F_1)*. For every* $a \in A_1$ *such that* $a \neq z$ *it holds that* $\{a\} \in$ conflict-free(F_1)*; since* v *is isolated, also* $\{v, a\} \in$ conflict-free(F_1) *for every* $a \in A_{F_1}$ *with* $a \neq z$*. Argument* z *is not contained in any* $S \in$ conflict-free(F_1)*, since it attacks itself. Finally,* conflict-free$(F_1) = \{\emptyset, \{v\}, \{w\}, \{x\}, \{y\}, \{v,w\}, \{v,x\}, \{v,y\}, \{x,y\}, \{v,x,y\}\}$*. Argument* x *can never be part of any admissible extension as* z *has a self-loop. We have that* admissible$(F_1) = \{\emptyset, \{v\}, \{w\}, \{v,w\}\}$*. The set* $\emptyset$ *is not complete since* $\mathrm{def}_{F_1}(\emptyset) = \{v\}$*;* $\{w\} \notin$ complete$(\{w\})$*, since* $\mathrm{def}_{F_1}(\{w\}) = \{v,w\}$*. In the end,* complete$(F_1) = \{\{v\}, \{v,w\}\}$*. Observe that* preferred$(F_1) =$ semi-stable$(F_1) =$ stage$(F_1) = \{\{v,w\}\}$*. Finally, since* z *is not contained in any extension* $S \in$ conflict-free(F_1) *and it is not attacked by any* $a \in S$ *(*z *only attacks itself), there cannot be any stable extension.*

2.3 Problems of Interest

In argumentation, one is usually interested extension existence as well as 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 semantics, $F = (A, R)$ be an argumentation framework, and $a \in A$ be an argument. The three central decision problems are as follows.

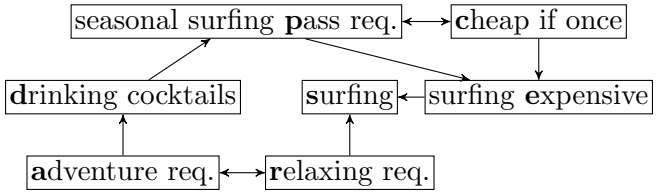

Figure 4: Argumentation framework F: surfing vs. cocktails. See an explanation in Ex. 2.

- The *extension existence problem*, $\mathsf{Ext}_{\mathcal{S}}$, asks whether a F has an extension according to $\mathcal{S}$.
- The *credulous reasoning problem* $\mathsf{Cred}_{\mathcal{S}}$ asks to decide whether there is an $\mathcal{S}$-extension of F that contains the given argument a.
- The *skeptical reasoning problem* $\mathsf{Skep}_{\mathcal{S}}$ asks to decide whether all $\mathcal{S}$-extensions of F contain a given argument a.

Now, we turn towards counting versions. Here, the following are of interest.

- The *extenstion counting problem* $\#\mathsf{Ext}_{\mathcal{S}}$ asks to output the number of $\mathcal{S}$-extensions.
- The *credulous counting problem* $\#\mathsf{Cred}_{\mathcal{S}}$ asks to output the number of $\mathcal{S}$-extensions of F that contain given argument a, i.e., $|\{\, S \mid S \in \mathcal{S}(F), a \in S \,\}|$.

We are also interested in projections in the counting environment.

- The *projected extension counting problem* $\#\mathsf{PExt}_{\mathcal{S}}$ asks to output the number of $\mathcal{S}$-extensions restricted to the projection arguments P, i.e., $|\{\, S \cap P \mid S \in \mathcal{S}(F) \,\}|$.
- 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 2. *Consider the framework F from Figure 4, which depicts a framework for deciding between surfing and drinking cocktails. The narrative here is, for example, if you need to relax, you will not go surfing; or if you want adventure, you will probably not drink cocktails. Framework F admits three stable 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*. Intuitively, the projection here serves as a filter for counting the extensions that contain arguments from P. By this, one has more flexibility in emphasizing certain arguments in the counting process.*

Refering back to the example, Konieczny et al. (Konieczny, Marquis, & Vesic, 2015b) have used the number of times an argument is contained in an extension to determine the strength of that argument, and then compared such extensions based on the quality of the arguments contained.

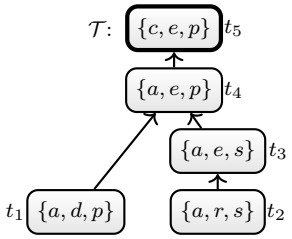

Figure 5: A Tree decomposition of F.

2.4 Tree Decompositions (TDs)

For a tree $T = (N, E_T, n)$ with root n and a node $t \in N$, we let children(t, T) be the sequence of all nodes t' in arbitrarily but fixed order, which have an edge $(t, t') \in A$.

Definition 3 (Robertson and Seymour, (1986))**.** *Let $G = (V, E)$ be an undirected graph. A* tree decomposition (TD) *of graph G is a pair $\mathcal{T} = (T, \chi)$, where $T = (N, E_T, n)$ 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, width$(\mathcal{T}) := \max_{t \in N} |\chi(t)| - 1$. *The* treewidth *$tw(G)$ of G is the minimum* 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, & Woltran, 2017) to compute a tree decomposition are often sufficient. For the case of unfixed w the problem is NP-complete (Arnborg, Corneil, & Proskurowski, 1987). 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 type(t) is *leaf* if children$(t, T) = \langle\rangle$; *join* if children$(t, T) = \langle t', t'' \rangle$ where $\chi(t) = \chi(t') = \chi(t'') \neq \emptyset$; *int* ("introduce") if children$(t, T) = \langle t' \rangle$, $\chi(t') \subseteq \chi(t)$ and $|\chi(t)| = |\chi(t')| + 1$; *rem* ("remove") if children$(t, T) = \langle t' \rangle$, $\chi(t') \supseteq \chi(t)$ and $|\chi(t')| = |\chi(t)| + 1$. If for every node $t \in N$, type$(t) \in \{\mathit{leaf}, \mathit{join}, \mathit{int}, \mathit{rem}\}$ and bags of leaf nodes and the root are empty, then the TD is called *nice*.

Example 4. *Figure 5 shows a tree decomposition $\mathcal{T}$ of F of Figure 4. Observe that the width of $\mathcal{T}$ is optimal, i.e., there is no TD of F of width 2. Intuitively, $\mathcal{T}$ allows us to evaluate F in parts, where the overall computation of extensions of F is guided by the TD.*

2.5 Logic

We use basic concepts in propositional logic (Ebbinghaus, Flum, & Thomas, 1994). We denote a negated variable x often by $\bar{x}$. A *literal* is either a variable or a negated variable. A *clause* is a disjunction of literals. A formula is said to be in *conjunctive normal form*

(CNF) if it is a conjunction of clauses. Symmetrically, a formula is in *disjunctive normal form (DNF)* if it is a disjunction of conjunctions of literals.

3. Classical Counting Complexity

In this section, we study the classical counting complexity of the credulous reasoning problem.

Lemma 5. $\#\mathsf{Cred}_{\mathcal{S}}$ *is in*

1. $\# \cdot \mathrm{P}$ *if* $\mathcal{S}$ *is conflict-free, stable, admissible, or complete.*
2. $\# \cdot \mathrm{coNP}$ *if* $\mathcal{S}$ *is preferred, semi-stable, or stage.*

Proof. The nondeterministic machine first guesses a candidate extension set S and then verify whether it is an extension of the desired semantics plus if the given argument is contained in it. The number of computation paths then corresponds one-to-one to the possible extensions.

1. Being conflict-free can be checked in P. Coste-Marquis, Devred, and Marquis (2005) show that the verification process of extensions for the semantics admissible, stable, and complete can be done in deterministic polynomial time.
2. For semi-stable, resp., stage extensions, we need to ensure that there exists no set $S' \subseteq A$ whose range is a superset of the range of the extension candidate. This property can be verified with a coNP oracle. Similarly, Dunne and Bench-Capon (2002) claim that verifying if a given extension is preferred is coNP-complete. □

Note that the next lemma considers neither conflict-free nor preferred extensions.

Lemma 6. $\#\mathsf{Cred}_{\mathcal{S}}$ *is*

1. $\# \cdot \mathrm{P}$*-hard under parsimonious reductions if* $\mathcal{S}$ *is stable, admissible, or complete.*
2. $\# \cdot \mathrm{coNP}$*-hard under subtractive reductions if* $\mathcal{S}$ *is semi-stable, or stage.*

Proof. 1. Start with the case of stable or complete extensions. Adopting ideas of Dunne and Bench-Capon (2002), we construction a parsimonious reduction from $\#\mathsf{SAT}$. Given a propositional formula $\varphi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i$ with clauses C_i, define an AF $F_\varphi = (A, R)$ where

$$\begin{aligned} A &= \{\, x_i, \bar{x}_i \mid 1 \leqslant i \leqslant n \,\} \cup \{\, C_i \mid 1 \leqslant i \leqslant m \,\} \cup \{t, \bar{t}\}, \\ R &= \{\, (x_i, \bar{x}_i), (\bar{x}_i, x_i) \mid 1 \leqslant i \leqslant n \,\} \\ &\quad \cup \{\, (x_i, C_j) \mid x_i \in C_j \,\} \cup \{\, (\bar{x}_i, C_j) \mid \bar{x}_i \in C_j \,\} \\ &\quad \cup \{\, (C_i, t) \mid 1 \leqslant i \leqslant m \,\} \cup \{(t, \bar{t}), (\bar{t}, t)\}. \end{aligned}$$

Then, due to the range maximality, the number of satisfying assignments of φ coincides with the number of stable (complete) extensions of F_φ which contain the argument t.

For the case of admissible extensions, to count correctly, it is crucial that for each variable 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 $\bar{x}_i$. As a result, for each admissible extension S, we have that $|S \cap \{x_i, \bar{x}_i\}| = 1$ for each $1 \leqslant i \leqslant n$. Notice that the arguments s_i are not part of any admissible extension. Because of this, these arguments have no influence on the overall count of admissible extensions. The modified framework for this case then is $F'_\varphi = (A', R')$, where

$$A' = A \cup \{\, s_i \mid 1 \leqslant i \leqslant n \,\},$$
$$R' = R \cup \{\, (s_i, t), (x_i, s_i), (\bar{x}_i, s_i) \mid 1 \leqslant i \leqslant n \,\}.$$

2. We state a parsimonious reduction from counting minimal models of CNFs to the $\#\mathsf{Cred}_{\mathcal{S}}$ problem. 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 *pointwise partial order* on 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 et al. (2005) showed that $\#\mathsf{Circumscription}$ is $\# \cdot \mathrm{coNP}$-complete via subtractive reductions (that is why the hardness in our result is only under this reduction type). Given a Boolean formula $\varphi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i$ with C_i are disjunctions of literals, we will construct an argumentation framework $F_\varphi = (A, R)$ as follows:

$$A = \{\, x_i, \bar{x}_i, b_i \mid 1 \leqslant i \leqslant n \,\} \cup \{\, C_i \mid 1 \leqslant i \leqslant m \,\} \cup \{t\},$$
$$R = \{\, (b_i, b_i), (\bar{x}_i, b_i), (x_i, \bar{x}_i), (\bar{x}_i, x_i) \mid 1 \leqslant i \leqslant n \,\}$$
$$\cup \{\, (x_i, C_j) \mid x_i \in C_j \,\} \cup \{\, (\bar{x}_i, C_j) \mid \bar{x}_i \in C_j \,\}$$
$$\cup \{\, (C_i, t) \mid 1 \leqslant i \leqslant m \,\}.$$

The crux is that choosing negative literals is more valuable than selecting positive ones. This is true as each negative literal additionally attack a corresponding b_i and thereby increases the range (more than the positive literal could). Consequently, this construction models subset minimal models. Finally, one merely needs to select models where t is in a range-maximal semi-stable, resp., stage extension. □

Lemma 5 and 6 together show the following theorem.

Theorem 7. *$\#\mathsf{Cred}_{\mathcal{S}}$ 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$\}$.*

While for the case of $\#\mathsf{Cred}_{\text{preferred}}$ membership in $\# \cdot \mathrm{coNP}$ is clear (by Lemma 5) the lower bound is still open. With the previous construction it is not clear how to utilize subset-maximization.

Yet, the complexity of counting extensions is a direct consequence of the previous theorem.

Corollary 8. *$\#\mathsf{Ext}_{\mathcal{S}}$ 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 9. *$\#\mathsf{PCred}_{\mathcal{S}}$ is in*

1. *$\# \cdot \mathrm{NP}$ if $\mathcal{S}$ is stable, admissible, or complete.*
2. *$\# \cdot \Sigma_2^{\mathrm{P}}$ if $\mathcal{S}$ is semi-stable, preferred, or stage.*

Proof. Given an argumentation framework AF, a projection set P, and an argument a. Nondeterministically branch on a possible projected extension S. Accordingly, we have that $S \subseteq P$. If $a \in S$ and S is of the respective semantics, then accept. Otherwise, make the one allowed nondeterministic oracle guess $S' \supseteq S$, verify if $P \cap S' = S$, $a \in S'$, and S' is of the desired semantics. As explained in the proof of Lemma 5, extension verification is (1.) in P for stable, admissible, or complete, and (2.) in coNP for semi-stable, preferred, or stage. 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. This yields either $\# \cdot \mathrm{NP}$ or $\# \cdot \Sigma_2^{\mathrm{P}}$ as upper bounds. □

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 et al. (2005) have shown that the problem is $\# \cdot \Sigma_k^{\mathrm{P}}$-complete via parsimonious reductions.

Lemma 10. *$\#\mathsf{PCred}_{\mathcal{S}}$ is*

1. *$\# \cdot \Sigma_2^{\mathrm{P}}$-hard w.r.t. parsimonious reductions if $\mathcal{S}$ is stage, preferred, or semi-stable.*
2. *$\# \cdot \mathrm{NP}$-hard w.r.t. parsimonious reductions if $\mathcal{S}$ is admissible, stable, or complete.*

Proof. 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

$$\begin{aligned}
A &= \{\, x, \bar{x} \mid x \in X \,\} \cup \{\, y, \bar{y}, y', \bar{y}' \mid y \in Y \,\} \\
&\quad \cup \{\, z, \bar{z} \mid z \in Z \,\} \cup \{t, \bar{t}, b\} \\
R &= \{\, (y', y'), (\bar{y}', \bar{y}'), (y, y'), (\bar{y}, \bar{y}'), (y, \bar{y}), (\bar{y}, y) \mid y \in Y \,\} \\
&\quad \cup \{(b, b), (t, \bar{t}), (\bar{t}, t), (t, b)\} \\
&\quad \cup \{\, (C_i, t) \mid 1 \leqslant i \leqslant m \,\} \\
&\quad \cup \{\, (u, C_i) \mid u \in X \cup Y \cup Z, u \in C_i, 1 \leqslant i \leqslant m \,\} \\
&\quad \cup \{\, (\bar{u}, C_i) \mid u \in X \cup Y \cup Z, \bar{u} \in C_i, 1 \leqslant i \leqslant m \,\}
\end{aligned}$$

Note that, by construction, the $y', \bar{y}'$ variables make the extensions w.r.t. 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. (This is crucial in our case, as stage as well as semi-stable strive for range maximal extensions.)

Notice that, each time, when there is a possible solution to $\psi'(X, Y, Z)$, semantically $\neg\psi(X, Y, Z)$ w.r.t. the free X-variables is to be considered. Accordingly, if for every assignment over the Y-variables there exists an assignment to the Z-variables, then the extension will contain t. As a result, the extensions containing t correspond to the dissatisfying assignments. So, this achieves a one-to-one correspondence between the number of dissatisfying assignments of $\neg\varphi(X)$ and the number of extensions containing t. In turn, this means there is a one-to-one correspondence between the number of satisfying assignments of $\varphi(X)$ and the number of extensions not containing t (yielding $\bar{t}$ in the extension).

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/preferred extensions which contain a and are projected to P. Then, the previous explanations establish $|A(\varphi(X))| = |B(AF, X, \bar{t})|$ proving the desired parsimonious 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 $\#\mathsf{PCred}_{\mathcal{S}}$. Consider a formula $\varphi(X) = \exists Y\ \psi(X, Y)$, where X, Y are sets of variables, $\psi = \bigwedge_{i=1}^{m} C_i$ and C_i is a disjunction of literals for $1 \leqslant i \leqslant m$. Essentially the reduction is the same, however we need the same extension as in the proof of Lemma 6 and we neither need the $y', \bar{y}'$ nor—of course—the z variables. Define the framework $AF = (A, R)$ as follows:

$$
\begin{aligned}
A &= \{\, x, \bar{x} \mid x \in X \,\} \cup \{\, y, \bar{y} \mid y \in Y \,\} \\
&\quad \cup \{t, \bar{t}, b\} \cup \{\, s_x, s_y \mid x \in X, y \in Y \,\} \\
R &= \{(b, b), (t, \bar{t}), (\bar{t}, t), (t, b)\} \\
&\quad \cup \{\, (C_i, t) \mid 1 \leqslant i \leqslant m \,\} \\
&\quad \cup \{\, (s_x, t), (s_y, t) \mid x \in X, y \in Y \,\} \\
&\quad \cup \{\, (x, s_x), (\bar{x}, s_x), (y, s_y), (\bar{y}, s_y) \mid x \in X, y \in Y \,\} \\
&\quad \cup \{\, (u, C_i) \mid u \in X \cup Y, u \in C_i, 1 \leqslant i \leqslant m \,\} \\
&\quad \cup \{\, (\bar{u}, C_i) \mid u \in X \cup Y, \bar{u} \in C_i, 1 \leqslant i \leqslant m \,\}
\end{aligned}
$$

This time, let $A(\varphi(X))$ denote the set of satisfying assignments of an $\Sigma_1\mathsf{SAT}$ instance. Similarly as for (1.), we achieve the desired one-to-one correspondence as before. Then, define $B(AF, P, a)$ be the set of admissible/stable/complete extensions which contain a and are projected to P. Finally, the explanation from above achieve $|A(\varphi(X))| = |B(AF, X, t)|$ showing the claimed reduction and $\# \cdot \mathrm{NP}$-hardness via parsimonious reductions. □

Theorem 11. $\#\mathsf{PCred}_{\mathcal{S}}$ *is* $\# \cdot \mathrm{NP}$*-complete via parsimonious reductions if* $\mathcal{S} \in \{$*stable, admissible, complete*$\}$*, and* $\# \cdot \Sigma_2^{\mathrm{P}}$*-complete via parsimonious reductions if* $\mathcal{S} \in \{$*stage, preferred, semi-stable*$\}$*.*

As before, the complexity of the projection variant for counting extensions is a direct consequence of the previous theorem.

Corollary 12. $\#\mathsf{PExt}_{\mathcal{S}}$ *is* $\# \cdot \mathrm{NP}$*-complete via parsimonious reductions if* $\mathcal{S} \in \{$*stable, admissible, complete*$\}$*, and* $\# \cdot \Sigma_2^{\mathrm{P}}$*-complete via parsimonious reductions if* $\mathcal{S} \in \{$*stage, preferred, semi-stable*$\}$*.*

Similarly, one can introduce problems of the form $\#\mathsf{Skep}_{\mathcal{S}}$ and $\#\mathsf{PSkep}_{\mathcal{S}}$, which correspond to the counting versions of the skeptical reasoning problem. Since skeptical is dual to credulous reasoning, one easily obtains completeness results for the dual counting classes. Formally, however, functions for the counting problems $\#\mathsf{Skep}_{\mathcal{S}}$ and $\#\mathsf{PSkep}_{\mathcal{S}}$ can only take the values 0 (in the negative case) or the number of (projected) extensions (when all extensions contain the given argument). As a result, this is a rather limited problem, since it is closely related to extension counting.

4. Dynamic Programming for Abstract Argumentation

In this section, we recall dynamic programming 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, for the framework F we consider the *underlying graph* G_F, where we simply drop the direction of each edge, i.e., $G_F = (A, R')$ where $R' \coloneqq \{\, \{u, v\} \mid (u, v) \in R \,\}$. Note that this does not affect the evaluation of the abstract argumentation framework along the tree decomposition. The parameter could be smaller if we keep the direction of the edges, but this would certainly require different algorithms. In fact, there exists the notion of directed treewidth (Johnson, Robertson, Seymour, & Thomas, 2001). In the following, we use the more common notion of undirected treewidth. Let $\mathcal{T} = (T, \chi)$ be a TD of the underlying graph of F. Furthermore, we need some auxiliary definitions. Let $T = (N, E_T, n)$ and $t \in N$. Then, 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 $F_t \coloneqq (A_t, R_t)$, where $A_t \coloneqq A \cap \chi(t)$ and $R_t \coloneqq (A_t \times A_t) \cap R$, the *framework below* t as $F_{\leqslant t} \coloneqq (A_{\leqslant t}, R_{\leqslant t})$, where $A_{\leqslant t} \coloneqq \{\, a \mid a \in \chi(t), t' \in \text{post-order}(T, t) \,\}$, and $R_{\leqslant t} \coloneqq (A_{\leqslant t} \times A_{\leqslant t}) \cap R$. It holds that $F_n = F_{\leqslant n} = F$.

A standard approach (Bodlaender & Kloks, 1996) to benefit algorithmically from small treewidths is to design dynamic programming algorithms that traverse a given TD and execute a so-called *local algorithm* $\mathbb{A}$ at each node. The local algorithm makes case distinctions based on the types 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 also access 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 a table to a node t of T. Then, the TTD is the triple $\mathcal{T} = (T, \chi, \tau)$. Later, for easy use in algorithms, we assume that $\tau(t)$ is initialized by the empty set for each node t of T. To solve the considered problem, we perform the following steps:

1. Compute a TD $\mathcal{T} = (T, \chi)$ of the underlying graph of F.

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

1 **if** type$(t) =$ *leaf* **then** $\tau_t \leftarrow \{\langle \emptyset, \emptyset, \emptyset \rangle\}$
2 **else if** type$(t) =$ *int and* $a \in \chi_t$ *is the introduced argument* **then**
3 $\quad \tau_t \leftarrow \{\langle J, O^{\uplus}_{A_t \rightarrowtail_{R_t} J}, D^{\uplus}_{A_t \leftarrowtail_{R_t} J} \rangle \mid \langle I, O, D \rangle \in \tau_1, J \in \{I, I^+_a\},$
$\qquad J \rightarrowtail_{R_t} J = \emptyset, J \cap \{c\} = \chi(t) \cap \{c\}\}$
4 **else if** type$(t) =$ *rem* **and** $a \notin \chi_t$ *is the removed argument* **then**
5 $\quad \tau_t \leftarrow \{\langle I^-_a, O^-_a, D^-_a \rangle \mid \langle I, O, D \rangle \in \tau_1, a \notin O \setminus D\}$
6 **else if** type$(t) =$ *join* **then**
7 $\quad \tau_t \leftarrow \{\langle I, O_1 {}^{\uplus}_{O_2}, D_1 {}^{\uplus}_{D_2} \rangle \mid \langle I, O_1, D_1 \rangle \in \tau_1, \langle I, O_2, D_2 \rangle \in \tau_2\}$
8 **return** τ_t

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

2. Run algorithm $\mathtt{DP}_{\mathbb{A}}$, which takes a TTD $\mathcal{T} = (T, \chi, \iota)$ with $T = (N, E_T, n)$ 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).

4.1 Credulous Reasoning

Dynamic programming algorithms for credulous reasoning of various semantics have already been established in the literature (Dvořák et al., 2012) and their implementations are also of practical interest (Dvořák, Morak, Nopp, & Woltran, 2013). While a dynamic programming algorithm for semi-stable (Bliem et al., 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 #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 (notice that '$\cdot$' are placeholders for positions that are not referenced. $\mathtt{DP}_{\mathbb{ADM}}$ outputs a new TTD that we will use to solve our actual counting problem. At each node t, we store the rows of the table $o(t)$ in the form $\vec{u} = \langle I, O, D \rangle$ and construct parts of the 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 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 have already been defeated (counterattacked) by any argument in the extension, thus in a sense compensating for the set O of attacking arguments. The idea of the algorithm is as follows. For nodes with type$(t) =$ *leaf*, Line 1 initially sets the extension part I, set O of

$\langle I_{1.i},$	$\mathcal{O}_{1.i},$	$\mathcal{D}_{1.i}\rangle$	i
$\langle\{a\},$	$\emptyset,$	$\{d\}\rangle$	1
$\langle\{a,p\},$	$\{d\},$	$\{d\}\rangle$	2

τ_1

Figure 6: The table for node t_1, obtained by Listing 1 and credulous argument $c = a$.

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 (type$(t) = \mathit{int}$), we guess whether $a \in I$ or $a \notin I$. We also make sure that I is conflict-free and that we only construct rows where $c \in I$ if $a = c$. Since ultimately every argument must be defended by the extension, we keep track of attacking arguments in O and defeated arguments in D. In Line 5, whenever we remove an argument a (type$(t) = \mathit{rem}$), we are not allowed to store a in the table anymore, because 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 lose the bound on the treewidth. However, we must make sure 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 no longer occurs in the bag, we have 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 (type$(t) = \mathit{join}$).

Intuitively, these dynamic programming algorithms compute and maintain tables from the leaf nodes towards the root of a tree decomposition.

Example 13. *Recall the argumentation framework F in Figure 4 and the tree decomposition $\mathcal{T}$ of F in Figure 5. Now, in order to maintain these tables, we execute the algorithm given in Listing 1. However, for simplicity, this listing is given for nice tree decompositions. While $\mathcal{T}$ is not nice, we can still easily follow Listing 1, but we would need to execute all cases for intermediate nodes that would be required to make $\mathcal{T}$ nice. We assume that the credulous argument $c = a$. In order to obtain the table for node t_1, we therefore follow the case for the empty leaf node (Line 1), followed by introducing arguments a, d, p (Line 2). Consequently, we obtain the table as highlighted in Figure 6. Indeed, since $c = a$, we can additionally either pick p or not. Observe that d can never be in any admissible extension including a, as the set would not be conflict-free.*

Similarly, we obtain the table for t_2. Observe that in the transition from t_2 to t_3 we forget r and introduce e. Then, from t_1 to t_4 we forget d, introduce e and join the result with the table obtained from the table for t_3 after forgetting s and introducing p. Finally, after computing the table for root node t_5, we can read the resulting admissible extensions restricted to $\{c, e, p\}$ from the table for t_5.

$\mathbb{ADM}$ 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 et al., 2012) that can be obtained by employing sophisticated data structures, especially for handling nodes t with $\text{type}(t) = \textit{join}$.

Proposition 14. *Algorithm* $\texttt{DP}_{\mathbb{STAB}}$ *runs in time* $\mathcal{O}(3^k \cdot k \cdot g)$, $\texttt{DP}_{\mathbb{ADM}}$ *in* $\mathcal{O}(4^k \cdot k \cdot g)$, *and* $\texttt{DP}_{\mathbb{COMP}}$ *in* $\mathcal{O}(5^k \cdot k \cdot g)$ *where* k *is the width and* g *the number of nodes of the TD.*

Proof (Sketch). Let $d = k + 1$ be maximum bag size of the TD $\mathcal{T}$. We only discuss the case for algorithm $\texttt{DP}_{\mathbb{ADM}}$ here. The table $\tau(t)$ has at most 4^d rows of the form $\langle I, \mathcal{A}, \mathcal{D} \rangle$, since an argument actually can be either in one of these sets $I, \mathcal{A}, \mathcal{D}$ or in none of them (just modify $\mathbb{ADM}$ such that $\mathcal{A} \cap \mathcal{D} = \emptyset$). In total, with the help of efficient data structures, e.g., for nodes t with $\text{type}(t) = \textit{join}$, one can establish a runtime bound of $\mathcal{O}(4^d)$. Then, we check within the bag for admissibility, keeping in mind only the changes and apply this to every node t of the TD, which resulting in running time $\mathcal{O}(4^d \cdot d \cdot g) \subseteq \mathcal{O}(4^k \cdot k \cdot g)$. □

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 dynamic programming, where the counter-witness tries to invalidate subset-maximality of the corresponding witness (Jakl, Pichler, & Woltran, 2009). In the counter-witness one stores sets of arguments that are supersets of the considered extension, so that in the end there was no superset of an extension in the counter-witness at the root while traversing the TD. In other words, for a witness, the counter-witness has failed to invalidate maximality, and accordingly the witness is subset-maximal. In the literature (Dvořák et al., 2012), algorithms that involving such an interplay between witnesses and counter-witnesses have been defined for preferred and semi-stable semantics, we simply refer to them as $\texttt{DP}_{\mathbb{PREF}}$ and $\texttt{DP}_{\mathbb{SEMI}}$.

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 I. This then allows us to subset-maximize on the range part $I \cup \mathcal{AC}$, by trying to find counter-witnesses $\mathcal{C}$ to subset-maximality. An element c of the set set $\mathcal{C}$ also comprises the three tuple components extension part J, attack candidates $\mathcal{AC}$, attacked arguments $\mathcal{A}$, but also contains an additional fourth component σ. This component σ is just a Boolean variable that indicates whether we found a witness that the range of J is strictly larger than the range of I. Indeed, we also need to maintain extensions, whose range is not strictly larger (yet, up to the current tree decomposition node). Note that the operator "$\vee$" used in Listing 2 is just a regular logical disjunction.

Example 15. *Recall the argumentation framework* F *in Figure 4 and the tree decomposition* $\mathcal{T}$ *of* F *in Figure 5. As the computation of stage semantics is more involved, also the local algorithm for maintaining tables takes more effort. Similar to above, in order to maintain these tables, we execute the algorithm given in Listing 2. We assume that the credulous argument* $c = a$. *In order to compute the table for node* t_1, *we follow the case for the empty leaf node (Line 1), followed by introducing arguments* a, d, p *(Line 2). This allows us to obtain the table as highlighted in Figure 7. Observe that this algorithm maintains potential subsets* I *of extensions with additional information on attacks* $\mathcal{A}$ *and candidate*

$\langle I_{1.i},$	$\mathcal{A}_{1.i},$	$\mathcal{AC}_{1.i},$	$\mathcal{C}_{1.i}\rangle$	i
$\langle\{a\},$	$\{d\},$	$\{d\},$	$\{\langle\langle\{a\},\{d\},\{d\}\rangle,\bot\rangle,$ $\langle\langle\{a\},\{d\},\{d,p\}\rangle,\bot\rangle,$ $\langle\langle\{a,p\},\{d\},\{d\}\rangle,\top\rangle\}\rangle$	1
$\langle\{a\},$	$\{d\},$	$\{d,p\},$	$\{\langle\langle\{a\},\{d\},\{d,p\}\rangle,\bot\rangle,$ $\langle\langle\{a,p\},\{d\},\{d\}\rangle,\bot\rangle\}\rangle$	2
$\langle\{a,p\},$	$\{d\},$	$\{d\}$	$\{\langle\langle\{a\},\{d\},\{d,p\}\rangle,\bot\rangle,$ $\langle\langle\{a,p\},\{d\},\{d\}\rangle,\bot\rangle\}\rangle$	3

τ_1

Figure 7: The table for node t_1, obtained by Listing 2 and credulous argument $c = a$.

attacks $\mathcal{AC}$. Further, the set $\mathcal{C}$ maintains extensions of potentially larger range ($\top$ indicates strictly larger range).

Again, a more detailed runtime analysis yields the following result.

Proposition 16. *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.*

Proof. For each node t of T, we consider the table $\nu(t)$ of $\mathcal{T}_{\text{purged}}$. Let TDD (T, χ, π) be the output of $\mathtt{DP}_{\mathbb{PROJ}}$. In the worst case, we store in $\pi(t)$ each subset $\rho \subseteq \nu(t)$ together with exactly one counter. Hence, we have at most 2^m many rows in ρ. In order to compute ipc for ρ, we consider every subset $\varphi \subseteq \rho$ and compute pc. Since $|\rho| \leqslant m$, we have at most 2^m many subsets φ of ρ. Finally, for computing pc, we consider in the worst case each subset of the origins of φ for each child table, which are at most $2^m \cdot 2^m$ because of nodes t with $\text{type}(t) = \textit{join}$. In total, we obtain a runtime bound of $\mathcal{O}(2^m \cdot 2^m \cdot 2^m \cdot 2^m \cdot \gamma(\|F\|)) \subseteq \mathcal{O}(2^{4m} \cdot \gamma(\|F\|))$ due to multiplication of two n-bit integers for nodes t with $\text{type}(t) = \textit{join}$ at costs $\gamma(n)$. Then, we apply this to every node of T resulting in runtime $\mathcal{O}(2^{4m} \cdot g \cdot \gamma(\|F\|))$. □

4.2 Lower Bounds

A natural question is whether we can significantly improve the algorithms given in propositions 14 and 16. 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 to 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). We then establish lower bounds, assuming that ETH uses known reductions from the literature, and show that there is no hope for a better algorithm.

Theorem 17. *Let $\mathcal{S} \in \{\textit{admissible, complete, stable}\}$, 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 $\mathcal{S} = \textit{semi-stable}$, $\mathsf{Cred}_{\mathcal{S}}$ and $\mathsf{Skep}_{\mathcal{S}}$ cannot be solved in time $2^{2^{o(k)}} \cdot \|F\|^{o(k)}$.*

Proof. 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. □

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

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

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

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

$$\mathcal{C}_1 \bowtie \mathcal{C}_2 \coloneqq \{ \langle \langle I, \mathcal{A}_1 \overset{\uplus}{\mathcal{A}_2}, \mathcal{AC} \rangle, \sigma_1 \vee \sigma_2 \rangle \mid \langle \langle I, \mathcal{A}_1, \mathcal{AC} \rangle, \sigma_1 \rangle \in \mathcal{C}_1, \langle \langle I, \mathcal{A}_2, \mathcal{AC} \rangle, \sigma_2 \rangle \in \mathcal{C}_2 \}.$$

Listing 3 presents a local algorithm $\mathbb{CONF}$ for conflict-free extensions, whose core is also used in Listing 2. A local algorithm $\mathbb{STAB}$ for stable extensions, which, in fact, is a simplification of Listing 1, is provided in Listing 4. Finally, Listing 5 depicts an algorithm $\mathbb{COMP}$ for complete semantics working with five different states, as mentioned in Section "Dynamic Programming for Abstract Argumentation". For computing preferred semantics via dynamic programming ($\mathtt{DP}_{\mathbb{PREF}}$), one can use the idea of the local algorithm $\mathbb{ADM}$ for admissible semantics and subset-maximize using counterwitnesses (similar to Listing 2) accordingly. Finally, local algorithm $\mathbb{SEMI}$ finally is similar to $\mathbb{STAB}$, but relies on the idea of $\mathbb{ADM}$.

Listing 3: Local algorithm $\mathbb{CONF}(t, \chi_t, \cdot, (F_t, c, \cdot), \langle \tau_1, \tau_2 \rangle)$.

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** type$(t) =$ *leaf* **then** $\tau_t \leftarrow \{\langle \emptyset \rangle\}$
2 **else if** type$(t) =$ *int and* $a \in \chi_t$ *is the introduced argument* **then**
3 $\quad \tau_t \leftarrow \{\langle J \rangle \mid \langle I \rangle \in \tau_1, J \in \{I, I^+_a\}, J \rightarrowtail_{R_t} J = \emptyset, J \cap \{c\} = \chi(t) \cap \{c\}\}$
4 **else if** type$(t) =$ *rem* **and** $a \notin \chi_t$ *is the removed argument* **then**
5 $\quad \tau_t \leftarrow \{\langle I^-_a \rangle \mid \langle I \rangle \in \tau_1\}$
6 **else if** type$(t) =$ *join* **then**
7 $\quad \tau_t \leftarrow \{\langle I \rangle \mid \langle I \rangle \in \tau_1, \langle I \rangle \in \tau_2\}$
8 **return** τ_t

Listing 4: Local algorithm $\mathbb{STAB}(t, \chi_t, \cdot, (F_t, c, \cdot), \langle \tau_1, \tau_2 \rangle)$, c.f., (Dvořák et al., 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.

1 **if** type$(t) =$ *leaf* **then** $\tau_t \leftarrow \{\langle \emptyset, \emptyset \rangle\}$
2 **else if** type$(t) =$ *int and* $a \in \chi_t$ *is the introduced argument* **then**
3 $\quad \tau_t \leftarrow \{\langle J, D \uplus_{A_t \leftarrow_{R_t} J} \rangle \mid \langle I, D \rangle \in \tau_1, J \in \{I, I^+_a\}, J \rightarrowtail_{R_t} J = \emptyset, J \cap \{c\} = \chi(t) \cap \{c\}\}$
4 **else if** type$(t) =$ *rem* **and** $a \notin \chi_t$ *is the removed argument* **then**
5 $\quad \tau_t \leftarrow \{\langle I^-_a, D^-_a \rangle \mid \langle I, D \rangle \in \tau_1, a \in I \cup D\}$
6 **else if** type$(t) =$ *join* **then**
7 $\quad \tau_t \leftarrow \{\langle I, D_1 \uplus D_2 \rangle \mid \langle I, D_1 \rangle \in \tau_1, \langle I, D_2 \rangle \in \tau_2\}$
8 **return** τ_t

Listing 5: Local algorithm $\mathbb{COMP}(t, \chi_t, \cdot, (F_t, c, \cdot), \langle \tau_1, \tau_2 \rangle)$, c.f., (Dvořák et al., 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.

1 **if** type$(t) =$ *leaf* **then** $\tau_t \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$
2 **else if** type$(t) =$ *int and* $a \in \chi_t$ *is the introduced argument* **then**
3 $\quad \tau_t \leftarrow \{\langle J, \mathcal{D} \uplus_{D \leftarrow_{R_t} J}, D, \mathcal{O} \uplus_{O \leftarrow_{R_t} O}, O \rangle \mid \langle I, \mathcal{D}, \mathcal{DC}, \mathcal{O}, \mathcal{OC} \rangle \in \tau_1, J \in \{I, I^+_a\},$
$\quad\quad D \in \{\mathcal{DC}, \mathcal{DC}^+_a\}, O \in \{\mathcal{OC}, \mathcal{OC}^+_a\}, J \cap D \cap O = \emptyset, J \rightarrowtail_{R_t} J = \emptyset, J \rightarrowtail_{R_t} O = \emptyset,$
$\quad\quad O \rightarrowtail_{R_t} J = \emptyset, J \cap \{c\} = \chi(t) \cap \{c\}\}$
4 **else if** type$(t) =$ *rem* **and** $a \notin \chi_t$ *is the removed argument* **then**
5 $\quad \tau_t \leftarrow \{\langle I^-_a, \mathcal{D}^-_a, \mathcal{DC}^-_a, \mathcal{O}^-_a, \mathcal{OC}^-_a \rangle \mid \langle I, \mathcal{D}, \mathcal{DC}, \mathcal{O}, \mathcal{OC} \rangle \in \tau_1, a \in I \cup \mathcal{D} \cup \mathcal{O}\}$
6 **else if** type$(t) =$ *join* **then**
7 $\quad \tau_t \leftarrow \{\langle I, \mathcal{D}_1 \uplus \mathcal{D}_2, \mathcal{DC}, \mathcal{O}_1 \uplus \mathcal{O}_2, \mathcal{OC} \rangle \mid \langle I, \mathcal{D}_1, \mathcal{DC}, \mathcal{O}_1, \mathcal{OC} \rangle \in \tau_1, \langle I, \mathcal{D}_2, \mathcal{DC}, \mathcal{O}_2, \mathcal{OC} \rangle \in \tau_2\}$
8 **return** τ_t

5. Algorithms for Projected Credulous Counting by Exploiting Bounded Treewidth

In the previous section, we presented algorithms for solving abstract reasoning problems. These algorithms can be extended relatively easily to count extensions without projection by adding counters to each row at quadratic runtime instead of linear in the size of the input instance. One can even reconstruct extensions (Pichler, Rümmele, & 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., 2023) 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 $T = (N, E_T, n)$ 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 intermediate traverse $\mathcal{T}_{\mathbb{S}\text{-}Cred}$ in pre-order and *prune irrelevant rows*, thereby removing all rows that cannot be extended to a credible 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, since it only removes rows where the count is already 0 without considering the projection. However, pruning serves as a technical trick for the last traversal to avoid counter correction 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 $\mathtt{DP}_{\mathbb{PROJ}}$ using local algorithm $\mathbb{PROJ}$ as given in Listing 6. 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 need a notion to reconstruct extensions of $\mathcal{T}$, more precisely, for a given row to define its predecessors in the corresponding child tables. Therefore, let t be a node of T with children t_1 and t_2, if it exists. Since the sequences used in the following depend on the number of children, it is assumed for the sake of simplicity that sequences are implicitly of appropriate length, even if they are given as of length 2. For sequence $\vec{s} = \langle s_1, s_2 \rangle$, let $\langle\{\vec{s}\}\rangle \coloneqq \langle \{s_1\}, \{s_2\} \rangle$. For a given row $\vec{u} \in \tau(t)$, we define the originating rows of $\vec{u}$ in node t by $\mathrm{origins}(t, \vec{u}) \coloneqq \{ \vec{s} \mid \vec{s} \in \tau(t_1) \times \tau(t_2), \vec{u} \in \mathbb{S}(t, \chi(t), \cdot, (F_t, \cdot), \langle\{\vec{s}\}\rangle) \}$ and for a table σ as the union over the origins for all rows $\vec{u} \in \sigma$. Next, let $\sigma \subseteq \nu(t)$. To combine rows and solve the 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 $=_{\mathrm{P}} \coloneqq \{ (\vec{u}, \vec{v}) \mid \vec{u}, \vec{v} \in \sigma, E(\vec{u}) \cap P = E(\vec{v}) \cap P \}$. Let $\mathrm{buckets}_P(\sigma)$ be the set of equivalence classes induced by $=_{\mathrm{P}}$ on σ, i.e., $\mathrm{buckets}_P(\sigma) \coloneqq (\sigma / =_{\mathrm{P}}) = \{ [\vec{u}]_P \mid \vec{u} \in \sigma \}$, 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 $\text{s-ipc}(\pi(t), \sigma) \coloneqq \sum_{\langle \sigma, c \rangle \in \pi(t)} c$. 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

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

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 $\mathrm{ipc}(t, \sigma, s) \coloneqq 1$ if $\mathrm{type}(t) = \textit{leaf}$ and otherwise $\mathrm{ipc}(t, \sigma, s) \coloneqq \big|\, \mathrm{pc}(t, \sigma, s) + \sum_{\emptyset \subsetneq \varphi \subsetneq \sigma} (-1)^{|\varphi|} \cdot \mathrm{ipc}(t, \varphi, s) \big|$, 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 $\nu(t)$.

Listing 6: 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., 2023).

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 $c \in \mathbb{N}$.
1 $\pi_t \leftarrow \{ \langle \sigma, \mathrm{ipc}(t, \sigma, \langle \pi_1, \pi_2 \rangle) \rangle \mid \emptyset \subsetneq \sigma \subseteq \mathrm{buckets}_P(\nu_t) \}$
2 **return** π_t

Theorem 18. *Algorithm* $\mathtt{PCNT}_{\mathbb{S}}$ *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 projected credulous count at the root* n *for resp. semantics* $\mathcal{S}$*.*

Proof. We can establish an invariant for each row of each table. Then, we show this invariant by simultaneous 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 for the algorithm $\mathbb{PROJ}$, which directly lead to runtime results for $\mathtt{PCNT}_{\mathbb{S}}$. So let $\gamma(n)$ be the number of operations needed to multiply two n-bit integers. Note that $\gamma(n) \in O(n \cdot \log(n) \cdot \log(\log(n)))$ (Knuth, 1998). Further note that in the following proposition m depends on the treewidth k. However, the actual order depends on the semantics.

Proposition 19 (Fichte and Hecher, (2018))**.** $\mathtt{DP}_{\mathbb{PROJ}}$ *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* $m \coloneqq \max\{|\nu(t)| \mid t \in N\}$ *for input TTD* $\mathcal{T}_{purged} = (T, \chi, \nu)$ *of* $\mathtt{DP}_{\mathbb{PROJ}}$*.*

Corollary 20. *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 again consider the Exponential Time Hypothesis (ETH) to establish lower bounds for counting projected extensions. In particular, we find that, under reasonable assumptions, we cannot expect to significantly improve the algorithms presented.

Theorem 21. *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. 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 & 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 10. 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. Note 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 the ETH for quantified Boolean formulas (QBF) together with the following result.

Proposition 22 (Fichte, Hecher, and Pfandler, 2020, Thm. 13)**.** *Unless ETH fails, 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 ETH together with the previous proposition, we establish the following result.

Theorem 23. *Let* $\mathcal{S} \in \{$*preferred, semi-stable, stage*$\}$ *be a semantics. Unless ETH 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. Assuming ETH, Proposition 22 implies that 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 21). Thereby, we apply the reduction provided in Statement 1 of Lemma 10, set $\ell \coloneqq |X|$ and proceed analogously to Theorem 21. □

6. Conclusion and Outlook

We classified the classical complexity of counting problems in abstract argumentation. Furthermore, we presented 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. Then, assuming ETH, we established that the runtime of the algorithms are asymptotically tight. While the upper bounds in Lemma 5 can be easily transferred to counting, the number of extensions of a certain kind, the corresponding lower bounds cannot be immediately adopted directly from Lemma 6. Moreover, we derived similar results for counting (projected) extensions.

An open question is to investigate whether $\# \cdot$ 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. While dynamic programming techniques have already been used for counting extensions (without projections) (Dewoprabowo, Fichte, Gorczyca, & Hecher, 2022), techniques for incremental counting (Fichte, Gaggl, Hecher, & Rusovac, 2024), massive parallelization (Fichte, Hecher, & Roland, 2021), or approximate

counting (Kabir, Everardo, Shukla, Hecher, Fichte, & Meel, 2022) could be interesting for argumentation as well. Today, the best argumentation-based solvers (Niskanen & Järvisalo, 2020, 2023) rely on techniques from SAT-based solving in its core (Fichte, Berre, Hecher, & Szeider, 2023). So it might also be interesting to consider SAT-based counting techniques (Fichte et al., 2021) for argumentation. Furthermore, our technique might also be applicable to problems such as circumscription (Durand et al., 2005), default logic (Fichte, Hecher, & Schindler, 2022a), or QBFs (Charwat & Woltran, 2016). Since argumentation has been applied to questions in human reasoning (Dietz, Kakas, & Michael, 2021, 2022) and quantitative reasoning plays an interesting role there as well (Dietz, Fichte, & Hamiti, 2022), we suspect that counting and probabilistic questions can have an interesting application there as well. Considering the (parameterized) enumeration complexity (Johnson, Papadimitriou, & Yannakakis, 1988; Creignou, Meier, Müller, Schmidt, & Vollmer, 2017; Creignou, Ktari, Meier, Müller, Olive, & Vollmer, 2019; Meier, 2020) of the studied problems is also planned as future work. Finally, other measures such as fractional hypertree width or backdoors (Dvořák, Hecher, König, Schidler, Szeider, & Woltran, 2022) might be interesting to consider. Also studying implementation aspect might yield insights that lead to algorithmic improvements. Lower bounds for other decomposition-based parameters have recently been established for QBF (Fichte, Ganian, Hecher, Slivovsky, & Ordyniak, 2023), we suspect that these results can be directly applied to argumentation using decomposition guided reductions (Fichte, Hecher, Mahmood, & Meier, 2021).

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. Authors are ordered alphabetically. The work has been carried out while Hecher visited the Simons Institute at UC Berkeley. Research is supported by the the Austrian Science Fund (FWF) grant J4656; ELLIIT funded by the Swedish government; the German Research Fund DFG grants ME 4279/1-2 and ME 4279/3-1, and the Society for Research Funding in Lower Austria (GFF) grant ExzF-0004.

References

Abseher, M., Musliu, N., & Woltran, S. (2017). htd – a free, open-source framework for (customized) tree decompositions and beyond. In Salvagnin, D., & Lombardi, M. (Eds.), *Proceedings of the 14th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR'17)*, Padova, Italy.

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

Arnborg, S., Corneil, D. G., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, *8*(2), 277–284.

Audemard, G., Lagniez, J.-M., & Miceli, M. (2022). A new exact solver for (weighted) Max#SAT. In Meel, K. S., & Strichman, O. (Eds.), *Proceedings of the 25th International Conference on Theory and Applications of Satisfiability Testing (SAT'22)*,

Vol. 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 28:1–28:20. Dagstuhl Publishing.

Aziz, R. A., Chu, G., Muise, C., & Stuckey, P. (2015). #(∃)SAT: Projected Model Counting. In Heule, M., & Weaver, S. (Eds.), *Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT'15)*, pp. 121–137, Austin, TX, USA. Springer Verlag.

Baroni, P., Dunne, P. E., & Giacomin, M. (2010). On extension counting problems in argumentation frameworks. In *COMMA*, Vol. 216 of *Frontiers in Artificial Intelligence and Applications*, pp. 63–74. IOS Press.

Bliem, B., Hecher, M., & Woltran, S. (2016). On efficiently enumerating semi-stable extensions via dynamic programming on tree decompositions. In Baroni, P., Gordon, T. F., Scheffler, T., & Stede, M. (Eds.), *Proceedings of the 6th International Conference on Computational Models of Argument (COMMA'16)*, Vol. 287 of *Frontiers in Artificial Intelligence and Applications*, pp. 107–118, Potsdam, Germany. IOS Press.

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

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

Böhl, E., Gaggl, S. A., & Rusovac, D. (2023). Representative answer sets: Collecting something of everything. In Gal, K., Nowé, A., Nalepa, G. J., Fairstein, R., & Radulescu, R. (Eds.), *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI'23)*, Vol. 372 of *Frontiers in Artificial Intelligence and Applications*, pp. 271–278, Kraków, Poland. IOS Press.

Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory*, Vol. 244 of *Graduate Texts in Mathematics*. Springer Verlag, New York, USA.

Charwat, G., & Woltran, S. (2016). Dynamic programming-based QBF solving. In Lonsing, F., & Seidl, M. (Eds.), *Proceedings of the 4th International Workshop on Quantified Boolean Formulas (QBF'16)*, Vol. 1719, pp. 27–40. CEUR Workshop Proceedings (CEUR-WS.org). co-located with 19th International Conference on Theory and Applications of Satisfiability Testing (SAT'16).

Coste-Marquis, S., Devred, C., & Marquis, P. (2005). Symmetric argumentation frameworks. In *ECSQARU*, Vol. 3571 of *Lecture Notes in Computer Science*, pp. 317–328. Springer.

Creignou, N., Meier, A., Müller, J., Schmidt, J., & Vollmer, H. (2017). Paradigms for parameterized enumeration. *Theory Comput. Syst.*, *60*(4), 737–758.

Creignou, N., Ktari, R., Meier, A., Müller, J., Olive, F., & Vollmer, H. (2019). Parameterised enumeration for modification problems. *Algorithms*, *12*(9), 189.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., & Saurabh, S. (2015). *Parameterized Algorithms*. Springer Verlag.

Dachselt, R., Gaggl, S. A., Krötzsch, M., Méndez, J., Rusovac, D., & Yang, M. (2022). NEXAS: A visual tool for navigating and exploring argumentation solution spaces. In Toni, F., Polberg, S., Booth, R., Caminada, M., & Kido, H. (Eds.), *Proceedings of the*

9th International Conference on Computational Models of Argument Computational Models of Argument (COMMA'22), Vol. 353 of *Frontiers in Artificial Intelligence and Applications*, pp. 116–127, Cardiff, Wales, UK. IOS Press.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *J. Artif. Intell. Res.*, *17*(1), 229–264.

De Raedt, L., & Kimmig, A. (2015). Probabilistic (logic) programming concepts. *Machine Learning*, *100*(1), 5–47.

Dewoprabowo, R., Fichte, J. K., Gorczyca, P. J., & Hecher, M. (2022). A practical account into counting dung's extensions by dynamic programming. In Gottlob, G., Inclezan, D., & Maratea, M. (Eds.), *Proceedings of the 16th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'22)*, pp. 387–400, Genova, Italy. Springer Verlag.

Dietz, E., Fichte, J. K., & Hamiti, F. (2022). A quantitative symbolic approach to individual human reasoning. In Culbertson, J., Perfors, A., Rabagliati, H., & Ramenzoni, V. (Eds.), *Proceedings of the 44th Annual Conference of the Cognitive Science Society (CogSci'22)*, pp. 2838–2846.

Dietz, E., Kakas, A. C., & Michael, L. (2021). Computational argumentation & cognitive AI. In Chetouani, M., Dignum, V., Lukowicz, P., & Sierra, C. (Eds.), *Human-Centered Artificial Intelligence - Advanced Lectures, 18th European Advanced Course on AI, ACAI 2021, Berlin, Germany, October 11-15, 2021, extended and improved lecture notes*, Vol. 13500 of *Lecture Notes in Computer Science*, pp. 363–388. Springer.

Dietz, E., Kakas, A. C., & Michael, L. (2022). Argumentation: A calculus for human-centric AI. *Frontiers Artif. Intell.*, *5*.

Downey, R. G., & Fellows, M. R. (1999). *Parameterized Complexity*. Monographs in Computer Science. Springer.

Downey, R. G., & Fellows, M. R. (2013). *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer Verlag, London, UK.

Dueñas-Osorio, L., Meel, K. S., Paredes, R., & Vardi, M. Y. (2017). Counting-based reliability estimation for power-transmission grids. In *AAAI*, pp. 4488–4494. AAAI Press.

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

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

Dunne, P. E., Hunter, A., McBurney, P., Parsons, S., & Wooldridge, M. (2011). Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, *175*(2), 457–486.

Durand, A., Hermann, M., & Kolaitis, P. G. (2005). Subtractive reductions and complete problems for counting complexity classes. *Theoretical Computer Science*, *340*(3), 496–513.

Dvořák, W., Morak, M., Nopp, C., & Woltran, S. (2013). dynpartix - a dynamic programming reasoner for abstract argumentation. In *Applications of Declarative Programming and Knowledge Management*, pp. 259–268. Springer Berlin Heidelberg.

Dvořák, W. (2012). *Computational aspects of abstract argumentation.* Ph.D. thesis, TU Wien.

Dvořák, W., Hecher, M., König, M., Schidler, A., Szeider, S., & Woltran, S. (2022). Tractable abstract argumentation via backdoor-treewidth. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(5), 5608–5615.

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

Dvořák, W., Pichler, R., & Woltran, S. (2012). Towards fixed-parameter tractable algorithms for abstract argumentation. *Artificial Intelligence*, *186*, 1–37.

Ebbinghaus, H., Flum, J., & Thomas, W. (1994). *Mathematical logic (2. ed.).* Undergraduate texts in mathematics. Springer Verlag.

Eiter, T., Fichte, J. K., Hecher, M., & Woltran, S. (2024). Epistemic logic programs: Non-ground and counting complexity. In Larson, K. (Ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24).* To appear.

Fichte, J., Hecher, M., Mahmood, Y., & Meier, A. (2021). Decomposition-guided reductions for argumentation and treewidth. In Zhou, Z.-H. (Ed.), *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21)*, pp. 1880–1886. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Fichte, J. K., Berre, D. L., Hecher, M., & Szeider, S. (2023). The silent (r)evolution of SAT. *Communications of the ACM*, *66*(6), 64–72.

Fichte, J. K., Gaggl, S. A., Hecher, M., & Rusovac, D. (2024). IASCAR: Incremental answer set counting by anytime refinement. *Theory Pract. Log. Program.*, *2*, 1–28.

Fichte, J. K., Ganian, R., Hecher, M., Slivovsky, F., & Ordyniak, S. (2023). Structure-aware lower bounds and broadening the horizon of tractability for QBF. In *Procedings of the 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'23)*, pp. 1–14.

Fichte, J. K., & Hecher, M. (2018). Exploiting treewidth for counting projected answer sets. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR'18).*

Fichte, J. K., Hecher, M., & Hamiti, F. (2021). The model counting competition 2020. *ACM J. Exp. Algorithmics*, *26*.

Fichte, J. K., Hecher, M., Morak, M., Thier, P., & Woltran, S. (2023). Solving projected model counting by utilizing treewidth and its limits. *Artificial Intelligence*, *314*, 103810.

Fichte, J. K., Hecher, M., & Nadeem, M. A. (2022). Plausibility reasoning via projected answer set counting - a hybrid approach. In Raedt, L. D. (Ed.), *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'22)*, pp. 2620–2626. International Joint Conferences on Artificial Intelligence Organization.

Fichte, J. K., Hecher, M., & Roland, V. (2021). Parallel model counting with cuda: Algorithm engineering for efficient hardware utilization. In *Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming (CP'21)*. Dagstuhl Publishing.

Fichte, J. K., Hecher, M., & Schindler, I. (2022a). Default logic and bounded treewidth. *Information and Computation*, *283*, 104675. Selected papers of the 12th International Conference on Language and Automata Theory and Applications, LATA 2018.

Fichte, J. K., Gaggl, S. A., & Rusovac, D. (2022b). Rushing and strolling among answer sets – navigation made easy. In Honavar, V., & Spaan, M. (Eds.), *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*, pp. 5651–5659.

Fichte, J. K., Hecher, M., Mahmood, Y., & Meier, A. (2023). Quantitative reasoning and structural complexity for claim-centric argumentation. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI'23)*, pp. 3212–3220. ijcai.org.

Fichte, J. K., Hecher, M., & Meier, A. (2019). Counting complexity for reasoning in abstract argumentation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, pp. 2827–2834. The AAAI Press.

Fichte, J. K., Hecher, M., & Pfandler, A. (2020). Lower bounds for QBFs of bounded treewidth. In Hermanns, H., Zhang, L., Kobayashi, N., & Miller, D. (Eds.), *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'20)*, pp. 410–424. Assoc. Comput. Mach., New York.

Graham, R. L., Grötschel, M., & Lovász, L. (1995). *Handbook of combinatorics*, Vol. I. Elsevier Science Publishers, North-Holland.

Hecher, M., & Fichte, J. K. (2023). The 4th competition on model counting (MC 2023). `https://mccompetition.org/past_iterations`.

Hecher, M., Mahmood, Y., Meier, A., & Schmidt, J. (2024). Quantitative claim-centric reasoning in logic-based argumentation. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI'24)*. ijcai.org. To appear.

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

Impagliazzo, R., Paturi, R., & Zane, F. (2001). Which problems have strongly exponential complexity?. *J. of Computer and System Sciences*, *63*(4), 512–530.

Jakl, M., Pichler, R., & Woltran, S. (2009). Answer-set programming with bounded treewidth. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, Vol. 2, pp. 816–822.

Johnson, D. S., Papadimitriou, C. H., & Yannakakis, M. (1988). On generating all maximal independent sets. *Inf. Process. Lett.*, *27*(3), 119–123.

Johnson, T., Robertson, N., Seymour, P. D., & Thomas, R. (2001). Directed tree-width. *J. Combin. Theory Ser. B*, *82*(1), 138–154.

Kabir, M., Everardo, F. O., Shukla, A. K., Hecher, M., Fichte, J. K., & Meel, K. S. (2022). ApproxASP – a scalable approximate answer set counter. In Honavar, V., & Spaan, M.

(Eds.), *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*, pp. 5755–5764.

Käfer, N., Baier, C., Diller, M., Dubslaff, C., Gaggl, S. A., & Hermanns, H. (2022). Admissibility in probabilistic argumentation. *J. Artif. Intell. Res.*, *74*.

Kloks, T. (1994). *Treewidth. Computations and Approximations*, Vol. 842 of *Lecture Notes in Computer Science*. Springer Verlag.

Knuth, D. E. (1998). How fast can we multiply?. In *The Art of Computer Programming* (3 edition)., Vol. 2 of *Seminumerical Algorithms*, chap. 4.3.3, pp. 294–318. Addison-Wesley.

Konieczny, S., Marquis, P., & Vesic, S. (2015a). On supported inference and extension selection in abstract argumentation frameworks. In Destercke, S., & Denoeux, T. (Eds.), *Proceeding of the 13th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'15)*, pp. 49–59, Cham. Springer Verlag.

Konieczny, S., Marquis, P., & Vesic, S. (2015b). On supported inference and extension selection in abstract argumentation frameworks. In *ECSQARU*, Vol. 9161 of *Lecture Notes in Computer Science*, pp. 49–59. Springer.

Lagniez, J.-M., Lonca, E., Mailly, J.-G., & Rossit, J. (2020). Introducing the fourth international competition on computational models of argumentation. In Gaggl, S. A., Thimm, M., & Vallati, M. (Eds.), *Proceedings of the 3rd International Workshop on Systems and Algorithms for Formal Argumentation co-located with the 8th International Conference on Computational Models of Argument (COMMA 2020)*, Vol. 2672, pp. 80–85. CEUR Workshop Proceedings (CEUR-WS.org).

Lagniez, J.-M., Lonca, E., Mailly, J.-G., & Rossit, J. (2021). Results of the fourth international competition on computational models of argumentation. `https://argumentationcompetition.org/2021/downloads/iccma_results_ijcai.pdf`.

Lampis, M., Mengel, S., & Mitsou, V. (2018). QBF as an Alternative to Courcelle's Theorem. In Beyersdorff, O., & Wintersteiger, C. M. (Eds.), *Theory and Applications of Satisfiability Testing – SAT 2018*, pp. 235–252. Springer Verlag.

Lampis, M., & Mitsou, V. (2017). Treewidth with a quantifier alternation revisited. In Lokshtanov, D., & Nishimura, N. (Eds.), *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC'17)*. Dagstuhl Publishing.

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

Meier, A. (2020). *Parametrised enumeration*. Gottfried Wilhelm Leibniz Universität Hannover, Hannover, Germany. Habilitation thesis. `https://doi.org/10.15488/9427`.

Niskanen, A., & Järvisalo, M. (2020). μ-toksia: An Efficient Abstract Argumentation Reasoner. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR'20)*, pp. 800–804.

Niskanen, A., & Järvisalo, M. (2023). μ-toksia in iccma 2023. Tech. rep., Department of Computer Science Series of Publications B, University of Helsinki.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley.

Pichler, R., Rümmele, S., & Woltran, S. (2010). Counting and enumeration problems with bounded treewidth. In Clarke, E. M., & Voronkov, A. (Eds.), *Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'10)*, Vol. 6355 of *Lecture Notes in Computer Science*, pp. 387–404. Springer Verlag.

Rago, A., Cocarascu, O., & Toni, F. (2018). Argumentation-based recommendations: Fantastic explanations and how to find them. In Lang, J. (Ed.), *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, pp. 1949–1955, Stockholm, Sweden. The AAAI Press.

Rahwan, I. (2007). Argumentation in artificial intelligence. *Artificial Intelligence*, *171*(10-15), 619–641.

Robertson, N., & Seymour, P. D. (1986). Graph minors. II. algorithmic aspects of treewidth. *J. Algorithms*, *7*(3), 309–322.

Toda, S., & Watanabe, O. (1992). Polynomial time 1-Turing reductions from #PH to #P. *Theor. Comput. Sci.*, *100*(1), 205–221.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theor. Comput. Sci.*, *8*, 189–201.

Vigouroux, T., Bozga, M., Ene, C., & Mounier, L. (2024). Function synthesis for maximizing model counting. In Dimitrova, R., Lahav, O., & Wolff, S. (Eds.), *Proceedings of the 25th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'24)*, pp. 258–279. Springer Verlag.

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

Yang, J., Chakraborty, S., & Meel, K. S. (2022). Projected model counting: Beyond independent support. In Bouajjani, A., Holík, L., & Wu, Z. (Eds.), *Proceedings of the 20th International Symposium on Automated Technology for Verification and Analysis (ATVA'22)*, pp. 171–187. Springer Verlag.