Reasoning in Assumption-Based Argumentation Using Tree-Decompositions

Andrei Popescu and Johannes P. Wallner(✉)

Institute of Software Technology, Graz University of Technology, Graz, Austria
{andrei.popescu,wallner}@ist.tugraz.at

Abstract. We address complex reasoning tasks in assumption-based argumentation (ABA) by developing dynamic programming algorithms based on tree-decompositions. As one of the prominent approaches in computational argumentation, our focus is on NP-hard reasoning in ABA. We utilize tree-width, a structural measure describing closeness to trees, for an approach to handle computationally complex tasks in ABA. We contribute to the state of the art by first showing that many reasoning tasks in ABA are fixed-parameter tractable w.r.t. tree-width using Courcelle's theorem, informally signaling wide applicability of dynamic programming algorithms for ABA. Secondly, we develop such algorithms operating on tree-decompositions of given ABA frameworks. We instantiate the algorithms in the recent D-FLAT framework allowing for declarative and extensible specification of dynamic programming algorithms. In an experimental evaluation on a resulting prototype, we show promise of the approach in particular for complex counting tasks.

1 Introduction

Computational approaches to arguing in favour or against statements under scrutiny are a main research theme in the field of computational argumentation [4,38], within Artificial Intelligence (AI). Placed in the area of knowledge representation and reasoning and non-monotonic reasoning, computational argumentation features a diverse set of application avenues, such as legal reasoning, medical reasoning, and e-government [3].

Approaches to formalize argumentative reasoning are studied in the field of structured argumentation [5]. Formalisms in structured argumentation usually follow the so-called argumentation workflow [14] to prescribe ways of finding arguments and their relationships. A starting point are knowledge bases, oftentimes assumed to be in a rule-based form. Arguments are then instantiated as derivations applicable within the knowledge base. Reasoning based on the arguments and their relations is carried out by using argumentation semantics, through which one can specify acceptable sets of arguments. Several approaches to structured argumentation have been studied, e.g., assumption-based argumentation (ABA) [12,20], ASPIC$^+$ [50,51], defeasible logic programming (DeLP) [40,41], and deductive argumentation [6,7].

S. Gaggl et al. (Eds.): JELIA 2023, LNAI 14281, pp. 192–208, 2023.
https://doi.org/10.1007/978-3-031-43619-2_14

Computationally speaking, argumentative reasoning in these approaches to structured argumentation is hard: in almost all cases reasoning defined in these formalisms is NP-hard, see, e.g., the survey by Dunne and Dvořák [32]. To address the complexity barrier, several algorithmic approaches were developed [15,16], and in a biannual International Competition on Computational Models of Argumentation (ICCMA) [8,39,46,53], which this year is being held for the fifth time, systems compete in terms of runtime performance.

An approach to tackle high complexity is the utilization of structural properties of instances, such as viewing instances in a graph-like manner and considering, e.g., acyclicity or other graph properties. A prominent such property is tree-width [10], informally measuring closeness of instances to trees. The milder complexity of many problems on trees oftentimes transfers to problems on graphs of low tree-width. Algorithms following dynamic programming can then operate on a tree-decomposition of the initial instance, with which one confines the combinatorial explosion of complex problems into subproblems, whose size can be bounded by the tree-width of the original instance. Tree-based forms, or forms close to trees, appear appealing to computational argumentation, since, e.g., dialogues might be represented in a tree-like structure. Indeed, tree-width has been studied in several works in argumentation [27–31,34,47]. These studies focus on the field of abstract argumentation, i.e., on formalisms where arguments are given in an abstracted form such as the well-known argumentation framework (AF) [26], and a form of deductive argumentation. To the best of our knowledge, there is no current account of the utilization of tree-width for rule-based structured argumentation formalisms such as ABA, ASPIC$^+$, or DeLP.

Recent works show that lifting computational approaches in abstract argumentation to structured argumentation is not immediate, and seems to involve dedicated research on the structured formalisms [48,49]. We follow this line and take up this opportunity to study algorithmic approaches utilizing tree-width for ABA, as one of the prominent structured argumentation approaches with applications in medical decision making [19,22] and in multi-agent contexts [33].

Our main contributions are as follows.

- We first show wide applicability of algorithms using tree-width by showing fixed-parameter tractability of reasoning tasks in ABA, under the parameter tree-width. We show these results by making use of Courcelle's theorem [17] and expressing reasoning in ABA in monadic second order logic (MSO).
- We develop tree-decomposition-based algorithms for ABA. Towards wider extensibility, we first give a detailed account of a dynamic programming algorithm for the stable semantics and instantiate algorithms for admissible, complete, and stable semantics in the framework of D-FLAT [1,2,9], which enables declarative specification of such algorithms in answer set programming (ASP) [13,43,52]. Together with expressing ABA reasoning in MSO, the declarative approach of D-FLAT leads to a system that has potential for adaptation to other forms of structured argumentation, further semantics, or other modes of reasoning.

– Finally, we present an experimental evaluation of a prototype using D-FLAT, showing promise of our approach for complex counting tasks involving in particular a high number of solutions.

Further material, including ASP encodings used within D-FLAT, can be found at https://gitlab.tugraz.at/krr/astra.

2 Background

We recall assumption-based argumentation (ABA) [12,20], monadic second order logic, and tree-width and tree-decompositions [10], next.

Assumption-Based Argumentation. We assume a deductive system $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a set of atoms and $\mathcal{R}$ a set of inference rules over $\mathcal{L}$. A rule $r \in \mathcal{R}$ has the form $a_0 \leftarrow a_1, \ldots, a_n$ with each $a_i \in \mathcal{L}$. We denote the head of rule r by $head(r) = a_0$ and the (possibly empty) body of r with $body(r) = \{a_1, \ldots, a_n\}$.

Definition 1. *An ABA framework is a tuple $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$, where $(\mathcal{L}, \mathcal{R})$ is a deductive system, $\mathcal{A} \subseteq \mathcal{L}$ a non-empty set of assumptions, and ${}^-$ a function mapping assumptions $\mathcal{A}$ to atoms $\mathcal{L}$.*

In words, an ABA framework includes a deductive system, a distinction between assumptions and non-assumptions, and a contrary function that assigns contraries to assumptions. In this work, we focus on the commonly used logic programming fragment of ABA [12]. We assume that all sets and rules in an ABA framework are finite, and no assumption occurs in the head of a rule. The last condition means that the ABA frameworks are flat. As a slight generalization, we allow the contrary function to be partial.

Derivability in ABA can be defined in multiple ways, we recall the so-called forward-derivations, here called simply derivations. An atom $a \in \mathcal{L}$ is derivable from a set $X \subseteq \mathcal{A}$ using rules $\mathcal{R}$, denoted by $X \vdash_{\mathcal{R}} a$, if $a \in X$ or there is a sequence of rules $(r_1, \ldots, r_n)$ such that $head(r_n) = a$ and for each rule r_i we have $r_i \in \mathcal{R}$ and each atom in the body of r_i is derived from rules earlier in the sequence or is in X, i.e., $body(r_i) \subseteq X \cup \bigcup_{j<i} \{head(r_j)\}$. The deductive closure for an assumption set X w.r.t. rules $\mathcal{R}$ is defined as $Th_{\mathcal{R}}(X) = \{a \in \mathcal{L} \mid X \vdash_{\mathcal{R}} a\}$.

Example 1. Our running example ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ is given with $\mathcal{A} = \{a, b\}$, $\mathcal{L} = \mathcal{A} \cup \{x, y, z\}$, the rules $r_1 = (x \leftarrow a)$, $r_2 = (y \leftarrow x)$, $r_3 = (z \leftarrow b)$, and contraries $\overline{a} = z$ and $\overline{b} = y$.

The contrary function is used to define attacks between assumption sets.

Definition 2. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABA framework, and $A, B \subseteq \mathcal{A}$ be two sets of assumptions. Assumption set A attacks assumption set B in F if $A' \vdash_{\mathcal{R}} \overline{b}$ for some $A' \subseteq A$ and $b \in B$.*

Conflict-free assumption sets and defense are defined, as follows.

Definition 3. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. An assumption set $A \subseteq \mathcal{A}$ is conflict-free in F iff A does not attack itself. Set A defends assumption set $B \subseteq \mathcal{A}$ in F iff for all $C \subseteq \mathcal{A}$ that attack B it holds that A attacks C.*

The ABA semantics we focus on in this work are then defined next.

Definition 4. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. Further, let $A \subseteq \mathcal{A}$ be a conflict-free set of assumptions in F. In F, set A is*

- admissible *iff A defends itself,*
- complete *iff A is admissible and contains all assumption sets defended by A,*
- preferred *iff A is admissible and there is no admissible set of assumptions B with $A \subset B$, and*
- stable *iff each $\{x\} \subseteq \mathcal{A} \setminus A$ is attacked by A.*

Reasoning tasks on ABA include verifying whether a given set of assumptions is a σ-assumption set, and enumerating or counting σ-assumption sets. In addition, often a relevant question is to find out whether a given atom is acceptable under a semantics. To answer this question, two prominent reasoning modes are credulous and skeptical acceptance of atoms in an ABA framework. A given atom $s \in \mathcal{L}$ is credulously accepted in F under semantics σ iff there is a σ-assumption set A such that $s \in Th_{\mathcal{R}}(A)$, and skeptically accepted in F under semantics σ iff $s \in Th_{\mathcal{R}}(A)$ for all σ-assumption sets A. Credulous reasoning under admissible, complete, stable, and preferred semantics is NP-complete and skeptical acceptance under stable is coNP-complete, and Π_2^P-complete under preferred semantics [21,24].

Example 2. Continuing Example 1, there are two stable assumption sets $\{a\}$ and $\{b\}$, with deductive closures $Th_{\mathcal{R}}(\{a\}) = \{a, x, y\}$ and $Th_{\mathcal{R}}(\{b\}) = \{b, z\}$, respectively. In this example, atoms x, y, and z are credulously accepted under stable semantics, and no atom is skeptically accepted under stable semantics.

Monadic Second Order Logic and Tree-Decompositions. We recap monadic second order logic and tree-decompositions, following Gottlob et al. [44].

Monadic second order logic (MSO) extends first order logic by allowing set variables, which range over sets of domain variables, and quantification over these set variables. We write individual variables as lowercase letters x and set variables as uppercase letters X. For a set $\tau = \{R_1, \ldots, R_k\}$ of predicate symbols, a finite structure I over τ, also called a τ-structure, has a finite domain $D = dom(I)$ and relations $R_i^I \subseteq D^{r_i}$ of arity r_i for each predicate symbol $R_i \in \tau$. Evaluation of an MSO formula ϕ over a τ-structure I is defined, as usual. For our purposes, it is sufficient to only consider unary and binary predicates.

A tree-decomposition of a τ-structure I is a pair $(T, (D_t)_{t \in T})$, with T being a rooted tree and each $D_t \subseteq D = dom(I)$, satisfying the following properties.

1. Every domain element $x \in D$ is part of some D_t, i.e., $x \in D_t$ for some $t \in T$.
2. For every $R_i \in \tau$ and tuple $(a_1, \ldots, a_{r_i}) \in R_i^I$ it holds that there is some node $t \in T$ with $\{a_1, \ldots, a_{r_i}\} \subseteq D_t$.
3. The set $\{t \mid a \in D_t\}$ induces a subtree of T, for each $a \in D$.

In brief terms, a tree-decomposition is a tree formed of so-called bags D_t consisting of sets of domain elements. The second condition ensures that each relation is fully part of at least one bag. The third condition, often referred to as the connectedness condition, states that whenever two bags D_t and $D_{t'}$ both contain an a, then on the path between those two bags, we encounter a in the bags.

The width of a tree-decomposition $(T, (D_t)_{t \in T})$ is the maximum number of domain elements in bags minus one, i.e., $max\{|D_t| \mid t \in T\} - 1$. The tree-width of a τ-structure I is the minimum width of all tree-decompositions of I.

Complexity-wise, MSO and tree-width are connected, as stated by Courcelle's theorem.

Theorem 1 ([17]). *Let ϕ be an MSO formula over a structure τ, and I be a τ-structure of tree-width w. It holds that evaluating ϕ over I can be achieved in $\mathcal{O}(f(|\phi|, w) \cdot |I|)$, for some function f.*

In brief, a problem expressible in MSO is then said to be fixed-parameter tractable (FPT) for the parameter tree-with of the underlying τ-structure. For an overview on parametrized complexity (including FPT), see the book by Downey and Fellows [25].

3 Complexity of ABA Under the Lens of Tree-Width

In this section, we show that a large range of problems in ABA can be addressed algorithmically via utilizing tree-width, formally by stating that these problems are FPT with the parameter tree-width.

Towards these results, we represent reasoning in ABA in MSO. We make use of the following set of predicate symbols: $\tau_{ABA} = \{atom/1, asm/1, rule/1, head/2, body/2, contrary/2, query/1\}$, together with the arities of the predicates. The intention of the predicates is formalized next.

Definition 5. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. The associated τ_{ABA} structure, denoted by I_F, is defined by $atom(x)$ for each $x \in \mathcal{L}$, $asm(a)$ for each $a \in \mathcal{A}$, $rule(r)$, $head(r, h)$, and $body(r, b_1), \ldots, body(r, b_k)$ for each rule $r = h \leftarrow b_1, \ldots, b_k$, and $contrary(a, x)$ for each $a \in \mathcal{A}$ and $x \in \mathcal{L}$ s.t. $\overline{a} = x$.*

The remaining *query* predicate is used to indicate what to query for credulous or skeptical reasoning.

Example 3. Consider again the ABA framework from Example 1, which can be written as a τ_{ABA} structure containing $atom(a)$, $atom(b)$, $atom(x)$, $atom(y)$, $atom(z)$, $asm(a)$, $asm(b)$, $rule(r_1)$, $rule(r_2)$, and $rule(r_3)$ for the unary predicates, and $contrary(a, z)$, $contrary(b, y)$, $head(r_1, x)$, $head(r_2, y)$, $head(r_3, z)$, $body(r_1, a)$, $body(r_2, x)$, and $body(r_3, b)$ for the binary predicates. Part of a tree-decomposition of this ABA framework is depicted in Fig. 1. This decomposition is a so-called "nice" tree-decomposition, which has to satisfy further constraints useful for algorithms operating on such a nice tree-decomposition. We recall the formal definition of nice tree-decompositions in the next section. In this tree-decomposition there are 22 nodes. For instance, r_1 and a are together in the bag

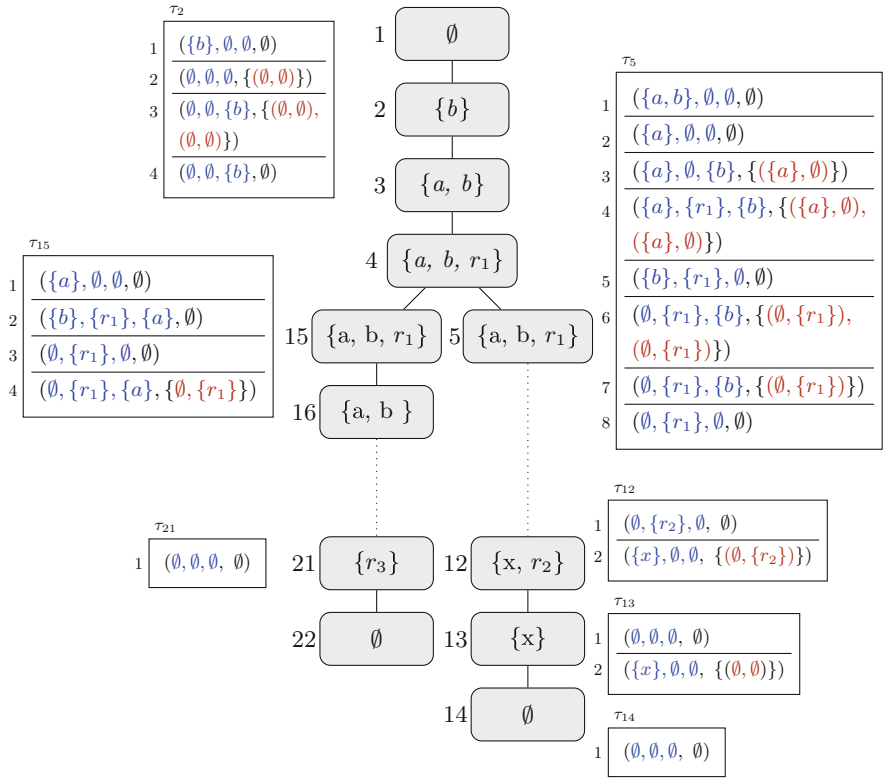

Fig. 1. Part of an example tree-decomposition of the ABA framework of Example 1. (Color figure online)

of node 15, satisfying the condition that these two have to be together in one bag because $body(r_1, a)$ holds.

We move on to expressing ABA semantics in terms of MSO. We make use of several common shortcuts, as defined next, in addition to "$x \in X$" to denote that a domain element is in the set.

$$x \notin X := \neg(x \in X) \qquad X \subseteq Y := \forall x(x \in X \rightarrow x \in Y)$$
$$X \subsetneq Y := (X \subseteq Y) \wedge \neg(Y \subseteq X) \qquad X \subseteq_{\mathcal{A}} Y := \forall x\big((x \in X \wedge asm(x)) \rightarrow x \in Y\big)$$
$$X \subsetneq_{\mathcal{A}} Y := (X \subseteq_{\mathcal{A}} Y) \wedge \neg(Y \subseteq_{\mathcal{A}} X) \qquad X =_{\mathcal{A}} Y := (X \subseteq_{\mathcal{A}} Y) \wedge (Y \subseteq_{\mathcal{A}} X)$$

First, we encode derivability. For a given ABA framework F and $A \subseteq \mathcal{A}$, there is a direct connection between $Th_{\mathcal{R}}(A)$ and the unique $\subseteq$-minimal classical model of the propositional Horn formula $\bigwedge_{a \in A} a \wedge \bigwedge_{r \in \mathcal{R}} \big(\bigwedge_{b \in body(r)} \rightarrow head(r)\big)$, that is, the Horn theory consisting of each assumption in A as facts and rules in $\mathcal{R}$ as implications. It holds that $M \subseteq \mathcal{L}$ is the $\subseteq$-minimal model of this formula iff $M = Th_{\mathcal{R}}(A)$.

We can directly make use of this fact and represent derivability in ABA by the following MSO formula, where we use quantification to express the same reasoning as in the above Horn formula. First, we define satisfaction of rules by

$$\forall r\Big(\mathit{rule}(r) \rightarrow \exists s\big((\mathit{head}(r,s) \wedge s \in E) \vee (\mathit{body}(r,s) \wedge s \notin E)\big)\Big).$$

In this formula the set variable E is open. The formula states that whenever r is a rule, then the rule has to be satisfied by E in the logical sense: either the head is in E or some body element is missing from E. We call this formula $\phi_{Sat}(E)$.

Derivability is then expressible by

$$\phi_{Th}(E) = \ \phi_{Sat}(E) \wedge \forall E'\big((E' \subsetneq E \wedge E' =_{\mathcal{A}} E) \rightarrow \neg\phi_{Sat}(E')\big),$$

which states that E should satisfy the rules and no proper subset $E' \subsetneq E$ that shares the same assumptions satisfies the rules. Then E corresponds to the least model of the above Horn formula (with assumptions stated as facts).

Attacks are expressible via $\phi_{att}(E,S) = \exists x, a\big(x \in E \wedge a \in S \wedge \mathit{contrary}(a,x)\big)$, that is, set E attacks set S if there is a contrary in E of S. The contrary only contains assumptions in the first position and atoms in the second.

The property of being conflict-free can be expressed as $\phi_{cf}(E) = \phi_{Th}(E) \wedge \neg\phi_{att}(E,E)$. The notion of defense can then be represented, as follows.

$$\phi_{def}(E,A) = \ \forall S\big((S \subseteq \mathcal{L} \wedge \phi_{Th}(S) \wedge \phi_{att}(S,A)) \rightarrow \phi_{att}(E,S)\big)$$

In words, if E defends A if for each S attacking A we find E attacks S (for S we also need to check derivability via $\phi_{Th}(S)$).

Admissibility, the complete, and preferred semantics can then be represented, as stated next.

$$\begin{aligned}
\phi_{adm}(E) &= \phi_{cf}(E) \wedge \phi_{def}(E,E)\\
\phi_{com}(E) &= \phi_{adm}(E) \wedge \forall S\big((\phi_{def}(E,S) \wedge \phi_{Th}(S)) \rightarrow S \subseteq E\big)\\
\phi_{prf}(E) &= \phi_{adm}(E) \wedge \forall E' \neg\big(E' \subseteq \mathcal{L} \wedge E \subsetneq E' \wedge \phi_{adm}(E')\big)
\end{aligned}$$

Finally, the stable semantics can be expressed by formula $\phi_{stb}(E)$, given as $\phi_{cf}(E) \wedge \forall a(\mathit{asm}(a) \rightarrow a \in E \vee \exists x\big(x \in E \wedge \mathit{contrary}(a,x)\big))$, capturing directly the definition of stable semantics.

Credulous and skeptical reasoning can then be specified by stating

$$\begin{aligned}
\phi_{\sigma}^{Cred} &= \exists E\big(E \subseteq \mathcal{L} \wedge \phi_{query}(E) \wedge \phi_{\sigma}(E)\big) \text{ and}\\
\phi_{\sigma}^{Skept} &= \forall E\Big(\big(E \subseteq \mathcal{L} \wedge \phi_{\sigma}(E)\big) \rightarrow \big(\phi_{query}(E)\big)\Big).
\end{aligned}$$

with $\phi_{query}(E) = (\forall x(\mathit{query}(x) \rightarrow x \in E))$. The formula $\phi_{query}(E)$ directly states that the atoms defined by query are in E. We tacitly assume that queries defined by *query* refer only to atoms and not to rules.

By utilizing Courcelle's results (Theorem 1), we can directly infer the following FPT result. The proof of this theorem directly follows from the previous formulas, declaratively representing the definitions of ABA, and Theorem 1.

Theorem 2. *Deciding credulous or skeptical acceptance of atoms in a given ABA framework under admissible, complete, stable, or preferred semantics is FPT w.r.t. tree-width.*

4 Dynamic Programming Algorithms for ABA

In this section, we present our approach to compute reasoning tasks in ABA using dynamic programming (DP) algorithms. Due to page limitations, we present a DP algorithm using tree-decompositions for computing stable assumption sets, and discuss changes needed for admissible and complete assumption sets. We explain how to instantiate our algorithms in the D-FLAT [1,2,9] framework, allowing for a declarative specification of the DP algorithms, and give full declarative encodings in the online supplementary material.

On a high level, DP algorithms operating on tree-decompositions of a given instance usually work bottom-up, by computing tables for each bag in post-order. The tables computed for each bag represent current partial solution candidates that can be inferred from the information encountered "so far". In a final step, partial solutions are then combined into full solutions. We delegate this step to D-FLAT, which follows so called extension pointers in a top-down fashion, and combines compatible partial solutions.

For the sake of clarity, we present our DP algorithm for stable semantics by requiring that the tree-decomposition is nice and has empty bags as leaves and as the root. Our implementation in D-FLAT does not require nice tree-decompositions, however. In nice tree-decompositions each node has a type and is either a *leaf*, the *root*, an *introduction* node, a *removal* node, or a *join* node. Except for leaves and join nodes, the nodes have exactly one child, and join nodes have exactly two children. Bags of introduction nodes have all objects of the child bag and one additional object, while bags of removal nodes have exactly one object less. Join nodes and their children have exactly the same bags. These conditions allow for a more compact algorithm representation. One can efficiently compute a nice tree-decomposition from a given tree decomposition [11].

Our algorithm is inspired by concepts for DP algorithms [35] for answer set programming [43]. For a given ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, we define a partial stable assumption set as a quadruple (I, R, D, CW), where $I \subseteq \mathcal{L}$ is called a witness, $R \subseteq \mathcal{R}$, $D \subseteq \mathcal{A}$, and CW is a set of so-called counterwitnesses, which are pairs (C, R_C) with $C \subseteq I$ and $R_C \subseteq \mathcal{R}$. We utilize the concepts of witnesses counterwitnesses, as presented by Fichte et al. [35], in our DP algorithms. Intuitively, each partial stable assumption set consists of a witness set I of atoms that is a candidate for a stable assumption set and all atoms that can be derived from the stable assumption set, while D contains the assumptions attacked ("defeated") by I. That is, I can be seen as atoms and assumptions we "assume" to be part of a stable assumption set. Since we might encounter components of rules in various places in the tree-decomposition, we view the rules as Boolean Horn clauses and store in R all rules that are satisfied by I (similar as in Sect. 3). Finally, a counterwitness represents pairs (C, R_C) where

C shares the same assumptions as in I, i.e., $I \cap \mathcal{A} = C \cap \mathcal{A}$, but has strictly fewer atoms (i.e., represents proper a subset) and R_C the associated satisfied rules. A counterwitness testifies that one can satisfy the rules R_C with fewer atoms, and, thus, is a counter to derivability of the atoms in I from the assumptions in I if both I and C satisfy all rules in the root node. During bottom-up computation, partial stable assumption sets and their counterwitnesses are modified, added, or removed, depending on the objects encountered in the bags. A partial stable assumption set contains only components of the current bag (and may use information from child bags).

Let us go over Algorithm 1 and Algorithm 2 for stable assumption sets. We show here the case for enumerating stable assumption sets, but credulous and skeptical reasoning can be achieved via small modifications: enumerating only stable assumption sets containing or not containing a query.

In Algorithm 1 we call Algorithm 2 for each node in the tree-decomposition of the given ABA instance and store the result in a table (Tab). Algorithm 2 computes these tables, given the tables of the children nodes. We store partial stable assumption sets computed in Res (initially empty). In Line 3 we merge the tables of the two children tables, by combining compatible partial stable assumption sets. Two such sets are compatible if they coincide on their first components I, i.e., for two sets τ_i and τ_j of partial stable assumptions sets, the function *merge* returns $\{(I, R_i \cup R_j, D_i \cup D_j, C_i \sqcup C_j) \mid (I, R_i, D_i, C_i) \in \tau_i, (I, R_j, D_j, C_j) \in \tau_j\}$. For merging the counterwitnesses, we use $C_i \sqcup C_j = \{(C, R \cup R') \mid (C, R) \in C_i, (C, R') \in C_j\}$. If we are not in a join node, there is only one child and we go over all previous partial stable assumption sets (loop beginning with Line 5, after extracting the single table in the line before).

For the root node, there is a simple check: if there are no counterwitnesses ($C = \emptyset$, Line 6), we have found an entry leading to stable assumption set.

Line 7 considers bags in which an atom a is introduced. In this case, the set of partial stable assumption sets for this bag is obtained by utilizing each partial stable assumption set in the child partial solution I, and creating two partial stable assumption sets I_{out} and I_{in}, the former by not adding the introduced atom a (Line 8), and the latter by adding it (Line 13). Note that the partial stable assumption set with introduced atom a is only constructed if not conflicting (Line 12). This can be seen in Fig. 1 at node 13 which introduces x, resulting in two witnesses on lines 1 and 2 of τ_{13}. In the figure, witnesses are shown in blue and counterwitnesses are in red.

Algorithm 1. Compute partial stable assumption sets on T

Require: ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ and a nice tree-decomposition T of F.
Out: Function $Tab(t)$ assigning each node T a set of partial solutions.

1: **for** t **in** post-order(T) **do**
2: $\quad childTables := \{Tab(t') \mid t'$ child of t in $T\}$
3: $\quad Tab(t) := ComputeTable_{stb}(T, t, childTables)$

The *Sat* function is used to compute satisfied rules, for a given node t, and set I of atoms by defining $Sat(t, I, R) = \{r \in R \mid head(r) \in I \text{ or } body(r) \cap bag(t) \nsubseteq I\}$, following a similar line of reasoning as in Sect. 3.

To construct new sets of counterwitnesses for each newly constructed partial stable assumption set, an analogous process occurs (Line 10 and Line 15). In the most trivial case, that of a new witness I_{out} constructed by not adding the

Algorithm 2. $ComputeTable_{stb}(t, childTables)$

Require: node t, and set of sets of partial stable assumption sets $childTables$
Out: Return partial stable assumption sets

1: $Res := \emptyset$
2: **if** $type(t) = leaf$ **then return** $\emptyset$
3: **if** $type(t) = join$ **then return** $merge_{ST}(childTables)$
4: Let $\tau \in childTables$ ▷ $childTables$ is a singleton
5: **for** $(I, R, D, C) \in \tau$ **do**
6: **if** $type(t) = root \wedge C = \emptyset$ **then** $Res := Res \cup \{(I, R, D, C)\}$
7: **if** $type(t) = intro \wedge a$ is the introduced atom
8: $I_{out} := I$, $R_{out} := R \cup Sat(t, I_{out}, Rules(t))$
9: **if** $\overline{a} \in I_{out}$ **then** $D_{out} := D \cup \{a\}$ **else** $D_{out} := D$
10: $C_o = \{(I_{Co}, R_{Co} \cup Sat(t, I_{Co}, Rules(t))) \mid (I_{Co}, R_{Co}) \in C\}$
11: $Res := Res \cup \{(I_{out}, R_{out}, D_{out}, C_o)\}$
12: **if** $\overline{a} \notin I \cup \{a\}$ ▷ Conflict-free check
13: $I_{in} := I \cup \{a\}$
14: $R_{in} := Sat(t, I_{in}, Rules(t))$ ▷ New rules can become satisfied.
15: $C_{in} := \{(I_c \cup \{a\}, Sat(t, I_c \cup \{a\}, Rules(t))) \mid (I_c, R_c) \in C\}$
16: **if** $a \notin \mathcal{A}$ **then** $C_{in} := C_{in} \cup C$
17: $Res := Res \cup \{(I_{in}, R_{in}, D, C_{in})\} \cup \{(I, Sat(t, I, Rules(t)))\}$
18: **if** $type(t) = rem \wedge a$ is the removed atom
19: **if** $a \in \mathcal{A}$
20: **if** $a \in I \vee a \in D$ ▷ Stable check: only preserve if either In or Def.
21: $R' := Sat(t, I \setminus \{a\}, Rules(t))$
22: $C' := \{(I_c \setminus \{a\}, Sat(t, I_c \setminus \{a\}, Rules(t))) \mid (I_c, R_c) \in C\}$
23: $Res := Res \cup \{(I \setminus \{a\}, R', D \setminus \{a\}, C')\}$
24: **else**
25: $R' := Sat(t, I \setminus \{a\}, Rules(t))$
26: $C' := \{(I_c \setminus \{a\}, Sat(t, I_c \setminus \{a\}, Rules(t))) \mid (I_c, R_c) \in C\}$
27: $Res := Res \cup \{(I \setminus \{a\}, R', D, C')\}$
28: **if** $type(t) = intro \wedge r$ is the introduced rule
29: $R' := R \cup Sat(t, I, \{r\})$
30: $C' := \{(I_c, R_c \cup Sat(t, I_c, \{r\})) \mid (I_c, R_c) \in C\}$
31: $Res := Res \cup \{(I, R', D, C')\}$
32: **if** $type(bag) = rem \wedge r$ is the removed rule
33: **if** $r \in R$ ▷ only keep an answer if r sat
34: $R' := R \setminus \{r\}$
35: $C' := \{(I_c, R_c \setminus \{r\}) \mid (I_c, R_c) \in C\}$
36: $Res := Res \cup \{(I, R', D, C')\}$
37: **return** Res

introduced atom a, the set C_o is given by preserving the counterwitnesses from the child partial solution, with an updated set of satisfied bag rules (Line 10).

In the case of a witness constructed by the addition of a to a child witness, if a is an introduced assumption, we require a to be part of the constructed counterwitnesses, and thus we do not preserve all the child counterwitnesses C. On the contrary, if the introduced atom is not an assumption (Line 16), we add to our set of newly created counterwitnesses C_{in} the child counterwitnesses C. In intuitive terms, for a witness constructed by the addition of an introduced atom $a \notin A$, there can be counterwitnesses, subsets of the witness, such that the introduced atom has not been derived.

To enforce the return of stable assumption sets only, Algorithm 2 ensures that when an atom is removed at node t, only those child assumption sets that were stable are preserved as partial stable assumption sets in t. This is achieved through the check for stable assumption sets in Line 20. In case an atom $a \notin \mathcal{A}$ is removed, this check is not required, and partial stable assumption sets are preserved with updated sets I and R. With respect to Fig 1, node 2 removes a, and row 2 in τ_2 is the result of preserving a witness from τ_3 (not shown) s.t. $a \in D$, and $b \notin I$, $b \notin D$.

Finally, consider the two possibilities of a bag either introducing or removing a rule r (Line 28 and Line 32 respectively). When a rule is introduced, witnesses and counterwitnesses are preserved, both with an updated set of satisfied rules accounting for the status of the introduced rule. On the contrary, when a rule is removed, by the connectedness property we have visited all atoms in the rule, hence the rule could not become satisfied elsewhere. Algorithm 2 enforces the satisfiability of the removed rule by not preserving partial stable assumption sets for which the removed rule has not been satisfied. Figure 1 depicts the case for an introduced rule in table τ_{12}, which introduces rule r_2. In this case, counterwitnesses that have been preserved, e.g., on line 2 of τ_{12}, must have an updated set of satisfied rules.

Admissibility and Complete Semantics. To verify whether an assumption set defends itself against all attacks, one can ensure that all its attackers are attacked by the set, or inversely, that there are no undefeated attackers. One can adapt Algorithm 2 as follows: (i) we track derivability for undefeated atoms, (ii) we check its correctness by ensuring that undefeated atoms are not attacked by the set of supported assumptions, i.e. by the candidate admissible set I, (iii) we track the set of atoms that are defeated, (iv) we require an assumption to be either undefeated or defeated, and (v) we add the admissibility check by not preserving those assumption sets that are attacked by some undefeated atom.

For complete semantics, one can ensure that the supported set of assumptions I includes all those assumptions that are undefeated and not attacked by some undefeated set of assumptions, i.e., those assumptions that are *defended*. In an algorithm for the complete semantics we can track an additional set AU of assumptions that are attacked by undefeated. The algorithm then avoids preserving assumption sets that do not include undefeated assumptions that are not attacked by an undefeated assumption set.

Instantiating Our DP Algorithms. D-FLAT is a problem solving framework based on the DP paradigm that was specifically developed to provide means for declarative specification of algorithms operating on tree-decompositions of given problem instances. The framework allows prototyping an algorithm to solve a given problem by means of the ASP language.

We utilize D-FLAT to instantiate our algorithms for admissible, complete, and stable assumption sets, and associated reasoning tasks, in the ASP language. In the D-FLAT workflow, one can delegate the burden of computing a tree decomposition of the problem instance, and of combining partial solutions to D-FLAT itself. In this workflow, one specifies how partial solutions look like (for stable semantics, our Algorithm 2), and how they can be validly combined, also referred to as extended in D-FLAT terms. D-FLAT then takes care of the storage and actual combination of partial solutions. More concretely, at each node of the tree decomposition, D-FLAT performs a call to an ASP solver, and computes a partial solution. Finally, D-FLAT combines partial solutions by following external pointers, which intuitively specify which partial solutions can be appropriately combined into a complete solution of the problem instance.

5 Experiments

In this section, we present an empirical evaluation of our prototype implementation using D-FLAT. The encodings, instances, and an instance generator used in our evaluation are available with the online material.

Our prototype supports enumeration and counting of σ-assumption sets, for $\sigma \in \{adm, com, stb\}$, and checking skeptical and credulous acceptance for these semantics. Due to a potential high number of assumption sets, we considered counting admissible, complete, and stable assumption sets in two modes, similar to previous works [18,48]. First, counting all σ-assumption sets and second counting all σ-assumption sets with a given atom being derivable.

We observed that previous random generation methods for ABA instances often result in instances with high tree-width. To explore the potential of our tree decomposition-based approach, we generated ABA frameworks exhibiting a (controlled) low tree-width. For general undirected graphs, $k \times n$ grids (with vertices connected only to vertical and horizontal neighbours) have a controlled tree-width of $min(k, n)$. We adapted this behavior, by constructing $k \times n$ grids of $k \cdot n$ atoms. A third of these are randomly and uniformly chosen as assumptions. For each non-assumption atom in the grid, this atom is used as a head in a predefined number *rph* (rules per head) of rules. The *rph* number is randomly picked in a restricted range (0–3). The number of body elements for each rule is picked randomly between 0 and the number of cross neighbours of an atom. The selected amount of body elements are then randomly picked from the cross neighbourhood. This process is repeated an *rph* number of times, and during the first iteration only assumptions are allowed as body elements. Contraries are generated based on the flip of a coin from the two steps cross neighbourhood of an assumption in the grid. While the contraries might lead to a higher tree-width,

we observed that the resulting tree-width is sufficiently bounded. A randomly chosen query atom is generated for each instance (uniform probability).

Our experiments were conducted on a Linux machine with 64-bit architecture, powered by an Intel i5 CPU with 8 cores and 8 GB of memory. We imposed a timeout limit of 600 s per run, and a memory limit of 8192 MB.

We compared our approach against the current state-of-the-art ASP-based approach [48,49] using Clingo [42] (version 5.4.1) as the ASP solver. Table 1 shows an overview of the results, computed over a total of 81 instances generated by four instances for each $k \in \{2, 3, 5\}$ and $n \in \{10, 20, 100, 200, 400, 500, 700\}$, excluding 3 instances that resulted in errors for D-FLAT.

The results indicate that for solving the counting tasks, the semantics plays a major role: our prototype had fewer timeouts than when using clingo when counting admissible assumption sets (the case with query denoted by appending "-q"). For stable semantics, both approaches are somewhat on-par with Clingo having an edge over the D-FLAT based approach. For complete semantics, Clingo outperforms the D-FLAT approach.

We hypothesize that the number of solutions (assumption sets) plays a major role, together with the fact that the D-FLAT encoding of stable semantics is simpler, in explaining the runtime. There are more admissible and complete assumption sets than for stable semantics, and in particular, the number of admissible assumption sets might be high.

6 Discussions

In this work we looked at complex computational tasks arising in assumption-based argumentation, and showed that many such reasoning tasks are fixed-parameter tractable w.r.t. the parameter tree-width, of a given graph representation of ABA frameworks. We showed these results by using monadic second order logic (MSO) and Courcelle's theorem. We developed DP algorithms for reasoning in ABA and implemented these in the D-FLAT framework, allowing for declarative specification of such algorithms.

Table 1. Median running time and timeouts per task (in seconds)

Task	Clingo		D-FLAT	
	Median	Timeouts	Median	Timeouts
count-adm	600.0	55	287.179	29
count-co	0.039	16	254.624	30
count-st	0.034	0	5.97	0
count-adm-q	600.0	46	98.24	27
count-co-q	0.039	11	98.39	28
count-st-q	0.034	0	5.30	2

Taken together, our MSO and D-FLAT encodings can be useful for the extension of our work to other structured argumentation formalisms: the MSO encodings suggest wide applicability of FPT results, and our D-FLAT encodings have the potential for direct adaptations to related computational problems arising in structured argumentation, e.g., for the ASPIC$^+$ formalism [51].

An interesting direction for future work is to utilize recent findings [36,47] of developing theoretical upper and lower runtime bounds by encoding problems in quantified Boolean logic, instead of using MSO, under certain constraints. These allow for showing more tight bounds in terms of running time (e..g, include also lower bounds). Moreover, decomposition-guided reductions were recently shown to be viable for problems in abstract and logic-based argumentation [34]. These reductions are guided by tree-decompositions and result in (quantified) Boolean formulas which linearly preserve tree-width. In contrast to these works, our approach uses D-FLAT, enabling ASP encodings of the DP algorithms. Our theoretical result (Theorem 2) complements existing results for abstract argumentation [27–31,34,47] and logic-based argumentation [34].

Performance of our prototype relies on the performance of D-FLAT. Recent works [23,37] show that one can specify DP algorithms using database management systems (DBMS), which give another interesting route for extending our work with a different declarative framework for specifications of DP algorithms. Developing systems for ABA based on DP algorithms using DBMS and decomposition-guided reductions to quantified Boolean logic appear as a natural next step for further evaluating strengths of these approaches, also comparing them to D-FLAT.

Our empirical results indicate strength of our approach for complex counting tasks. We believe this could also be interesting for computationally intensive tasks in probabilistic argumentation [45], where counting or weighted summation problems arise naturally. Investigating possibilities for applying our approach to probabilistic argumentation appears to be a natural avenue for future work.

Acknowledgements. This work was supported by the Austrian Science Fund (FWF) P35632.

References

1. Abseher, M., Bliem, B., Charwat, G., Dusberger, F., Hecher, M., Woltran, S.: The D-FLAT system for dynamic programming on tree decompositions. In: Fermé, E., Leite, J. (eds.) JELIA 2014. LNCS (LNAI), vol. 8761, pp. 558–572. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11558-0_39
2. Abseher, M., Bliem, B., Hecher, M., Moldovan, M., Woltran, S.: Dynamic programming on tree decompositions with D-FLAT. Künstliche Intell. **32**(2–3), 191–192 (2018)
3. Atkinson, K., et al.: Towards artificial argumentation. AI Mag. **38**(3), 25–36 (2017)
4. Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.): Handbook of Formal Argumentation. College Publications (2018)
5. Besnard, P., et al.: Introduction to structured argumentation. Argument Comput. **5**(1), 1–4 (2014)

6. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press, Cambridge (2008)
7. Besnard, P., Hunter, A.: A review of argumentation based on deductive arguments. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 9, pp. 437–484. College Publications (2018)
8. Bistarelli, S., Kotthoff, L., Santini, F., Taticchi, C.: Summary report for the third international competition on computational models of argumentation. AI Mag. **42**(3), 70–73 (2021)
9. Bliem, B., Charwat, G., Hecher, M., Woltran, S.: D-FLAT^2: Subset minimization in dynamic programming on tree decompositions made easy. Fundam. Informaticae **147**(1), 27–61 (2016)
10. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybern. **11**(1–2), 1–21 (1993)
11. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. Comput. J. **51**(3), 255–269 (2007)
12. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artif. Intell. **93**, 63–101 (1997)
13. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
14. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artif. Intell. **171**(5–6), 286–310 (2007)
15. Cerutti, F., Gaggl, S.A., Thimm, M., Wallner, J.P.: Foundations of implementations for formal argumentation. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 15, pp. 688–767. College Publications (2018)
16. Charwat, G., Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation - a survey. Artif. Intell. **220**, 28–63 (2015)
17. Courcelle, B.: Graph rewriting: an algebraic and logic approach. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pp. 193–242. Elsevier and MIT Press (1990)
18. Craven, R., Toni, F.: Argument graphs and assumption-based argumentation. Artif. Intell. **233**, 1–59 (2016)
19. Craven, R., Toni, F., Cadar, C., Hadad, A., Williams, M.: Efficient argumentation for medical decision-making. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Proceedings of the KR, pp. 598–602. AAAI Press (2012)
20. Čyras, K., Fan, X., Schulz, C., Toni, F.: Assumption-based argumentation: disputes, explanations, preferences. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 7, pp. 365–408. College Publications (2018)
21. Cyras, K., Heinrich, Q., Toni, F.: Computational complexity of flat and generic assumption-based argumentation, with and without probabilities. Artif. Intell. **293**, 103449 (2021)
22. Cyras, K., Oliveira, T., Karamlou, A., Toni, F.: Assumption-based argumentation with preferences and goals for patient-centric reasoning with interacting clinical guidelines. Argument Comput. **12**(2), 149–189 (2021)
23. Dewoprabowo, R., Fichte, J.K., Gorczyca, P.J., Hecher, M.: A practical account into counting Dung's extensions by dynamic programming. In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) LPNMR 2022. LNCS, vol. 13416, pp. 387–400. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15707-3_30

24. Dimopoulos, Y., Nebel, B., Toni, F.: On the computational complexity of assumption-based argumentation for default reasoning. Artif. Intell. **141**(1/2), 57–78 (2002)
25. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
26. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–358 (1995)
27. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. Artif. Intell. **171**(10–15), 701–729 (2007)
28. Dvořák, W., Hecher, M., König, M., Schidler, A., Szeider, S., Woltran, S.: Tractable abstract argumentation via backdoor-treewidth. In: Proceedings of the AAAI, pp. 5608–5615. AAAI Press (2022)
29. Dvořák, W., Morak, M., Nopp, C., Woltran, S.: dynPARTIX - a dynamic programming reasoner for abstract argumentation. In: Tompits, H., et al. (eds.) INAP/WLP -2011. LNCS (LNAI), vol. 7773, pp. 259–268. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41524-1_14
30. Dvořák, W., Pichler, R., Woltran, S.: Towards fixed-parameter tractable algorithms for abstract argumentation. Artif. Intell. **186**, 1–37 (2012)
31. Dvořák, W., Szeider, S., Woltran, S.: Abstract argumentation via monadic second order logic. In: Hüllermeier, E., Link, S., Fober, T., Seeger, B. (eds.) SUM 2012. LNCS (LNAI), vol. 7520, pp. 85–98. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33362-0_7
32. Dvořák, W., Dunne, P.E.: Computational problems in formal argumentation and their complexity. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 14. College Publications (2018)
33. Fan, X., Toni, F., Mocanu, A., Williams, M.: Dialogical two-agent decision making with assumption-based argumentation. In: Bazzan, A.L.C., Huhns, M.N., Lomuscio, A., Scerri, P. (eds.) Proceedings of the AAMAS, pp. 533–540. IFAAMAS/ACM (2014)
34. Fichte, J.K., Hecher, M., Mahmood, Y., Meier, A.: Decomposition-guided reductions for argumentation and treewidth. In: Zhou, Z. (ed.) Proceedings of the IJCAI, pp. 1880–1886. ijcai.org (2021)
35. Fichte, J.K., Hecher, M., Morak, M., Woltran, S.: Answer set solving with bounded treewidth revisited. In: Balduccini, M., Janhunen, T. (eds.) LPNMR 2017. LNCS (LNAI), vol. 10377, pp. 132–145. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61660-5_13
36. Fichte, J.K., Hecher, M., Pfandler, A.: Lower bounds for QBFs of bounded treewidth. In: Hermanns, H., Zhang, L., Kobayashi, N., Miller, D. (eds.) Proceedings of the LICS, pp. 410–424. ACM (2020)
37. Fichte, J.K., Hecher, M., Thier, P., Woltran, S.: Exploiting database management systems and treewidth for counting. Theory Pract. Log. Program. **22**(1), 128–157 (2022)
38. Gabbay, D., Giacomin, M., Simari, G.R., Thimm, M. (eds.): Handbook of Formal Argumentation, vol. 2. College Publications (2021)
39. Gaggl, S.A., Linsbichler, T., Maratea, M., Woltran, S.: Summary report of the second international competition on computational models of argumentation. AI Mag. **39**(4), 77–79 (2018)
40. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory Pract. Log. Program. **4**(1–2), 95–138 (2004)

41. García, A.J., Simari, G.R.: Argumentation based on logic programming. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 8, pp. 409–435. College Publications (2018)
42. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. Theory Pract. Log. Program. **19**(1), 27–82 (2019)
43. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Proceedings of the ICLP, pp. 1070–1080. MIT Press (1988)
44. Gottlob, G., Pichler, R., Wei, F.: Bounded treewidth as a key to tractability of knowledge representation and reasoning. Artif. Intell. **174**(1), 105–132 (2010)
45. Hunter, A., Polberg, S., Potyka, N., Rienstra, T., Thimm, M.: Probabilistic argumentation: a survey. In: Gabbay, D., Giacomin, M., Simari, G.R., Thimm, M. (eds.) Handbook of Formal Argumentation, vol. 2, chap. 7. College Publications (2021)
46. Lagniez, J., Lonca, E., Mailly, J., Rossit, J.: Introducing the fourth international competition on computational models of argumentation. In: Gaggl, S.A., Thimm, M., Vallati, M. (eds.) Proceedings of the SAFA. CEUR Workshop Proceedings, vol. 2672, pp. 80–85. CEUR-WS.org (2020)
47. Lampis, M., Mengel, S., Mitsou, V.: QBF as an alternative to Courcelle's theorem. In: Beyersdorff, O., Wintersteiger, C.M. (eds.) SAT 2018. LNCS, vol. 10929, pp. 235–252. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94144-8_15
48. Lehtonen, T., Wallner, J.P., Järvisalo, M.: Declarative algorithms and complexity results for assumption-based argumentation. J. Artif. Intell. Res. **71**, 265–318 (2021)
49. Lehtonen, T., Wallner, J.P., Järvisalo, M.: An answer set programming approach to argumentative reasoning in the ASPIC+ framework. In: Calvanese, D., Erdem, E., Thielscher, M. (eds.) Proceedings of the KR, pp. 636–646 (2020)
50. Modgil, S., Prakken, H.: A general account of argumentation with preferences. Artif. Intell. **195**, 361–397 (2013)
51. Modgil, S., Prakken, H.: Abstract rule-based argumentation. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 6, pp. 287–364. College Publications (2018)
52. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. **25**(3–4), 241–273 (1999)
53. Thimm, M., Villata, S.: The first international competition on computational models of argumentation: results and analysis. Artif. Intell. **252**, 267–294 (2017)