Structure-Aware Encodings of Argumentation Properties for Clique-width

Yasir Mahmood[1], **Markus Hecher**[2], **Johanna Groven**[3], **Johannes K. Fichte**[3]

[1]Data Science Group, Heinz Nixdorf Institute, Paderborn University, Germany
[2]CNRS, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), University of Artois, France
[3]Linköping University, Sweden
yasir.mahmood@uni-paderborn.de, hecher@cril.fr, johanna.groven@liu.se, johannes.fichte@liu.se

Abstract

Structural measures of graphs, such as treewidth, are central tools in computational complexity resulting in efficient algorithms when exploiting the parameter. It is even known that modern SAT solvers work efficiently on instances of small treewidth. Since these solvers are widely applied, research interests in compact encodings into (Q)SAT for solving and to understand encoding limitations. Even more general is the graph parameter clique-width, which unlike treewidth can be small for dense graphs. Although algorithms are available for clique-width, little is known about encodings.

We initiate the quest to understand encoding capabilities with clique-width by considering *abstract argumentation*, which is a robust framework for reasoning with conflicting arguments. It is based on directed graphs and asks for computationally challenging properties, making it a natural candidate to study computational properties. We design novel reductions from argumentation problems to (Q)SAT. Our reductions linearly preserve the clique-width, resulting in directed decomposition-guided (DDG) reductions. We establish novel results for all argumentation semantics, including counting. Notably, the overhead caused by our DDG reductions cannot be significantly improved under reasonable assumptions.

Introduction

Many problems in combinatorics, symbolic AI, and knowledge representation and reasoning are computationally very hard (Eiter and Gottlob 1993; Truszczynski 2011; Dvořák 2012). In the literature, various structural restrictions have been identified under which problems become tractable (Niedermeier 2006; Downey and Fellows 2013; Cygan et al. 2015). In theory, we are interested whether an efficient algorithm exists (Courcelle 1990; Borie, Parker, and Tovey 1992; Fischer, Makowsky, and Ravve 2008; Courcelle 2018). In practice, we seek effective practical algorithms (Lampis, Mengel, and Mitsou 2018; Järvisalo, Lehtonen, and Niskanen 2025). From both perspectives, we want to understand the structural combinatorial core of the problem. Structure preserving reductions to other problems support this understanding by its algorithmic and logic-based definability character. Important structural properties of input instances are for example hierarchical graph decompositions, which are quite interesting for algorithmic purposes and for solving problems in polynomial-time in the input size and exponential in a parameter defined on the decomposition, e.g., treewidth (Freuder 1985; Dechter 1999).

Even more general than treewidth is the graph parameter *clique-width* (Courcelle, Engelfriet, and Rozenberg 1993), which can even be small for dense graphs where treewidth is large, measuring the distance from co-graphs (Courcelle and Olariu 2000). Dynamic programming algorithms for deciding acceptance under preferred semantics are known for clique-width (Dvorák, Szeider, and Woltran 2010), but little is known about structure guided reductions to other problems. This is particularly interesting when reasoning relies on tools such as SAT solving where research increasingly asks for efficient encodings and theoretical limitations (Heule et al. 2023) under the light that logic-based characterizations are known that allow to solve SAT faster.

Proposition 1 (Fischer, Makowsky, and Ravve, 2008)**.** *For Boolean formulas of directed incidence clique-width k and size n, counting SAT can be solved in time $2^{\mathcal{O}(k)} \cdot poly(n)$.*

The proposition immediately yields the natural research question: *Can we encode problems into (Q)SAT while preserving the clique-width?* Indeed, such encodings would allow to easily reuse Proposition 1 for solving. In this paper, we address this question for abstract argumentation, a natural knowledge representation framework widely used for reasoning with conflicting arguments (Dung 1995; Rahwan 2007). There, relationships between arguments are specified in directed graphs, so-called argumentation frameworks (AFs), and conditions are placed on sets (extensions) of arguments that allow AFs to be evaluated. The computational complexity of argumentation is well-studied (Dvorák, Szeider, and Woltran 2010; Dvořák, Pichler, and Woltran 2012; Dvořák 2012; Charwat et al. 2015; Fichte, Hecher, and Meier 2024). Since AFs are already given as direct graphs and the complexity is often even beyond NP, it makes it a perfect candidate to study clique-width aware encoding.

Our main contributions are as follows:

1. We design reductions from argumentation problems to satisfiability of (quantified) Boolean formulas. Our reductions are *directed decomposition-guided (DDG)*, employ k-expressions, and linearly preserve clique-width.

	$\mathsf{s}_\sigma(\mathsf{c}^\star_\sigma)/\#_\sigma$	
$\sigma \in$	$\{\mathsf{stab}, \mathsf{adm}, \mathsf{comp}\}$	$\{\mathsf{pref}, \mathsf{semiSt}, \mathsf{stage}\}$
CW-Aw.	$\mathcal{O}(k)$	$\mathcal{O}(k)$
CW-LB (ETH)	$\Omega(k)$	$\Omega(k)$
Rt (UB/LB)	$2^{\theta(k)} \cdot poly(n)$	$2^{2^{\theta(k)}} \cdot poly(n)$
Ref. (UB)	Thm 7–12	Thm 14–18
Ref. (LB)	Thm. 20/Corr. 21	Prop. 22

Table 1: Overview of our results, where $k = \mathsf{dcw}(\mathcal{G}_i^d(F))$, and $n = |A|$ for given AF $F = (A, R)$. We use c_σ for credulous acceptance, s_σ for skeptical acceptance, and $\#_\sigma$ is the extension counting problem. "CW-Aw." refers to the clique-width increase caused by DDG reductions. "CW-LB (ETH)" refers to clique-width lower bounds of DDG reductions under ETH, "Rt (UB/LB)" are runtime upper and ETH lower bounds. *: Results for $\mathsf{c}_{\mathsf{stab}}$ also apply to $\mathsf{exist}_{\mathsf{stab}}$ whereas exist_σ is trivial for all other semantics. Finally, $\mathsf{c}_{\mathsf{pref}}$ can be solved faster via $\mathsf{c}_{\mathsf{adm}}$ and therefore shares the same bounds.

2. For *all* common *argumentation semantics*, we establish favorable upper bounds (tractability) for *extensions existence*, *argument acceptance*, and *counting*. This also works for the maximization-based (second-level) semantics and demonstrates the flexibility of our approach. For these, we rely on an auxiliary result that establishes how one can convert mixed normal-form QBF matrices into DNF and CNF (for inner-most $\forall$ and $\exists$), respectively.
3. We show that the overhead caused by our DDG reductions cannot be significantly improved under reasonable assumptions providing *structurally optimal reductions*. Indeed, this already holds for skeptical reasoning and then immediately carries over to the counting problem.

Related Works. Abstract argumentation is widely studied in knowledge representation and reasoning (Dung 1995; Rahwan 2007; Amgoud and Prade 2009; Rago, Cocarascu, and Toni 2018). The computational complexity depends on the considered semantics and common decision tasks range between polynomial-time and the second level of the polynomial hierarchy. For example, deciding whether a given argument belongs to some extension (credulous acceptance) is **NP**-complete for stable semantics and $\boldsymbol{\Sigma}_2^p$-complete for the semi-stable semantics (Dunne and Bench-Capon 2002; Dvořák and Woltran 2010; Dvořák 2012). Counting complexity in abstract argumentation is well-studied (Baroni, Dunne, and Giacomo 2010; Fichte, Hecher, and Meier 2024) with complexity reaching $\# \cdot \mathbf{P}$ (admissible, complete, stable) and $\# \cdot \mathrm{co}\mathbf{NP}$ (preferred, semi-stable, stage) for extension counting and $\# \cdot \mathbf{NP}$ (admissible, complete, stable) and $\# \cdot \boldsymbol{\Sigma}_2^p$ (preferred, semi-stable, stage) for projected counting. For treewidth, many results in abstract argumentation (Dvořák, Pichler, and Woltran 2012), including and decomposition-guided reductions (Hecher 2020; Fichte et al. 2021), are known. For clique-width, Dvorák, Szeider, and Woltran (2010) established dynamic programming algorithms and tractability results for acceptability under preferred semantics. SAT encodings are commonly used for solving argumentation problems (Niskanen and Järvisalo 2020) and QBF encodings enable tight computational bounds (Lampis, Mengel, and Mitsou 2018; Fichte, Hecher, and Pfandler 2020). In propositional satisfiability, tractability results for directed incidence clique-width (Fischer, Makowsky, and Ravve 2008), modular treewidth (Paulusma, Slivovsky, and Szeider 2016), and symmetric incidence clique-width (Slivovsky and Szeider 2013) and hardness results for undirected clique-width (Fischer, Makowsky, and Ravve 2008) exist. Tractability results for validity of QBFs are also known for incidence treewidth and directed clique-width (Capelli and Mengel 2019).

Preliminaries

We assume that the reader is familiar with standard terminology in Boolean logic (Biere et al. 2021), computational complexity (Papadimitriou 1994), and parameterized complexity (Cygan et al. 2015). For an integer k, let $[k] := \{1, \ldots, k\}$ and $A \uplus B$ be the union over disjoint sets A, B.

Satisfiability. A literal is a (Boolean) variable x or its negation $\neg x$. A *clause* or *cube (also known as term)* is a finite set of literals, interpreted as the disjunction or conjunction of these literals, respectively. A *CNF formula* or *DNF formula* is a finite set of clauses, interpreted as the conjunction or disjunction of its clauses or cubes. Sometimes we say formula to refer to a CNF formula. We use the usual convention that an empty conjunction corresponds to $\top$ and an empty disjunction to $\bot$. Let φ be a formula. For a clause $c \in \varphi$, we let $\mathsf{var}(c)$ consist of all variables that occur in c and $\mathsf{var}(\varphi) := \bigcup_{c \in \varphi} \mathsf{var}(c)$. An *assignment* is a mapping $\alpha : \mathsf{var}(\varphi) \to \{0, 1\}$, α is total if it maps all variables in φ. For $x \in \mathsf{var}(\varphi)$, we define $\alpha(\neg x) := 1 - \alpha(x)$. The CNF formula φ *under the assignment* $\alpha \in 2^{\mathsf{var}(\varphi)}$ is the formula $\varphi_{|\alpha}$ obtained from φ by removing all clauses c containing a literal set to 1 by α and removing from the remaining clauses all literals set to 0 by α. An assignment α is *satisfying* if $\varphi_{|\alpha} = \emptyset$ and φ is *satisfiable* if there is a satisfying assignment α. SAT asks to decide satisfiability of φ and #SAT asks for its number of total satisfying assignments.

Computational Complexity. For integer $i \geq 0$, $\exp(i, p)$ means a tower of exponentials, i.e., $\exp(i-1, 2^p)$ if $i > 0$ and p if $i = 0$. We assume that $poly(n)$ is any polynomial for given positive integer n. The Exponential Time Hypothesis (ETH) (Impagliazzo, Paturi, and Zane 2001) is a widely accepted standard hypothesis in the fields of exact and parameterized algorithms. 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), where $|\varphi|$ refers to the number of variables and $\|\varphi\|$ to the *size* of φ, which is number of variables and clauses in φ.

Abstract Argumentation. We use the argumentation terminology by Dung (1995) and consider non-empty, finite sets A of arguments. An *argumentation framework (AF)* is a directed graph $F = (A, R)$ where A is a set of elements, called *arguments*, and $R \subseteq A \times A$, a set of pairs of arguments representing direct attacks of arguments. An argument $s \in A$ is called *defended by* S if for every $(s', s) \in R$, there exists $s'' \in S$ such that $(s'', s') \in R$.

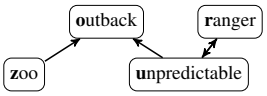

Figure 1: An example framework for deciding between going to see kangaroos in a zoo or in the outback.

The family $\mathrm{def}_F(S)$ is defined by $\mathrm{def}_F(S) := \{s \mid s \in A, s \text{ is defended by } S \text{ in } F\}$. 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. We say $S \subseteq A$ is *conflict-free* if $(S \times S) \cap R = \emptyset$; S is *admissible* if (i) S is *conflict-free*, and (ii) every $s \in S$ is *defended by* S. Assume an admissible set S. Then, (iiia) S is *complete* if $\mathrm{def}_F(S) = S$; (iiib) S is *preferred*, if there is no $S' \supset S$ that is *admissible*; (iiic) S is *semi-stable* if there is no admissible set $S' \subseteq A$ with $S'^+_R \subsetneq (S')^+_R$ where $S^+_R := S \cup \{a \mid (b,a) \in R, b \in S\}$; (iiid) S is *stable* if every $s \in A \setminus S$ is *attacked* by some $s' \in S$. A conflict-free set S is *stage* if there is no conflict-free set $S' \subseteq A$ with $S^+_R \subsetneq (S')^+_R$. We denote semantics by acronyms $\mathsf{adm}, \mathsf{comp}, \mathsf{pref}, \mathsf{semiSt}, \mathsf{stab}$, and stage, respectively. For a semantics $\sigma \in \{\mathsf{adm}, \mathsf{comp}, \mathsf{pref}, \mathsf{semiSt}, \mathsf{stab}, \mathsf{stage}\}$, $\sigma(F)$ is the set of *all extensions* of semantics σ in F. Given an AF $F = (A, R)$. Problem exist_σ asks whether $\sigma(F) \neq \emptyset$; $\#_\sigma$ asks for $|\sigma(F)|$; additionally given argument $a \in A$, *credulous acceptance* c_σ asks whether $a \in \bigcup_{e \in \sigma(F)} e$; and *skeptical acceptance* s_σ asks whether $a \in \bigcap_{e \in \sigma(F)} e$.

Example 2. *Consider an AF F with 4 arguments as depicted in Figure 1 arguing about watching kangaroos. Watching kangaroos in a "zoo" gives you all the excitement without needing to go the "outback". To observe them naturally, you need to see them in the "outback". However, kangaroos are "unpredictable" and can be dangerous. Regardless, you can go for a tour with a "ranger" making it safe to observe kangaroos in the wild, so it is not that dangerous.* ◁

Incidence Graphs and Clique-width. We follow standard terminology for graphs and directed (multi) graphs (Bondy and Murty 2008). The *directed incidence graph* $\mathcal{G}^d_i(\varphi)$ of CNF (DNF) formula φ (Ordyniak, Paulusma, and Szeider 2013) is a bipartite graph with the variables and clauses (terms) of φ as vertices and a directed edge between those, indicating whether variables occur positively or negatively in a clause (term), respectively. The *incidence graph* $\mathcal{G}_i(\varphi)$ of φ omits edge directions from $\mathcal{G}^d_i(\varphi)$. We use standard definitions for clique-width (Courcelle, Engelfriet, and Rozenberg 1993; Courcelle 1993). Intuitively, treewidth (Bodlaender 2006) measures the distance of a graph from being a tree and clique-width measures the distance of a graph to a co-graph. Clique-width is bounded by treewidth, see (Corneil and Rotics 2005). A *co-graph* is defined as follows: (i) a graph with one vertex is a co-graph; for two co-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, (iia) the disjoint union $G_1 \oplus G_2 := (V_1 \cup V_2, E_1 \cup E_2)$ is a co-graph; and (iib) the disjoint sum $G_1 \times G_2 := (V_1 \cup V_2, E_1 \cup E_2 \cup \{\{u, v\} \mid u \in V_1, v \in V_2\})$ is a co-graph.

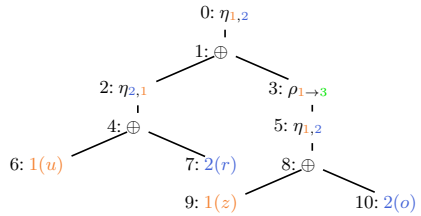

Figure 2: A k-expression of F (directed graph) from Example 2, illustrated as parse tree. We color the labels in the figure to distinguish between operation numbers and colors.

This lifts to k-graphs where k is a positive integer with k labels, called *colors*. The labeling of a graph $G = (V, E)$ is a function $\lambda : V \to [k]$ and a *k-graph* is a graph whose vertices are labeled by integers from $[k]$. An *initial k-graph* consists of exactly one vertex v colored by $c \in [k]$, denoted by $c(v)$, e.g., $1(v)$ is a shorthand for G that is a vertex v of color 1. Now, we can construct a graph G from initial k-graphs by repeatedly applying the following three *operations*. (i) *Disjoint union*, denoted by $\oplus$; (ii) *Relabeling*: changing all colors c to c', denoted by $\rho_{c\to c'}$; (iii) *Edge introduce*: connecting all vertices colored by c with all vertices colored by c', denoted by $\eta_{c,c'}$ or $\eta_{c',c}$; already existing edges are not doubled. A construction of a k-graph G using these operations can be represented by a *k-expression*, which is an algebraic term composed of $c(v)$, $\oplus$, $\rho_{c\to c'}$, and $\eta_{c,c'}$ where $c, c' \in [k]$ and v is a vertex. To construct directed graphs, $\eta_{c,c'}$ is introducing directed edges (from c to c').

We describe a k-expression by a parse-tree $T = (V_T, E_T)$ and use *parse-tree of width k* synonymously. We refer by $\mathsf{chldn}(b)$ to the *set of children* of a node b in V_T. We define $\mathsf{cols} : V_T \to 2^{[k]}$ that yields the *set of colors* in the graph $G = (V, E)$ constructed *up to operation* b. Additionally, $\mathrm{col} : (V \times V_T) \to [k]$ gives the *color of a vertex in G up to operation b*. The *last operation* is rt for *root*.

The *clique-width* $\mathsf{cw}(G)$ of an *undirected graph* is the smallest k such that G is definable by a k-expression. The *directed clique-width* $\mathsf{dcw}(G)$ of a *directed graph* is the smallest k such that G is definable by a k-expression. The *directed incidence clique-width* of a formula φ is $\mathsf{dcw}(\mathcal{G}^d_i(\varphi))$.

Example 3 (Cont.)**.** *Consider our argumentation framework F from Example 2. The clique-width of F is ≤ 3, since only three colors are needed to draw this AF. We illustrate this in Figure 2 where concrete operations are labeled by numbers $\{0, \ldots, 10\}$ for ease and 0 refers to the root.* ◁

Proposition 4 (Fischer, Makowsky, and Ravve, 2008)**.** *For any Boolean formula φ, $\mathsf{cw}(\mathcal{G}_i(\varphi)) \leq 2 \cdot \mathsf{dcw}(\mathcal{G}^d_i(\varphi))$.*

There is also a QSAT version of Proposition 1.

Proposition 5 (cf. Capelli and Mengel, 2019)**.** *For QBFs of directed incidence clique-width k, quantifier depth ℓ, and size n, counting QSAT (on free variables) can be solved in time $\mathsf{exp}(\ell + 1, \mathcal{O}(k)) \cdot poly(n)$.*

For fixed k and an undirected graph G, one can find an $f(k)$-expression of clique-width k in polynomial time (Oum and Seymour 2006); similarly for directed graphs (Kanté 2007).

K-Expression-Aware Encodings

We proceed towards reducing from argumentation problems to satisfiabilty of (quantified) Boolean formulas by a *directed decomposition-guided (DDG)* reduction that employs k-expressions. To this aim, let $F = (A, R)$ be an abstract argumentation framework from which we construct a QBF. There, we use a variable e_a for every argument $a \in A$ to store whether a is in the extension or not. We also use auxiliary variables to cover the state of attacks. In our encoding, we employ the structure and need to take care in particular of the following: (i) *initial color of arguments*, which is where we can decide whether the (single) color is defeated by being in the extension; (ii) *merge of colors*, which might occur during disjoint union or relabeling; and (iii) *edge introduce*, which occurs when drawing edges from color c' to c.

Basic Semantics

We start by presenting reductions for semantics whose classical complexity is located on the first level of the polynomial hierarchy, i.e., stable, admissible, complete extensions. Full proof details can be found in the extended version (Mahmood et al. 2025). In the following, we assume a k-expression that defines the considered argumentation framework (directed graph), while requiring only k different colors. We guide our encoding along the k-expression.

Stable Extensions. Here, we require conflict-freeness and that all arguments, which are not in the extension, are attacked by the extension. We encode this via a set E of *extension variables*. In the initial operation $c(a)$ for argument a of color c, the variable e_a encodes whether argument a is in the extension. Then, variable e^b_c encodes whether c contains an extension variable in operation b, which we refer to by *extension color*. To use the fact that colors have extension members, we guide the information along the k-expression. Finally, we ensure conflict-freeness of extension arguments in the (directed) edge introduction operations. Now, we construct our encoding formally.

$$e^b_c \leftrightarrow e_a \qquad \text{initial } b\text{, create } a \text{ of color } c \tag{1}$$

$$e^b_c \leftrightarrow \bigvee_{b' \in \mathsf{chldn}(b): c \in \mathsf{cols}(b')} e^{b'}_c \qquad \text{disjoint union } b\text{, every } c \in \mathsf{cols}(b) \tag{2}$$

$$e^b_c \leftrightarrow \bigvee_{\text{if } b \text{ relabeling } c' \mapsto c} e^{b'}_{c'} \vee \bigvee_{\text{if } c \in \mathsf{cols}(b')} e^{b'}_c \qquad \text{relabeling } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{3}$$

$$e^b_c \leftrightarrow e^{b'}_c\text{,} \quad \bigvee_{\text{if } b \text{ edge introduce } (c',c)} \neg e^b_c \vee \neg e^b_{c'} \qquad \text{edge intr. } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{4}$$

Note that the child b' in Eq. (3) is unique. However, c could be a fresh color only introduced in b and thus the disjunction would be empty (a case that is also covered here). Here, equations (1)–(4) suffice to model conflict-freeness, yielding formula $\varphi_{\#\mathsf{conf}} = \mathcal{R}_{\#\mathsf{conf}\to\#\mathrm{SAT}}(F, \mathcal{X})$. As stable extensions also attack non-extension arguments, we require for every color c that is used in an operation b an auxiliary variable d^b_c that indicates whether *every non-extension argument* of color c up to operation b is *attacked by the extension*. This is achieved via a set D of *defeated* variables, guided along the k-expression.

$$d^b_c \leftrightarrow e^b_c \qquad \text{initial } b\text{, every } c \in \mathsf{cols}(b) \tag{5}$$

$$d^b_c \leftrightarrow \bigwedge_{b' \in \mathsf{chldn}(b): c \in \mathsf{cols}(b')} d^{b'}_c \qquad \text{disjoint union } b\text{, every } c \in \mathsf{cols}(b) \tag{6}$$

$$d^b_c \leftrightarrow \bigwedge_{\text{if } b \text{ relabeling } c' \mapsto c} d^{b'}_{c'} \wedge \bigwedge_{\text{if } c \in \mathsf{cols}(b')} d^{b'}_c \qquad \text{relabeling } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{7}$$

$$d^b_c \leftrightarrow d^{b'}_c \vee \bigvee_{\text{if } b \text{ edge introduce } (c',c)} e^b_{c'} \qquad \text{edge introduce } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{8}$$

In the root rt, we ensure that all colors are defeated.

$$d^{rt}_c \qquad \text{for root colors } c \in \mathsf{cols}(rt) \tag{9}$$

Intuition. In the initial k-graphs $b = c(a)$, we remember whether c is an extension color, this information is guided along the k-expression via other formulas. The *disjunctive* encoding, in Formulas (1)–(4), guarantees that c is an extension color if it is an extension color in some operation b. Then, Formula (1) allows to retrieve arguments from extension colors and Formula (4) requires conflict-freeness of those arguments. Moreover, given an extension color, the initial graphs uniquely determine arguments in an extension. Finally, Formulas (5)–(8) encode whether each argument is either in the extension, or attacked by it. This is again encoded via colors, where a *conjunctive* encoding is used to enforce that each argument of a color has been considered.

Example 6 (cont.). *Consider F from Example 2 and the operations corresponding to the parse tree from Figure 2. For operation* 9*, which creates the initial graph* $1(z)$*, we add* $e^9_1 \leftrightarrow e_z$ *by Formula (1) and* $d^9_1 \leftrightarrow e^9_1$ *by Formula (5). For operation* 8 (disjoint union)*, we add the formulas* $e^8_1 \leftrightarrow e^9_1$*,* $e^8_2 \leftrightarrow e^{10}_2$ *by Formulas (2) and* $d^8_1 \leftrightarrow d^9_1$*,* $d^8_2 \leftrightarrow d^{10}_2$ *by Formulas (6). For operation* 5 (edge-introduce)*, we obtain* $e^5_1 \leftrightarrow e^8_1$*,* $e^5_2 \leftrightarrow e^8_2$*, and* $\neg e^5_2 \vee \neg e^5_1$ *by Formulas (4) and* $d^8_1 \leftrightarrow d^9_1$*,* $d^8_2 \leftrightarrow d^{10}_2$ *by Formulas (8). For operation* 3 (relabeling)*, we obtain,* $e^3_2 \leftrightarrow e^5_2$*,* $e^3_3 \leftrightarrow e^5_1$ *by Formulas (3) and* $d^3_2 \leftrightarrow e^5_2$*,* $d^3_3 \leftrightarrow e^5_1$ *by Formulas (7). Rest formulas are analogous.* ◁

From now on, we denote our reductions by *directed decomposing guided (DDG) reduction* $\mathcal{R}_{\#\sigma\to\#\mathrm{SAT}}$ and use this notion also for other semantics σ. Moreover, we denote the resulting (#)SAT-instance by $\varphi_{\#\sigma}$. First, we prove the correctness of our reduction for stable semantics.

Theorem 7 (⋆,Correctness)**.** *Let $F = (A, R)$ be an AF and $\mathcal{X}$ be a k-expression of F. Then, the DDG reduction $\mathcal{R}_{\#\mathsf{stab}\to\#\mathrm{SAT}}$ is correct, that is, $\#\mathsf{stab}$ on F coincides with $\#\mathrm{SAT}$ on $\mathcal{R}_{\#\mathsf{stab}\to\#\mathrm{SAT}}(F, \mathcal{X})$.*

Proof (Sketch). We establish a bijective correspondence between stable extensions of F and satisfying assignments of $\varphi_{\#\mathsf{stab}}$. In the forward direction: we construct a unique satisfying assignment α from a given stable extension S of F. This is achieved by setting the truth values for extension variables E according to S, i.e., $\alpha(e_a) = 1$ iff $a \in S$. Moreover, we set the value of $\alpha(e^b_c)$ according to the color c and operation b in the k-expression to simulate the propagation of the evaluation along the parse-tree. Then we repeat the same construction for defeated variables D. Our

construction of α from S ensures that $\alpha \models \varphi_{\#\mathsf{stab}}$. To prove the uniqueness of the assignment: we observe that S uniquely determines the evaluation of α for variables $e_a \in E$, whereas the value of α for remaining variables is simply propagated due to the nature of formulas in $\varphi_{\#\mathsf{stab}}$.

In the reverse direction: we construct a unique stable extension S by considering a satisfying assignment α for $\varphi_{\#\mathsf{stab}}$. Once again, S is obtained via the extension variables in E by letting S contain an argument a iff $\alpha(e_a) = 1$. Then, it follows from Formulas 4 and 9 that S is stable. □

Next, we establish that our encodings linearly preserve the clique-width. That is, reducing an AF instance to a SAT-instance increases the clique-width only linearly. Moreover, the size of the obtained formula grows with the size of the input decomposition and a cw-expression for the SAT instance does not have to be computed from scratch, given a cw-expression for the AF.

Theorem 8 (⋆,CW-Awareness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression of F. The DDG reduction $\mathcal{R}_{\#\mathsf{stab}\to\#\mathrm{SAT}}(F, \mathcal{X})$ constructs a SAT instance ψ that linearly preserves the directed clique-width, i.e., $\mathsf{dcw}(\mathcal{G}_i^d(\psi)) \in \mathcal{O}(k)$.*

Proof (Sketch). Given an AF F and a k-expression $\mathcal{X}$ of F, we construct a k-expression $\mathcal{X}'$ of $\mathcal{G}_i^d(\psi)$ as follows. We first specify additional colors. For each color c, we need an *extension-version* e_c and a *defeat-version* d_c of c. Then, we additionally need two more copies for each of these versions, called *child-versions* ec_c and dc_c to add clauses corresponding to the child-operation b' of an operation b and *expired-versions* ex_c and dx_c to simulate the effect that all the edges between certain variables and their clauses have been added. This is required, since otherwise, we will keep adding edges from the child-versions in future, which is undesirable. Finally, to handle both positive and negative literals in clauses, we duplicate current and the child versions of each colors for $x \in \{e_c, ec_c, d_c, dc_c\}$, denoted as x^+ for positive and x^- for negative literals. For clauses, we use two additional colors, called *clause-making* (cm) and *clause-ready* (cr) to add edges between clauses and their respective literals, since we are in the setting of incidence graphs. Intuitively, when adding edges between clauses and their respective variables, we initiate a clause C to be of color cm and its literals x to be of the appropriate color x^+ or x^-. Then, we draw directed edges between C and its literals, and later relabel C to cr to avoid any further edges to/from C. Similarly, when going from an operation b' to its parent b, we change the labels of extension and defeated variables in b' to their child-versions. We conclude by observing that the above mentioned additional colors suffice to construct a k-expression $\mathcal{X}'$ of $\mathcal{G}_i^d(\psi)$, leading to a requirement of $11k + 2$ colors. □

One might be tempted to consider AFs as undirected graphs. However, this would not work as outlined next. Two AFs could be "same" considering undirected edges (or, symmetric edges) and can thus be constructed using the same labels. However, one cannot insert directed edges using the labels for undirected edges. Thus, information is lost when encoding only undirected edges, even for stable semantics. Importantly, one can construct AFs (Mahmood et al. 2025) where directed and undirected clique-width differ and such that both AFs admit different number of extensions.

Admissible Extensions. For admissible extensions, we require conflict-freeness and that every attack from outside is defeated. Therefore, we reuse Equations (1)–(4) to model conflict-free extensions. Furthermore, we need to maintain attacks between colors containing extension arguments. This is achieved via *attack variables* T, specified as follows.

$$\neg a_c^b \qquad \text{initial } b, \text{ every } c \in \mathsf{cols}(b) \tag{10}$$

$$a_c^b \leftrightarrow \bigvee_{b' \in \mathsf{chldn}(b): c \in \mathsf{cols}(b')} a_c^{b'} \qquad \text{disjoint union } b, \text{ every } c \in \mathsf{cols}(b) \tag{11}$$

$$a_c^b \leftrightarrow \bigvee_{\text{if } b \text{ relabeling } c' \mapsto c} a_{c'}^{b'} \vee \bigvee_{\text{if } c \in \mathsf{cols}(b')} a_c^{b'} \qquad \text{relabeling } b, \text{ every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{12}$$

$$a_c^b \leftrightarrow a_c^{b'} \vee \bigvee_{\text{if } b \text{ edge introduce } (c,c')} e_{c'}^b \qquad \text{edge introduce } b, \text{ every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b), \text{not introd. } (c', c) \tag{13}$$

$$a_c^b \leftrightarrow a_c^{b'} \wedge \neg e_{c'}^b \qquad \text{edge introduce } b, \text{ every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b), \text{introd. } (c', c) \tag{14}$$

In the root rt we ensure that no color remains attacking.

$$\neg a_c^{rt} \qquad \text{for root colors } c \in \mathsf{cols}(rt) \tag{15}$$

Intuition. The intuition behind this encoding is to remember whether some non-extension argument attacks an extension argument. To this aim, Formulas (10) initiate that no argument attacks an extension to begin with. Then, Formulas (11)–(12) guide this information along the k expression. Our disjunctive encoding via colors enforces that each argument of a particular color has been considered. Formulas (13)–(14) update the status of attacking argument depending on whether an argument of color c attacks an argument of the extension color c'. Finally, Formula (15) forces that no color remains attacking to our extension.

Theorem 9 (⋆,Correctness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression of F. Then, the DDG reduction $\mathcal{R}_{\#\mathsf{adm}\to\#\mathrm{SAT}}$ is correct, that is, $\#\mathsf{adm}$ on F coincides with $\#\mathrm{SAT}$ on $\mathcal{R}_{\#\mathsf{adm}\to\#\mathrm{SAT}}(F, \mathcal{X})$.*

Theorem 10 (⋆,CW-Awareness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression of F. The DDG reduction $\mathcal{R}_{\#\mathsf{adm}\to\#\mathrm{SAT}}(F, \mathcal{X})$ constructs a SAT instance ψ that linearly preserves the width, i.e., $\mathsf{dcw}(\mathcal{G}_i^d(\psi)) \in \mathcal{O}(k)$.*

Complete Extensions. For complete semantics, we compute admissible extensions such that there is no argument that could have been included. That is, there is no argument that is not in the extension but defended by it. Moreover, we additionally need the information of whether a color has been defeated or not. To achieve this, we reuse Equations (1)–(4), (10)–(14) and, on top, we compute whether we could have included an argument via a collection O of *out variables*. These are encoded and propagated as follows.

$$o_c^b \leftrightarrow \neg e_c^b \qquad \text{initial } b, \text{ every } c \in \mathsf{cols}(b) \tag{16}$$

$$o_c^b \leftrightarrow \bigvee_{b' \in \mathsf{chldn}(b): c \in \mathsf{cols}(b')} o_c^{b'} \qquad \text{disjoint union } b, \text{ every } c \in \mathsf{cols}(b) \tag{17}$$

$$o^b_c \leftrightarrow \bigvee_{\text{if } b \text{ relabeling } c' \mapsto c} o^{b'}_{c'} \vee \bigvee_{\text{if } c \in \mathsf{cols}(b')} o^{b'}_c \quad \text{relabeling } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{18}$$

$$o^b_c \leftrightarrow o^{b'}_c \wedge \bigwedge_{\text{if} b \text{ edge introduce } (c,c')} \neg e^b_{c'} \quad \text{edge introduce } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b)\text{, not introd. } (c', c) \tag{19}$$

$$o^b_c \leftrightarrow o^{b'}_c \wedge (c \neq c') \wedge d^{\geq b}_{c'} \quad \text{edge introduce } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b)\text{, introd. } (c', c) \tag{20}$$

To compute $d^{\geq b}_c$, we need to propagate the information of being defeated backwards, from the root towards the leaves.

$$d^{\geq rt}_c \leftrightarrow \bigvee_{\text{if } rt \text{ edge introduce } (c',c)} e^{rt}_{c'} \quad \text{for root colors } c \in \mathsf{cols}(rt) \tag{21}$$

$$d^{\geq b'}_c \leftrightarrow d^{\geq b}_c \vee \bigvee_{\text{if } b' \text{ edge introduce } (c',c)} e^{b'}_{c'} \quad \text{non-relabeling } b \text{ with } b' \in \mathsf{chldn}(b), c \in \mathsf{cols}(b') \tag{22}$$

$$d^{\geq b'}_c \leftrightarrow \bigvee_{\text{if } b \text{ relabeling } c \mapsto c'} d^{\geq b}_{c'} \vee \bigvee_{\text{if } c \in \mathsf{cols}(b)} d^{\geq b}_c \quad \text{relabeling } b \text{ with } b' \in \mathsf{chldn}(b), c \in \mathsf{cols}(b') \tag{23}$$

In the root rt no color remains attacking or out.

$$\neg a^{rt}_c, \quad \neg o^{rt}_c \quad \text{for every } c \in \mathsf{cols}(rt) \tag{24}$$

Intuition. We encode whether some argument is incorrectly left out via a set O of variables. To achieve this, Formula (16) initiates arguments not in an extension as candidates for incorrectly left out. Formulas (17)–(18) guide this information along the k-expression. Our disjunctive encoding here guarantees that we have considered each argument of any color. Then, Formulas (19)–(20) update the status of an argument depending on whether there is a valid reason to leave this out. Precisely, Formula (19) guarantees that if a color c attacks an extension argument, it can not be left out since this must be defeated by the extension, due to the admissibility, and Formula (20) encodes that an argument was correctly left out, if there is some undefended attacker. Then, Formulas (24) confirm that no argument is incorrectly left out.

Theorem 11 ($\star$,Correctness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression of F. The DDG reduction $\mathcal{R}_{\#\mathsf{comp}\to\#\text{SAT}}$ is correct, that is, $\#\mathsf{comp}$ on F coincides with $\#\text{SAT}$ on $\mathcal{R}_{\#\mathsf{comp}\to\#\text{SAT}}(F, \mathcal{X})$.*

Theorem 12 ($\star$,CW-Awareness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression. The DDG reduction $\mathcal{R}_{\#\mathsf{comp}\to\#\text{SAT}}(F, \mathcal{X})$ constructs a SAT instance ψ that linearly preserves the width, i.e., $\mathsf{dcw}(\mathcal{G}^d_i(\psi))) \in \mathcal{O}(k)$.*

Semantics Using Subset-Maximization

Before we turn our attention to maximization-based semantics, we require some meta-result on DNF matrices, which will substantially simplify our constructions below.

Lemma 13 ($\star$)**.** *Let Q be a QBF with inner-most $\forall$ quantifier and matrix $\varphi \wedge \psi$, where φ is in CNF and ψ is in DNF. Assuming $k = \mathsf{dcw}(\mathcal{G}^d_i(\varphi) \sqcup \mathcal{G}^d_i(\psi))$, there is a model-preserving DNF matrix φ' with $\mathsf{dcw}(\mathcal{G}^d_i(\varphi'')) \in \mathcal{O}(k)$.*

Proof (Sketch). It suffices to encode the CNF φ into a DNF formula φ' along the k-expression. The idea of our encoding is to guide the satisfiability of clauses along the k-expression, thereby keeping the information of whether a variable is assigned false or true. The resulting DNF matrix uses the formula φ' instead. Importantly, our encoding linearly preserves the directed incidence clique-width. □

Preferred Extensions. As before, we take an AF F, a k-expression $\mathcal{X}$ of F, and compute admissible extensions via the formula $\varphi_{\#\mathsf{adm}}$. However, we also need to keep track of subset-larger extension candidates. So, we additionally create the formula $\varphi^*_{\#\mathsf{adm}}$, where starring refers to renaming every resulting variable v by a new copy v^*. It remains to design a structure-aware encoding of subset-larger extensions via starred variables. We construct the following CNF φ_{pref} (given as Equations (25)–(31)).

$$\psi \quad \text{for every } \psi \in \varphi^*_{\#\mathsf{adm}} \tag{25}$$

$$s^b_c \leftrightarrow e^{*b}_c \wedge \neg e^b_c \quad \text{initial } b\text{, create } a \text{ of color } c \tag{26}$$

$$s^b_c \leftrightarrow \bigvee_{b' \in \mathsf{chldn}(b): c \in \mathsf{cols}(b')} s^{b'}_c \quad \text{disjoint union } b\text{, every } c \in \mathsf{cols}(b) \tag{27}$$

$$s^b_c \leftrightarrow \bigvee_{\text{if } b \text{ relabeling } c' \mapsto c} s^{b'}_{c'} \vee \bigvee_{\text{if } c \in \mathsf{cols}(b')} s^{b'}_c \quad \text{relabeling } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{28}$$

$$s^b_c \leftrightarrow s^{b'}_c \quad \text{edge introduce } b\text{, every } c \in \mathsf{cols}(b),\ b' \in \mathsf{chldn}(b) \tag{29}$$

We skip subset-larger extension counter candidates that are not in a superset relation to the candidate.

$$e^b_c \rightarrow e^{*b}_c \quad \text{initial } b\text{, create } a \text{ of color } c \tag{30}$$

Further, for the root operation rt, we need to find an admissible extension that is subset-maximal, expressed as follows.

$$\bigvee_{c \in \mathsf{cols}(rt)} s^{rt}_c \quad \text{for root operation } rt \tag{31}$$

Then, the reduction $\mathcal{R}_{\#\mathsf{pref}\to\#2\text{-QBF}}(F, \mathcal{X})$ constructs a QBF $\varphi_{\#\mathsf{pref}} := (\varphi_{\#\mathsf{adm}} \wedge \neg(\exists E^*, T^*, S.\varphi_{\mathsf{pref}}))$, which searches for an admissible extension (free variables) where there is no subset-larger admissible extension ($\neg\exists$). So, if there indeed is a larger extension than the candidate given via e^b_a variables, the QBF evaluates to false.

If we bring this QBF into prenex normal form (shifting negation inside), we obtain an $\forall$-QBF with free variables whose matrix is of the form $\varphi_{\#\mathsf{adm}} \wedge \varphi_{\mathsf{pref}}$ with $\varphi_{\#\mathsf{adm}}$ in CNF and φ_{pref} in DNF. This matrix can then be converted to DNF by Lemma 13. We obtain the following result.

Theorem 14 ($\star$,Correctness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression of F. The DDG reduction $\mathcal{R}_{\#\mathsf{pref}\to\#2\text{-QBF}}$ is correct, that is, $\#\mathsf{pref}$ on F coincides with $\#2\text{-QBF}$ on $\mathcal{R}_{\#\mathsf{pref}\to\#2\text{-QBF}}(F, \mathcal{X})$.*

Theorem 15 ($\star$,CW-Awareness)**.** *Let F be an AF and $\mathcal{X}$ k-expression. The reduction $\mathcal{R}_{\#\mathsf{pref}\to\#2\text{-QBF}}(F, \mathcal{X})$ constructs a QSAT instance ψ that linearly preserves the width, i.e., $\mathsf{dcw}\,\mathcal{G}^d_i(\mathsf{matrix}(\psi)) \in \mathcal{O}(k)$.*

Semi-Stable Extensions. To compute semi-stable extensions, we must maximize the range of arguments. Interestingly, the range is computed, but via the information on colors we can not directly access it. We construct the following CNF $\varphi_{\mathsf{semiSt}}$ (given as equivalences).

$$\gamma \quad \text{for every } \gamma \in \text{Formulas } (5) - (8)$$

$$\psi \qquad \text{for every } \psi \in \varphi^*_{\#\mathsf{adm}} \tag{32}$$

$$s^b_c \leftrightarrow d^{*b}_c \wedge \neg d^b_c \qquad \text{initial } b, \text{ every } c \in \mathsf{cols}(b) \tag{33}$$

$$s^b_c \leftrightarrow \bigvee_{b' \in \mathsf{chldn}(b): c \in \mathsf{cols}(b')} s^{b'}_c \qquad \text{disjoint union } b, \text{ every } c \in \mathsf{cols}(b) \tag{34}$$

$$s^b_c \leftrightarrow \bigvee_{\text{if } b \text{ relabeling } c' \mapsto c} s^{b'}_{c'} \vee \bigvee_{\text{if } c \in \mathsf{cols}(b')} s^{b'}_c \qquad \text{relabeling } b, \text{ every } c \in \mathsf{cols}(b), b' \in \mathsf{chldn}(b) \tag{35}$$

$$s^b_c \leftrightarrow (s^{b'}_c \vee d^{*b}_c) \wedge \neg d^b_c \qquad \text{edge introduce } b, \text{ every } c \in \mathsf{cols}(b), b' \in \mathsf{chldn}(b) \tag{36}$$

For the root rt, we keep extensions of strictly larger range.

$$d^{rt}_c \rightarrow d^{*rt}_c, \qquad \bigvee_{c \in \mathsf{cols}(rt)} s^{rt}_c \qquad \text{for root operation } rt \tag{37}$$

The reduction $\mathcal{R}_{\#\mathsf{semiSt}\rightarrow\#2\text{-QBF}}(F, \mathcal{X})$ constructs a QBF $(\varphi_{\#\mathsf{adm}} \wedge (\forall E^*, D^*, T^*, S. \neg\varphi_{\mathsf{semiSt}}))$, where $\neg\varphi_{\mathsf{semiSt}}$ is in DNF, but $\varphi_{\#\mathsf{adm}}$ is in CNF. As above, Lemma 13 converts the matrix into DNF as desired.

Theorem 16 ($\star$,Correctness)**.** *Let F be an AF and $\mathcal{X}$ be a k-expression of F. Then, the DDG reduction $\mathcal{R}_{\#\mathsf{semiSt}\rightarrow\#2\text{-QBF}}$ is correct, that is, $\#\mathsf{semiSt}$ on F coincides with $\#2$-QBF on $\mathcal{R}_{\#\mathsf{semiSt}\rightarrow\#2\text{-QBF}}(F, \mathcal{X})$.*

Theorem 17 ($\star$,CW-Awareness)**.** *Let F be an AF and $\mathcal{X}$ a k-expression. DDG reduction $\mathcal{R}_{\#\mathsf{semiSt}\rightarrow\#2\text{-QBF}}(F, \mathcal{X})$ constructs a QSAT instance ψ that linearly preserves the width, i.e., $\mathsf{dcw}(\mathcal{G}^d_i(\mathsf{matrix}(\psi))) \in \mathcal{O}(k)$.*

Stage Extensions. The reduction $\mathcal{R}_{\#\mathsf{stage}\rightarrow\#2\text{-QBF}}(F, \mathcal{X})$ constructs a QBF $\forall E^*, D^*, S.(\varphi_{\#\mathsf{conf}} \wedge \neg\varphi_{\mathsf{stage}})$, where φ_{stage} is a CNF comprising ψ for every $\psi \in \varphi_{\#\mathsf{conf}}$ as well as Equations (33)–(37). Correctness and CW-Awareness works as above, so we obtain the following upper bounds.

Theorem 18 (Runtime-UBs)**.** *Let $F = (A, R)$ be an AF of size n and directed clique-width k, For a semantics σ, the problem $\#\sigma$ can be solved in time*

- $2^{\mathcal{O}(k)} \cdot poly(n)$ *for* $\sigma \in \{\mathsf{stab}, \mathsf{adm}, \mathsf{comp}\}$.
- $2^{2^{\mathcal{O}(k)}} \cdot poly(n)$ *for* $\sigma \in \{\mathsf{pref}, \mathsf{semiSt}, \mathsf{stage}\}$.

Credulous and Skeptical Reasoning

The preceding reductions can be extended to determine credulous and skeptical acceptance of an argument. Let F be an AF and σ be a semantics. To solve credulous acceptance c_σ for a, we append "e_a" to each formula $\varphi_{\#\sigma}$ where $e_a \in E$ is the extension variable corresponding to argument a. Then, each satisfying assignments for $\varphi_\sigma \wedge e_a$ yields an extension (via extension variables) containing a. Moreover, to solve skeptical acceptance s_σ, we add "$\neg e_a$" to $\varphi_{\#\sigma}$ and flip the answer in polynomial time, i.e., s_σ is true for a if and only if there is no satisfying assignment for $\varphi_{\#\sigma} \wedge \neg e_a$.

We observe that correctness and CW-awareness in both cases for each semantics follows from proofs of corresponding "Correctness" and "CW-Awareness" theorems, e.g., Thm. 7 and Thm. 8 for stable semantics.

Theorem 19 (Runtime-UBs)**.** *Let $F = (A, R)$ be an AF of size n and directed clique-width k, For a semantics σ, the problem s_σ can be solved in time*

- $2^{\mathcal{O}(k)} \cdot poly(n)$ *for* $\sigma \in \{\mathsf{stab}, \mathsf{adm}, \mathsf{comp}\}$.
- $2^{2^{\mathcal{O}(k)}} \cdot poly(n)$ *for* $\sigma \in \{\mathsf{pref}, \mathsf{semiSt}, \mathsf{stage}\}$.

c_σ behaves similarly, but $\mathsf{c}_{\mathsf{pref}}$ can be solved via $\mathsf{c}_{\mathsf{adm}}$.

Lower Bounds: Can We Improve?

It turns out that we can not significantly improve most of our reductions. Indeed, we can create a clique-width-aware reduction from 3SAT to asking whether some argument is included in an admissible extension.

Theorem 20 ($\star$,Admissible CW-LB)**.** *Unless ETH fails, we can not decide for an AF $F = (A, R)$ of directed clique-width w in time $2^{o(w)} \cdot poly(|A| + |R|)$ whether there exists an admissible extension of F containing argument a.*

We can easily extend this to other semantics. Indeed, the lower bound immediately carries over to other semantics.

Corollary 21 (Stable/Complete CW-LB)**.** *Under ETH we can not decide for an AF $F = (A, R)$ of directed clique-width w in time $2^{o(w)} \cdot poly(|A| + |R|)$ whether there is a stable/complete extension of F containing a.*

The bounds can be extended to second-level extensions.

Proposition 22 (Preferred/Semi-Stable/Stage CW-LB)**.** *Under ETH we can not decide for an AF $F = (A, R)$ of directed clique-width w in time $2^{2^{o(w)}} \cdot poly(|A|+|R|)$ whether there exists a preferred (semi-stable/stage) extension of F that does not contain a (contains a).*

These results indicate that we cannot significantly improve our reductions. Indeed, for solving the second-level semantics, a reduction to SAT is expected to be insufficient.

Conclusion

Our results answer whether we can efficiently encode knowledge representation and reasoning (KRR) formalisms into (Q)SAT while respecting the clique-width. Table 1 provides a comprehensive overview. Our directed decomposition guided (DDG) reductions based on k-expressions make existing results on clique-width for SAT (Proposition 1) and QSAT (Proposition 5) accessible to abstract argumentation. Using these results and our novel DDG reduction, we establish efficient solvability when exploiting clique-width for all argumentation semantics and the problems of extension existence, acceptance, and counting. Finally, we prove that we cannot significantly improve under reasonable assumptions. Our approach remains effective even when the attack graph includes large cliques or complete bipartite structures, where other parameters (e.g., treewidth) fail.

We see various directions for future works. We are interested whether these results can be extended to other KRR formalisms such as abductive reasoning, logic-based argumentation, answer-set programming, and many more. We expect that this work might be useful for simple classroom-type proofs of Courcelle's theorem for clique-width, see (Bannach and Hecher 2025). Since our reductions preserve the solutions bijectively, we are interested in enumeration complexity as well. Finally, generalized or orthogonal versions of clique-width such as modular incidence treewidth and symmetric incidence clique-width.

Acknowledgments

Authors are stated in reverse alphabetical order. Research was partly funded by the Austrian Science Fund (FWF), grant J4656, the French Agence nationale de la recherche (ANR), grant ANR-25-CE23-7647, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant TRR 318/1 2021 – 438445824, the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within project WHALE (LFN 1-04) funded under the Lamarr Fellow Network programme, and within project SAIL, grant NW21-059D. Part of the research was carried out while Hecher was a postdoc at MIT and while he was visiting the Simons institute for the theory of computing (part of the program *Logic and Algorithms in Database Theory and AI*). Fichte was funded by ELLIIT funded by the Swedish government.

References

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

Bannach, M.; and Hecher, M. 2025. Structure-Guided Automated Reasoning. In *Proceedings STACS 2025*, volume 327 of *LIPIcs*, 15:1–15:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Baroni, P.; Dunne, P. E.; and Giacomo, G. D., eds. 2010. *Proceedings of COMMA'10*. Desenzano del Garda, Italy.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability – 2nd Edition*, volume 336. IOS Press.

Bodlaender, H. L. 2006. Treewidth: Characterizations, Applications, and Computations. In Fomin, F. V., ed., *Graph-Theoretic Concepts in Computer Science*, 1–14. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-48382-3.

Bondy, J. A.; and Murty, U. S. R. 2008. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer. ISBN 978-1-84628-970-5.

Borie, R. B.; Parker, R. G.; and Tovey, C. A. 1992. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(1): 555–581.

Capelli, F.; and Mengel, S. 2019. Tractable QBF by Knowledge Compilation. In *Proceedings of STACS'19*, 18:1–18:16. Dagstuhl Publishing. ISBN 978-3-95977-100-9.

Charwat, G.; Dvořák, W.; Gaggl, S. A.; Wallner, J. P.; and Woltran, S. 2015. Methods for solving reasoning problems in abstract argumentation – A survey. *Artif. Intell.*, 220: 28–63.

Corneil, D. G.; and Rotics, U. 2005. On the Relationship Between Clique-Width and Treewidth. *SIAM Journal on Computing*, 34(4): 825–847.

Courcelle, B. 1990. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs* 1. *Information and computation*, 85(1): 12–75.

Courcelle, B. 1993. Monadic second-order logic and hypergraph orientation. In *Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, 179–190.

Courcelle, B. 2018. From tree-decompositions to clique-width terms. *Discr. Appl. Math.*, 248: 125–144.

Courcelle, B.; Engelfriet, J.; and Rozenberg, G. 1993. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2): 218–270.

Courcelle, B.; and Olariu, S. 2000. Upper bounds to the clique width of graphs. *Discr. Appl. Math.*, 101(1–3): 77–114.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Dániel Marx, M. P.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.*, 113(1): 41–85.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. London, UK: Springer. ISBN 978-1-4471-5558-4.

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

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

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

Dvorák, W.; Szeider, S.; and Woltran, S. 2010. Reasoning in Argumentation Frameworks of Bounded Clique-Width. In *Proceedings of COMMA'10*, volume 216, 219–230. IOS Press.

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

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

Eiter, T.; and Gottlob, G. 1993. Propositional Circumscription and Extended Closed World Reasoning are Π_2^P-Complete. *Theor. Comput. Sci.*, 114(2): 231–245.

Fichte, J. K.; Hecher, M.; Mahmood, Y.; and Meier, A. 2021. Decomposition-Guided Reductions for Argumentation and Treewidth. In *Proceedings of IJCAI'21*, 1880–1886. ijcai.org.

Fichte, J. K.; Hecher, M.; and Meier, A. 2024. Counting Complexity for Reasoning in Abstract Argumentation. *J. Artif. Intell. Res.*, 80: 805–834.

Fichte, J. K.; Hecher, M.; and Pfandler, A. 2020. Lower Bounds for QBFs of Bounded Treewidth. In Kobayashi, N., ed., *LICS'20*, 410–424. ACM.

Fischer, E.; Makowsky, J.; and Ravve, E. 2008. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.*, 156(4): 511–529.

Freuder, E. C. 1985. A sufficient condition for backtrack-bounded search. *J. ACM*, 32(4): 755–761.

Hecher, M. 2020. Treewidth-aware Reductions of Normal ASP to SAT - Is Normal ASP Harder than SAT after All? In *Proceedings of KR'20*, 485–495.

Heule, M. J. H.; Lynce, I.; Szeider, S.; and Schidler, A. 2023. SAT Encodings and Beyond (Dagstuhl Seminar 23261). *Dagstuhl Reports*, 13(6): 106–122.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4): 512–530.

Järvisalo, M.; Lehtonen, T.; and Niskanen, A. 2025. ICCMA 2023: 5th International Competition on Computational Models of Argumentation. *Artif. Intell.*, 104311.

Kanté, M. M. 2007. The rank-width of Directed Graphs. *CoRR*, abs/0709.1433.

Lampis, M.; Mengel, S.; and Mitsou, V. 2018. QBF as an Alternative to Courcelle's Theorem. In *Proceedings of SAT'18*, volume 10929, 235–252. Springer.

Mahmood, Y.; Hecher, M.; Groven, J.; and Fichte, J. K. 2025. Structure-Aware Encodings of Argumentation Properties for Clique-width. arXiv:2511.10767.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press. ISBN 978-0-19-856607-6.

Niskanen, A.; and Järvisalo, M. 2020. μ-toksia: An Efficient Abstract Argumentation Reasoner. In *Proceedings of KR'20*, 800–804.

Ordyniak, S.; Paulusma, D.; and Szeider, S. 2013. Satisfiability of acyclic and almost acyclic CNF formulas. *Theor. Comput. Sci.*, 481: 85–99.

Oum, S.; and Seymour, P. D. 2006. Approximating clique-width and branch-width. *J. Comb. Theory B*, 96(4): 514–528.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley. ISBN 0-470-86412-5.

Paulusma, D.; Slivovsky, F.; and Szeider, S. 2016. Model Counting for CNF Formulas of Bounded Modular Treewidth. *Algorithmica*, 76(1): 168–194.

Rago, A.; Cocarascu, O.; and Toni, F. 2018. Argumentation-Based Recommendations: Fantastic Explanations and How to Find Them. In *Proceedings of IJCAI'18*, 1949–1955. The AAAI Press.

Rahwan, I. 2007. Argumentation in Artificial Intelligence. *Artif. Intell.*, 171(10-15): 619–641.

Slivovsky, F.; and Szeider, S. 2013. Model Counting for Formulas of Bounded Clique-Width. In *Proceedings of ISAAC'13*, 677–687. Springer. ISBN 978-3-642-45030-3.

Truszczynski, M. 2011. Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs. *Theory Pract. Log. Program.*, 11(6): 881–904.