Satisfiability of Acyclic and Almost Acyclic CNF Formulas*

Sebastian Ordyniak[1]**, Daniel Paulusma[2]***, and Stefan Szeider[1]**

[1] Institute of Information Systems, Vienna University of Technology, A-1040 Vienna, Austria
sebastian.ordyniak@kr.tuwien.ac.at,stefan@szeider.net
[2] School of Engineering and Computing Sciences, Durham University, Durham, DH1 3LE, UK
daniel.paulusma@durham.ac.uk

Abstract. We show that the SATISFIABILITY (SAT) problem for CNF formulas with β-acyclic hypergraphs can be solved in polynomial time by using a special type of Davis-Putnam resolution where each resolvent is a subset of a parent clause. We extend this class to CNF formulas for which this type of Davis-Putnam resolution still applies and show that testing membership in this class is NP-complete. We compare the class of β-acyclic formulas and this superclass with a number of known polynomial formula classes. We then study the parameterized complexity of SAT for "almost" β-acyclic instances, using as parameter the formula's distance from being β-acyclic. As distance we use the size of smallest strong backdoor sets and the β-hypertree width. As a by-product we obtain the W[1]-hardness of SAT parameterized by the (undirected) clique-width of the incidence graph, which disproves a conjecture by Fischer, Makowsky, and Ravve.

Keywords acyclic hypergraph, chordal bipartite graph, Davis-Putnam resolution.

1 Introduction

We consider the SATISFIABILITY (SAT) problem on classes of CNF formulas (formulas in Conjunctive Normal Form) with restrictions on their associated hypergraphs, which are obtained from these formulas by ignoring negations and considering clauses as hyperedges on variables. This is a natural study, because many computationally hard problems can be solved efficiently on acyclic instances. However, there are several notions of acyclicity for hypergraphs: α-acyclicity, β-acyclicity, γ-acyclicity, and Berge acyclicity. We provide the relevant definitions in Section 2 and refer to Fagin [9] for a detailed description. The notions of acyclicity are strictly ordered with respect to their generality:

$$\alpha\text{-ACYC} \supsetneq \beta\text{-ACYC} \supsetneq \gamma\text{-ACYC} \supsetneq \text{Berge-ACYC} \tag{1}$$

where X-ACYC denotes the class of X-acyclic hypergraphs, which are in 1-to-1 correspondence to a class of CNF formulas called X-acyclic formulas. It is known that SAT is NP-complete for α-acyclic formulas [26] and polynomial-time solvable for Berge-acyclic formulas [10, 26].

* Extended abstracts of this paper appeared in the Proceedings of FSTTCS 2010 and SAT 2011.
** Supported by ERC (COMPLEX REASON, 239962).
*** Supported by EPSRC (EP/G043434/1).

Our Results. In Section 3 we determine the boundary between NP-completeness and polynomial-time tractability in the chain (1) by showing that SAT is polynomial-time solvable for β-acyclic formulas. Consequently, the same holds for γ-acyclic formulas. To prove our result we use a fundamental procedure called the Davis-Putnam procedure, which successively eliminates variables using Davis-Putnam resolution [7]. In general, this procedure is not efficient, because the number of clauses may increase after each application of Davis-Putnam resolution. However, β-acyclic formulas are related to chordal bipartite graphs [30], and this allows us to compute an elimination ordering of the variables with the property that each obtained resolvent is a subset of a parent clause. This type of resolution is known as subsumption resolution [19].

In Section 4 we show that there are CNF formulas that are not β-acyclic but that still admit an elimination ordering of their variables based on subsumption resolution, such that the Davis-Putnam procedure takes polynomial time. We call such an elimination ordering *DP-simplicial*. This leads to a new class DPS of CNF formulas that contains the class of β-acyclic formulas. In Section 5 we show that testing membership in this class is an NP-complete problem. The reason for the NP-hardness is that a formula may have several so-called *DP-simplicial* variables, one of which must be chosen to be eliminated but we do not know which one. In Section 6 we show how to work around this obstacle to some extent, i.e., we identify a subclass of DPS that is a proper superclass of the class of β-acyclic formulas for which SAT is polynomial-time solvable. In Section 7 we show that the class of β-acyclic formulas and its superclass DPS are *incomparable* with other known polynomial classes of CNF formulas. Hence, β-acyclic formulas form a new "island of tractability" for SAT.

In Section 8 we study the complexity of SAT for formulas that are parameterized by their "distance" from the class of β-acyclic CNF formulas. We use two distance measures. The first distance measure is based on the notion of a *strong backdoor set*. For a CNF formula F we define its "distance to β-acyclicity" as the size k of a smallest set B of variables such that for each partial truth assignment to B, the reduct of F under the assignment is β-acyclic; such a set B is a strong backdoor set. If we know B, then deciding the satisfiability of F reduces to deciding the satisfiability of at most 2^k β-acyclic CNF formulas, and is thus fixed-parameter tractable with respect to k. We show, however, that finding such a set B of size k (if it exists) is W[2]-hard, thus unlikely fixed-parameter tractable for parameter k, which limits the algorithmic usefulness of this distance measure.

The second distance measure we consider is the *β-hypertree width*, a hypergraph invariant introduced by Gottlob and Pichler [15]. The classes of hypergraphs of β-hypertree width $k = 1, 2, 3, \ldots$ form an infinite chain of proper inclusions. Hypergraphs of β-hypertree width 1 are exactly the β-acyclic hypergraphs. Thus β-hypertree width is also a way to define a "distance to β-acyclicity." The complexity of determining the β-hypertree width of a hypergraph is open [15]. However, we show that SAT parameterized by an upper bound on the β-hypertree width is W[1]-hard even if we are given the CNF formula together with a β-hypertree decomposition of width k. As a side effect, we obtain from this result that SAT is also W[1]-hard when parameterized by the *clique-width* (of the undirected incidence graph) of the CNF formula. This disproves a conjecture by Fischer, Makowsky, and Ravve [10].

2 Preliminaries

In this section we state our basic terminology and notations. We also present some known results that will be useful at several places in the paper.

2.1 Formulas and Resolution

We assume an infinite supply of propositional *variables*. A *literal* is a variable x or a negated variable $\overline{x}$; if $y = \overline{x}$ is a literal, then we write $\overline{y} = x$. For a set S of literals we put $\overline{S} = \{\, \overline{x} \mid x \in S \,\}$; S is *tautological* if $S \cap \overline{S} \neq \emptyset$. A *clause* is a finite non-tautological set of literals. A finite set of clauses is a *CNF formula* (or *formula*, for short). A variable x *occurs* in a clause C if $x \in C \cup \overline{C}$; $\mathrm{var}(C)$ denotes the set of variables which occur in C. A variable x *occurs* in a formula F if it occurs in one of its clauses, and we put $\mathrm{var}(F) = \bigcup_{C \in F} \mathrm{var}(C)$. If F is a formula and X a set of variables, then we denote by $F - B$ the formula obtained from F after removing all literals x and $\overline{x}$ with $x \in B$ from the clauses in F. If $X = \{x\}$ we simply write $F - x$ instead of $F - \{x\}$.

Let F be a formula and $X \subseteq \mathrm{var}(F)$. A *truth assignment* is a mapping $\tau : X \to \{\, 0, 1 \,\}$ defined on some set X of variables; we write $\mathrm{var}(\tau) = X$. For $x \in \mathrm{var}(\tau)$ we define $\tau(\overline{x}) = 1 - \tau(x)$. For a truth assignment τ and a formula F, we define

$$F[\tau] = \{\, C \setminus \tau^{-1}(0) \mid C \in F,\ C \cap \tau^{-1}(1) = \emptyset \,\},$$

i.e., $F[\tau]$ denotes the result of instantiating variables according to τ and applying the usual simplifications. A truth assignment τ *satisfies* a clause C if C contains some literal x with $\tau(x) = 1$; τ satisfies a formula F if it satisfies all clauses of F. A formula is *satisfiable* if it is satisfied by some truth assignment; otherwise it is *unsatisfiable*. Two formulas F and F' are *equisatisfiable* if either both are satisfiable or both are unsatisfiable. The SATISFIABILITY (SAT) problem asks whether a given CNF formula is satisfiable.

Let C, D be two clauses such that $C \cap \overline{D} = \{x\}$ for a variable x. The clause $(C \cup D) \setminus \{x, \overline{x}\}$ is called the *x-resolvent* (or *resolvent*) of C and D; the clauses C and D are called *parent clauses* of the x-resolvent. Note that by definition any two clauses have at most one resolvent. Let F be a formula. A sequence $C_1, \ldots, C_n$ is a *resolution derivation* of C_n from F if every C_i is either in F or the resolvent of two clauses C_j and $C_{j'}$ for some $1 \leq j < j' \leq i - 1$. If C_n is the empty clause, then the sequence is called a *resolution refutation* of F. The derivation is *minimal* if we cannot delete a clause from it and still have a resolution derivation of C_n from F. We call a clause C_n a *resolution descendant* of a clause $C_1 \in F$ if there is a minimal resolution derivation $C_1, \ldots, C_n$ of C_n from F.

Consider a formula F and a variable x of F. Let $\mathrm{DP}_x(F)$ denote the formula obtained from F after adding all possible x-resolvents and removing all clauses in which x occurs. We say that $\mathrm{DP}_x(F)$ is obtained from F by *Davis-Putnam resolution*, and that we *eliminated* x. It is well known (and easy to show) that F and $\mathrm{DP}_x(F)$ are equisatisfiable.

For an ordered sequence of variables $x_1, \ldots, x_k$ of F, we set $\mathrm{DP}_{x_1,\ldots,x_k}(F) = \mathrm{DP}_{x_k}(\cdots(\mathrm{DP}_{x_1}(F))\cdots)$ and $\mathrm{DP}_\emptyset(F) = F$. The Davis-Putnam procedure [7] is a

well-known algorithm that solves SAT. In its most basic form, it takes an ordering of the variables $x_1, \ldots, x_n$ of a formula F and checks whether $\mathrm{DP}_{x_1,\ldots,x_n}(F)$ is empty or contains the empty clause. In the first case F is satisfiable, and in the second case F is unsatisfiable. Note that this procedure computes a certificate for the (un)satisfiability of F; we can obtain a satisfying truth assignment of F from a satisfying truth assignment of $\mathrm{DP}_x(F)$, and we can obtain a resolution refutation of F from a resolution refutation of $\mathrm{DP}_x(F)$. However, $\mathrm{DP}_x(F)$ contains in general more clauses than F. Hence, repeated application of Davis-Putnam resolution to F may cause an exponential growth in the number of clauses. As a result, the Davis-Putnam procedure has an exponential worst-case running time.

2.2 Graphs and Hypergraphs

A *hypergraph* H is a pair (V, E) where V is the set of *vertices* and E is the set of *hyperedges*, which are subsets of V. If $|e| = 2$ then we call e an *edge*; we denote an edge $e = \{u, v\}$ simply as uv or vu. If all hyperedges of a hypergraph are edges then we call it a *graph*. We say that a hypergraph $H' = (V', E')$ is a *partial hypergraph* of $H = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. The *incidence graph* $I(H)$ of hypergraph $H = (V, E)$ is the bipartite graph where the sets V and E form the two partitions, and where $e \in E$ is incident with $v \in V$ if and only if $v \in e$. A hypergraph is α*-acyclic* if it can be reduced to the empty hypergraph by repeated application of the following rules:

1. Remove hyperedges that are empty or contained in other hyperedges.
2. Remove vertices that appear in at most one hyperedge.

A hypergraph H is β*-acyclic* if every partial hypergraph of H is α-acyclic. The *hypergraph* $H(F)$ of a formula F has vertex set $\mathrm{var}(F)$ and hyperedge set $\{\, \mathrm{var}(C) \mid C \in F \,\}$. We say that F is α*-acyclic* or β*-acyclic* if $H(F)$ is α-acyclic or β-acyclic, respectively.

Let F be a formula. The *incidence graph* of F is the bipartite graph $I(F)$ with vertex set $\mathrm{var}(F) \cup F$ and edge set $\{\, Cx \mid C \in F \text{ and } x \in \mathrm{var}(C) \,\}$. The *directed incidence graph* of F is the directed graph $D(F)$ with vertex set $\mathrm{var}(F) \cup F$ and arc set $\{\, (C, x) \mid C \in F \text{ and } x \in C \,\} \cup \{\, (x, C) \mid C \in F \text{ and } \overline{x} \in C \,\}$. We can also represent the orientation of edges by labeling them with the signs $+, -$, such that an edge between a variable x and a clause C is labeled $+$ if $x \in C$ and labeled $-$ if $\overline{x} \in C$. This gives rise to the *signed incidence graph* which carries exactly the same information as the directed incidence graph.

The graph parameter *clique-width* measures in a certain sense the structural complexity of a directed or undirected graph [4]. The parameter is defined via a graph construction process where only a limited number of vertex labels are available; vertices that share the same label at a certain point of the construction process must be treated uniformly in subsequent steps. In particular, one can use the following four operations: the creation of a new vertex with label i, the vertex-disjoint union of already constructed labeled graphs, the relabeling of all vertices of label i with label j denoted $\rho_{i \to j}$, and the insertion of all possible edges between vertices of label i and label j denoted $\eta_{i,j}$ (either undirected, in which case we can also write $\eta_{j,i}$, or directed from label i to j).

The clique-width $\mathrm{cw}(G)$ of a graph G is the smallest number k of labels that suffice to construct G by means of these four operations. An algebraic term representing such a construction of G is called a *k-expression* of G. The *(directed) clique-width* of a CNF formula is the clique-width of its (directed) incidence graph. The directed clique-width of a CNF formula can also be defined in terms of the signed incidence graph and is therefore sometimes called the *signed clique-width*.

Let $G = (V, E)$ be a graph. For a subset $U \subseteq V$, the subgraph of G *induced by* U is the graph with vertex set U and edge set $\{uv \mid u, v \in U \text{ with } uv \in E\}$. A *cycle* is a graph, the vertices of which can be ordered as $v_1, \ldots, v_n$ such that $E = \{v_i v_{i+1} \mid 1 \leq i \leq n-1\} \cup \{v_n v_1\}$. A graph is *chordal bipartite* if it has no induced cycle on 6 vertices or more. A vertex v in a graph G is *weakly simplicial* if (i) the neighborhood of v in G forms an independent set, and (ii) the neighborhoods of the neighbors of v form a chain under set inclusion. Uehara [31] showed the following, which also follows from results of Hammer, Maffray, and Preismann [17], see [24]. We call a bipartite graph *nontrivial* if it contains at least one edge.

Proposition 1 ([17, 31]). *A graph is chordal bipartite if and only if every induced subgraph has a weakly simplicial vertex. Moreover, a nontrivial chordal bipartite graph has a weakly simplicial vertex in each partition class.*

The following proposition shows how β-acyclic CNF formulas and chordal bipartite graphs are related. The equivalence between statement (i) and (ii) is due to Tarjan and Yannakakis [30], who presented this relationship in terms of β-acyclic hypergraphs. The equivalence between statement (ii) and (iii) follows from the facts that $I(H(F))$ is obtained from $I(F)$ after removing all but one clause vertices in $I(F)$ with the same neighbors, i.e., clauses with the same set of variables in F, and that a chordal bipartite graph remains chordal bipartite under vertex deletion.

Proposition 2 ([30]). *For a CNF formula F, statements (i)-(iii) are equivalent:*

(i) *F is β-acyclic;*
(ii) *$I(H(F))$ is chordal bipartite;*
(iii) *$I(F)$ is chordal bipartite.*

We also call a vertex of a hypergraph or a variable of a CNF formula *weakly simplicial* if the corresponding vertex in the associated incidence graph is weakly simplicial.

3 Polynomial-time SAT Decision for β-acyclic CNF Formulas

Note that we can make a hypergraph α-acyclic by adding a universal hyperedge that contains all vertices; by rule 1 we remove all other hyperedges, by rule 2 all vertices. By this observation, it is easy to see that SAT is NP-complete for the class of α-acyclic CNF formulas [26]. In contrast, it is well known that the satisfiability of α-acyclic instances of the CONSTRAINT SATISFACTION PROBLEM (CSP) can be decided in polynomial time [13]. Thus SAT and CSP behave differently with respect to α-acyclicity (representing a clause with k literals as a relational constraint requires exponential space of

order $k2^k$). However, in this section, we give a polynomial-time algorithm that solves SAT for β-acyclic CNF formulas.

If we can reduce a hypergraph H to the empty graph by repeated deletion of weakly simplicial vertices, then we say that H admits a *weakly simplicial elimination ordering*. If $H = H(F)$ for some formula F, then we also say that F admits a *weakly simplicial ordering* of its variables. The first key ingredient of our algorithm is the following lemma.

Lemma 1. *If F is a β-acyclic formula, then F admits a weakly simplicial elimination ordering. Moreover, such an ordering can be found in polynomial time.*

Proof. Let F be a β-acyclic formula. We must show that $H(F)$ admits a weakly simplicial elimination ordering. Proposition 2 tells us that $I(H(F))$ is chordal bipartite. Then $I(H(F))$ has a weakly simplicial vertex in each partition class due to Proposition 1. We choose the partition class of $I(H(F))$ that corresponds to the vertices of H. Then the lemma readily follows after observing that the class of chordal bipartite graphs is closed under vertex deletion and that weakly simplicial vertices can be identified in polynomial time by brute force. □

The following lemma is the second key ingredient for our algorithm. Recall that $\mathrm{DP}_x(F)$ denotes the formula obtained from a formula F after eliminating x by Davis-Putnam resolution.

Lemma 2. *If x is a weakly simplicial variable of a formula F, then $|\mathrm{DP}_x(F)| \leq |F|$.*

Proof. Let x be a weakly simplicial variable of a CNF formula F. Let $F - x := \{\, C \setminus \{x, \overline{x}\} \mid C \in F \,\}$. We show that $\mathrm{DP}_x(F) \subseteq F - x$.

Assume $C_1, C_2 \in F$ have a resolvent C with respect to x. Consequently we have $C_1 \cap \overline{C_2} \subseteq \{x, \overline{x}\}$. Because x is weakly simplicial, $\mathrm{var}(C_1) \subseteq \mathrm{var}(C_2)$ or $\mathrm{var}(C_2) \subseteq \mathrm{var}(C_1)$. Without loss of generality, assume the former is the case. If $x \in C_1$, then we have $C_1 \cap \overline{C_2} = \{x\}$, and so $C = C_2 \setminus \{\overline{x}\} \in F - x$. Similarly, if $\overline{x} \in C_1$, then we have $C_1 \cap \overline{C_2} = \{\overline{x}\}$, and so $C = C_2 \setminus \{x\} \in F - x$. Thus indeed $\mathrm{DP}_x(F) \subseteq F - x$. From $|\mathrm{DP}_x(F)| \leq |F - x| \leq |F|$ the result now follows. □

We are now ready to present our algorithm.

Algorithm solving SAT **for β-acyclic formulas**

Input : a β-acyclic formula F
Output : `Yes` if F is satisfiable
`No` otherwise

Step 1. compute a weakly simplicial elimination ordering $x_1, \ldots, x_n$ of F

Step 2. apply the Davis-Putnam procedure on ordering $x_1, \ldots, x_n$

We let BAC denote the class of all β-acyclic formulas and state the main result of this section.

Theorem 1. SAT *can be solved in polynomial time for* BAC.

Proof. Let F be a β-acyclic CNF formula. We apply our algorithm. Its correctness follows from Lemma 1 combined with the correctness of the Davis-Putnam procedure [7]. Steps 1 and 2 run in polynomial time due to Lemma 1 and Lemma 2, respectively. Hence, Theorem 1 follows. □

4 Generalizing β-Acyclic Formulas

Lemma 2 is one of the two key ingredients than ensures that our algorithm for solving SAT on BAC runs in polynomial time. It states that the number of clauses does not increase after applying Davis-Putnam resolution if x is a weakly simplicial variable of a formula F. We can ensure this by requiring the following property that is more general than being weakly simplicial. We say that a variable $x \in \text{var}(F)$ is *DP-simplicial* in a formula F if

(*) for any two clauses $C, D \in F$ that have an x-resolvent, this x-resolvent is a subset of C or a subset of D.

Observe that whenever an x-resolvent is a subset of a parent clause C then it is equal to $C \setminus \{x, \overline{x}\}$. The following lemma immediately follows from $(*)$.

Lemma 3. *If x is a DP-simplicial variable of a formula F, then $|\text{DP}_x(F)| \leq |F|$.*

An ordering $x_1, \ldots, x_n$ of the variables of F is a *DP-simplicial elimination ordering* if x_i is DP-simplicial in $\text{DP}_{x_1,\ldots,x_{i-1}}(F)$ for all $1 \leq i \leq n$. We let DPS denote the class of all formulas that admit a DP-simplicial elimination ordering. We observe that every weakly simplicial elimination ordering of $H(F)$ is a DP-simplicial elimination ordering of F. This means that BAC $\subseteq$ DPS. However, due to Example 4.1 below, the reverse is not true. Hence, we found the following result.

Proposition 3. BAC $\subsetneq$ DPS.

Given a DP-simplicial ordering, the Davis-Putnam procedure runs in polynomial time due to Lemma 3. This leads to the following result.

Proposition 4. *Let $F \in$ DPS. If a DP-simplicial elimination ordering of the variables in var(F) is given, then* SAT *can be solved in polynomial time for F.*

4.1 An Example

We give an example of a formula in DPS $\setminus$ BAC. Consider the formula F that has variables y, z, b, b', b^* and c and clauses $\{y, z, \overline{b}, b'\}$, $\{\overline{y}, \overline{z}, \overline{b}, b^*\}$, $\{y, \overline{b}\}$, $\{\overline{y}, \overline{b}\}$, $\{z, \overline{b}\}$, $\{\overline{z}, \overline{b}\}$, $\{y, b, b^*, c\}$, $\{\overline{y}, b, b', \overline{c}\}$, $\{\overline{b'}, b^*\}$, $\{\overline{b^*}, b'\}$, $\{c, b', b^*\}$, $\{\overline{c}, b', b^*\}$ and $\{\overline{b}, b'\}$.

We observe first that none of the variables of F are weakly simplicial. Consequently, there is no weakly simplicial elimination ordering of F. Hence $F \notin$ BAC. However, we will show below that y, b, b', b^*, c, z is a DP-simplicial elimination ordering of F. Then $F \in$ DPS, as desired.

We find that y is DP-simplicial in F and obtain $\mathrm{DP}_y(F) = \{\{z, \overline{b}, b'\}, \{\overline{z}, \overline{b}, b^*\}, \{z, \overline{b}\}, \{\overline{z}, \overline{b}\}, \{\overline{b'}, b^*\}, \{\overline{b^*}, b'\}, \{c, b', b^*\}, \{\overline{c}, b', b^*\}, \{\overline{b}, b'\}\}$. We then find that b is DP-simplicial in $\mathrm{DP}_y(F)$ and obtain $\mathrm{DP}_{y,b}(F) = \{\{\overline{b'}, b^*\}, \{\overline{b^*}, b'\}, \{c, b', b^*\}, \{\overline{c}, b', b^*\}\}$. We then find that b' is DP-simplicial in $\mathrm{DP}_{y,b}(F)$ and obtain $\mathrm{DP}_{y,b,b'}(F) = \{\{c, b', b^*\}, \{\overline{c}, b', b^*\}\}$. We then find that b^* is DP-simplicial in $\mathrm{DP}_{y,b,b'}(F)$ and obtain $\mathrm{DP}_{y,b,b',b^*}(F) = \emptyset$. Hence, y, b, b', b^*, c, z is a DP-simplicial elimination ordering of F.

We note that z is also DP-simplicial in F. Suppose that we started with z instead of y. We first derive that $\mathrm{DP}_z(F) = \{\{y, \overline{b}, b'\}, \{\overline{y}, \overline{b}, b^*\}, \{y, \overline{b}\}, \{\overline{y}, \overline{b}\}, \{y, b, b^*, c\}, \{\overline{y}, b, b', \overline{c}\}, \{\overline{b'}, b^*\}, \{\overline{b^*}, b'\}, \{c, b', b^*\}, \{\overline{c}, b', b^*\}, \{\overline{b}, b'\}\}$. In contrast to $\mathrm{DP}_y(F)$, the clauses $\{y, b, b^*, c\}$ and $\{\overline{y}, b, b', \overline{c}\}$ are still contained in $\mathrm{DP}_z(F)$. This implies that $\mathrm{DP}_z(F)$ has no DP-simplicial variables. Consequently, F has no DP-simplicial elimination ordering that starts with z.

We conclude that in contrast to weakly simplicial elimination orderings it is important to choose the right variable when we want to obtain a DP-simplicial elimination ordering. In the next section we will extend this consideration and show that making the right choice is in fact an NP-hard problem.

5 Recognizing Formulas in DPS

We prove that the problem of testing whether a given CNF formula belongs to the class DPS, i.e., admits a DP-simplicial elimination ordering, is NP-complete. This problem is in NP, because we can check in polynomial time whether an ordering of the variables of a CNF formula is a DP-simplicial elimination ordering. In order to show NP-hardness we reduce from SAT. In Section 5.1 we construct a CNF formula F' from a given CNF formula F. We also show a number of properties of F'. In Section 5.2 we use these properties to prove that F is satisfiable if and only if F' admits a DP-simplicial elimination ordering.

5.1 The Gadget and its Properties

For a given CNF formula F with variables $x_1, \ldots, x_n$ called the *x-variables* and clauses $C_1, \ldots, C_m$, we construct a CNF formula F' as follows. For every x_i we introduce two variables y_i and z_i. We call these variables the y-variables and z-variables, respectively. For every C_j we introduce a variable c_j. We call these variables the c-variables. We also add three new variables b, b' and b^* called the b-variables. We let $\mathrm{var}(F')$ consist of all b-variables, c-variables, y-variables, and z-variables.

Let C_j be a clause of F. We replace every x-variable in C by its associated y-variable if the occurrence of x in C is positive; otherwise we replace it by its associated z-variable. This yields a clause D_j. For instance, if $C_j = \{x_1, \overline{x_2}, x_3\}$ then $D_j = \{y_1, z_2, y_3\}$.

We let F' consist of the following $6n + 4m + 3$ clauses:

- $\{y_i, \overline{b}\}$ and $\{\overline{y_i}, \overline{b}\}$ for $i = 1, \ldots, n$ called by-clauses
- $\{z_i, \overline{b}\}$ and $\{\overline{z_i}, \overline{b}\}$ for $i = 1, \ldots, n$ called bz-clauses

- $\{y_i, z_i, \overline{b}, b'\}$ and $\{\overline{y_i}, \overline{z_i}, \overline{b}, b^*\}$ for $i = 1, \ldots, n$ called byz-clauses
- $\{c_j, b', b^*\}$ and $\{\overline{c_j}, b', b^*\}$ for $j = 1, \ldots, m$ called bc-clauses
- $D_j \cup \{b, b^*, c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ and $\overline{D_j} \cup \{b, b', c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ for $j = 1, \ldots, m$ called bcD-clauses
- $\{\overline{b}, b'\}$, $\{\overline{b'}, b^*\}$ and $\{b', \overline{b^*}\}$ called b-clauses.

We call a pair $D_j \cup \{b, b^*, c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ and $\overline{D_j} \cup \{b, b', c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ for some $1 \leq j \leq m$ a *bcD-clause pair.* We call a CNF formula M a *yz-reduction formula* of F' if there exists a sequence of variables $v^1, \ldots, v^k$, where every v^i is either a y-variable or a z-variable, such that $\text{DP}_{v^1,\ldots,v^k}(F') = M$, and v^i is DP-simplicial in $\text{DP}_{v^1,\ldots,v^{i-1}}(F')$ for $i = 1, \ldots, k$. We say that two clauses C and D *violate* (*) if they have a resolvent that is neither a subset of C nor a subset of D, i.e., $C \cap \overline{D} = \{v\}$ for some variable v but neither $(C \cup D) \setminus \{v, \overline{v}\} = C \setminus \{v\}$ nor $(C \cup D) \setminus \{v, \overline{v}\} = D \setminus \{\overline{v}\}$. We will now prove five useful lemmas valid for yz-reduction formulas.

Lemma 4. *Let M be a yz-reduction formula of F'. If M contains both clauses of some bcD-clause pair, then no b-variable and no c-variable is DP-simplicial in M.*

Proof. Let $E_1 = D_j \cup \{b, b^*, c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ and $E_2 = \overline{D_j} \cup \{b, b', c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ for some $1 \leq j \leq m$ be a bcD-clause pair in M. We observe that by definition M contains all b-clauses and bc-clauses. This enables us to prove the lemma. Let v be a b-variable or c-variable. Then we must distinguish 5 cases. If $v = b$, then $\{\overline{b}, b'\}$ and E_1 violate (*). If $v = b'$, then $\{\overline{b'}, b^*\}$ and E_2 violate (*). If $v = b^*$, then $\{b', \overline{b^*}\}$ and E_1 violate (*). If $v = c_j$, then $\{\overline{c_j}, b', b^*\}$ and E_1 violate (*). If $v = c_k$ for some $1 \leq k \leq m$ with $k \neq j$, then $\{c_k, b', b^*\}$ and E_1 violate (*). □

Lemma 5. *Let M be a yz-reduction formula of F'. Then $y_i \in var(M)$ or $z_i \in var(M)$ for $i = 1, \ldots, n$.*

Proof. Suppose that M does not contain y_i or z_i for some $1 \leq i \leq m$, say $y_i \notin \text{var}(M)$. We show that $z_i \in \text{var}(M)$. Let M' be the formula obtained from F' just before the elimination of y_i. Because M is a yz-reduction formula, M' is a yz-reduction formula as well. Hence, $\text{var}(M')$ contains all b-variables. Because y_i and z_i are in $\text{var}(M')$, we then find that M' contains the clauses $\{y_i, z_i, \overline{b}, b'\}$, $\{\overline{y_i}, \overline{b}\}$, $\{\overline{y_i}, \overline{z_i}, \overline{b}, b^*\}$ and $\{y_i, \overline{b}\}$. Because the first two clauses resolve into $\{z_i, \overline{b}, b'\}$, and the last two resolve into $\{\overline{z_i}, \overline{b}, b^*\}$, we obtain that $\text{DP}_{y_i}(M')$ contains $\{z_i, \overline{b}, b'\}$ and $\{\overline{z_i}, \overline{b}, b^*\}$, which violate (*). Because M contains all b-variables by definition, z_i will never become DP-simplicial when we process $\text{DP}_{y_i}(M')$ until we obtain M. Hence, $z_i \in \text{var}(M)$, as desired. □

Lemma 6. *Let M be a yz-reduction formula of F', and let $1 \leq j \leq m$. If there is a variable that occurs in D_j but not in M, then M neither contains $D_j \cup \{b, b^*, c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ nor $\overline{D_j} \cup \{b, b', c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ nor their resolution descendants.*

Proof. Let v be a variable that occurs in D_j but not in M. We may assume without loss of generality that v is the first variable in D_j that got eliminated and that $v = y_i$ for some

$1 \leq i \leq n$. Let S be the set that consists of all clauses $D_{j'} \cup \{b, b^*, c_{j'}\} \cup \{\, \overline{c_k} \mid k \neq j' \,\}$ and $\overline{D_{j'}} \cup \{b, b', c_{j'}\} \cup \{\, \overline{c_k} \mid k \neq j' \,\}$ in which y_i occurs.

Let M' be the formula obtained from F' just before the elimination of y_i. Because M is a yz-reduction formula, M' is a yz-reduction formula as well. Hence, by definition, all b-variables and all c-variables occur in M'. Then the clauses in M', in which y_i occurs, are $\{y_i, \overline{b}\}$, $\{\overline{y_i}, \overline{b}\}$, $\{y_i, z_i, \overline{b}, b'\}$,$\{\overline{y_i}, \overline{z_i}, \overline{b}, b^*\}$, together with clauses that are either from S or a resolution descendant of a clause in S. Note that these resolution descendants still contain all their b-variables and c-variables.

When we eliminate y_i, we remove all clauses in M' in which y_i occurs. Hence, $\mathrm{DP}_{y_i}(M')$, and consequently, M neither contains $E_1 = D_j \cup \{b, b^*, c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ nor $E_2 = \overline{D_j} \cup \{b, b', c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$. We show that $\mathrm{DP}_{y_i}(M')$ does not contain a resolvent of one of these two clauses either. This means that M' does not contain one of their resolution descendants, as desired. We only consider E_1, because we can deal with E_2 in the same way. There is no y_i-resolvent of E_1 and a clause C from $\{\{y_i, \overline{b}\}, \{\overline{y_i}, \overline{b}\}, \{y_i, z_i, \overline{b}, b'\}, \{\overline{y_i}, \overline{z_i}, \overline{b}, b^*\}\}$, because $E_1 \cap \overline{C}$ contains b. There is no y_i-resolvent of E_1 and a (resolution descendant from a) clause C of S either, because $E_1 \cap \overline{C}$ contains c_j. □

Lemma 7. *Let M be a yz-reduction formula of F', and let $1 \leq i \leq n$. If var(M) contains y_i and z_i, then both y_i and z_i are DP-simplicial in M.*

Proof. By symmetry, we only have to show that y_i is DP-simplicial in M. Let S be the set of all clauses $D_{j'} \cup \{b, b^*, c_{j'}\} \cup \{\, \overline{c_k} \mid k \neq j' \,\}$ and $\overline{D_{j'}} \cup \{b, b', c_{j'}\} \cup \{\, \overline{c_k} \mid k \neq j' \,\}$ in which y_i occurs. By definition, var(M) contains all b-variables and all c-variables. This has the following two consequences. First, as var(M) also contains y_i and z_i, we find that M contains the clauses $\{y_i, \overline{b}\}$, $\{\overline{y_i}, \overline{b}\}$, $\{y_i, z_i, \overline{b}, b'\}$, and $\{\overline{y_i}, \overline{z_i}, \overline{b}, b^*\}$. Second, by Lemma 6, the other clauses of M in which y_i occurs form a subset of S. This means that there are only 3 pairs of clauses C_1, C_2 in M with $C_1 \cap \overline{C_2} = \{y_i\}$, namely the pair $\{y_i, \overline{b}\}$, $\{\overline{y_i}, \overline{b}\}$, the pair $\{y_i, \overline{b}\}$, $\{\overline{y_i}, \overline{z_i}, \overline{b}, b^*\}$, and the pair $\{\overline{y_i}, \overline{b}\}$, $\{y_i, z_i, \overline{b}, b'\}$. Each of these pairs satisfies (*). This completes the proof of Lemma 7. □

Lemma 8. *Let M be a yz-reduction formula of F'. If M contains neither bcD-clauses nor resolution descendants of such clauses, then M has a DP-simplicial elimination ordering $b, c_1, \ldots, c_m, b', b^*, v^1, \ldots, v^\ell$, where $v^1, \ldots, v^\ell$ form an arbitrary ordering of the y-variables and z-variables in var(M).*

Proof. By our assumptions, the only clauses in M in which b occurs are by-clauses, bz-clauses, byz-clauses, and the clause $\{\overline{b}, b'\}$. In all these clauses b occurs as $\overline{b}$. Hence, b is (trivially) DP-simplicial in M. We then find that $\mathrm{DP}_b(M)$ consists of $\{\overline{b'}, b^*\}$, $\{b', \overline{b^*}\}$ and all bc-clauses. For every c_j, there exists exactly one bc-clause, namely $\{c_j, b', b^*\}$, in which c_j occurs as c_j, and exactly one bc-clause, namely $\{\overline{c_j}, b', b^*\}$, in which c_j occurs as $\overline{c_j}$. Hence, c_j is DP-simplicial in $\mathrm{DP}_{b,c_1,\ldots,c_{j-1}}(M)$ for $j = 1, \ldots, m$. We deduce that $\mathrm{DP}_{b,c^1,\ldots,c^m}(M) = \{\{b', b^*\}, \{\overline{b'}, b^*\}, \{b', \overline{b^*}\}\}$. Then b' is DP-simplicial in $\mathrm{DP}_{b,c_1,\ldots,c_m}(M)$, and we find that $\mathrm{DP}_{b,c_1,\ldots,c_m,b'}(M) = \{\{b^*\}\}$. Then b^* is DP-simplicial in $\mathrm{DP}_{b,c_1,\ldots,c_m,b'}(M)$, and we find that $\mathrm{DP}_{b,c_1,\ldots,c_m,b',b^*}(M) = \emptyset$. Consequently, v^i is DP-simplicial in $\mathrm{DP}_{b,c_1,\ldots,c_m,b',b^*,v^1,\ldots,v^{i-1}}(M)$ for $i = 1, \ldots, \ell$. This concludes the proof of Lemma 8. □

5.2 The Reduction

We are now ready to prove the main result of Section 5.

Theorem 2. *The problem of testing whether a CNF formula belongs to* DPS *is* NP*-complete.*

Proof. Recall that the problem is in NP. Given a CNF formula F that has variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, we construct in polynomial time the CNF formula F'. We claim that F is satisfiable if and only if F' admits a DP-simplicial elimination ordering.

First suppose that F is satisfiable. Let τ be a satisfying truth assignment of F. We define functions f and g that map every x-variable to a y-variable or z-variable in the following way. If $\tau(x_i) = 1$, then $f(x_i) = y_i$ and $g(x_i) = z_i$. If $\tau(x_i) = 0$, then $f(x_i) = z_i$ and $g(x_i) = y_i$. Let $x_1, \ldots, x_n$ be the x-variables in an arbitrary ordering. Then, for every $1 \leq i \leq n$, the formula $\mathrm{DP}_{f(x_1),\ldots,f(x_i)}(F')$ is a yz-reduction formula. From Lemma 7 we deduce that $f(x_i)$ is DP-simplicial in $\mathrm{DP}_{f(x_1),\ldots,f(x_{i-1})}(F')$ for every $1 \leq i \leq n$. Because τ satisfies F, $\mathrm{var}(D_j)$ contains a variable that is not in $\mathrm{var}(\mathrm{DP}_{f(x_1),\ldots,f(x_n)}(F'))$, for every $1 \leq j \leq m$. Lemma 6 implies that M does not contain any bcD-clause or any of their resolution descendants. Then, by Lemma 8, we find that $f(x_1), \ldots, f(x_n), b, c_1, \ldots, c_m, b', b^*, g(x_1), \ldots, g(x_n)$ is a DP-simplicial elimination ordering of F'.

Now suppose that F' admits a DP-simplicial elimination ordering $v^1, \ldots, v^{|\mathrm{var}(F')|}$. Let v^k be the first variable that is neither a y-variable nor a z-variable. Then $M = \mathrm{DP}_{v^1,\ldots,v^{k-1}}(F')$ is a yz-reduction formula. Let $A = \{v^1, \ldots, v^{k-1}\}$, and let X consist of all x-variables that have an associated y-variable or z-variable in A. We define a truth assignment $\tau : X \to \{0, 1\}$ by setting $\tau(x_i) = 1$ if $y_i \in A$ and $\tau(x_i) = 0$ if $z_i \in A$, for every $x_i \in X$. By Lemma 5, we find that τ is well defined. Because v^k is a DP-simplicial b-variable or a DP-simplicial c-variable in M, we can apply Lemma 4 and find that, for every $1 \leq j \leq m$, at least one of the two clauses $D_j \cup \{b, b^*, c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ and $\overline{D_j} \cup \{b, b', c_j\} \cup \{\, \overline{c_k} \mid k \neq j \,\}$ is not in M. This means that every clause C_j contains a literal x with $\tau(x) = 1$. Hence, F is satisfiable. This completes the proof of Theorem 2. □

6 Intermediate Classes

We discuss a possibility for coping with the NP-hardness result of the previous section. The ultimate reason for this hardness is that a formula may have several DP-simplicial variables, and it is hard to choose the right one. A simple workaround is to assume a fixed ordering of the variables and always choose the DP-simplicial variable which comes first according to this ordering. In this way we loose some generality but win polynomial time tractability. This idea is made explicit in the following definitions.

Let Ω denote the set of all strict total orderings of the propositional variables. Let $\prec \in \Omega$ and F be a CNF formula. A variable $x \in \mathrm{var}(F)$ is *$\prec$-DP-simplicial* in F if x is DP-simplicial in F, and $\mathrm{var}(F)$ contains no variable $y \prec x$ that is DP-simplicial in F. A strict total ordering $x_1, \ldots, x_n$ of the variables of F is a *$\prec$-DP-simplicial elimination*

ordering if x_i is $\prec$-DP-simplicial in $\mathrm{DP}_{x_1,\ldots,x_{i-1}}(F)$ for all $1 \leq i \leq n$. We let $\mathsf{DPS}_\prec$ denote the class of all CNF formulas that admit a $\prec$-DP-simplicial elimination ordering, and we set $\mathsf{DPS}_\forall = \bigcap_{\prec\in\Omega} \mathsf{DPS}_\prec$.

Proposition 5. *$\mathsf{DPS}_\prec$ can be recognized in polynomial time for every $\prec \in \Omega$. More precisely, it is possible to find in polynomial time a $\prec$-DP-simplicial elimination ordering for a given CNF formula F, or else to decide that F has no such ordering.*

Proof. Let $x_1, \ldots, x_n$ be the variables of F, ordered according to $\prec$. By brute force we check whether x_i is DP-simplicial in F, for $i = 1, \ldots, n$. This takes polynomial time for each check. When we have found the first DP-simplicial variable x_i, we replace F by $\mathrm{DP}_{x_i}(F)$. We iterate this procedure as long as possible. Let F' be the formula we end up with. If $\mathrm{var}(F') = \emptyset$ then $F \in \mathsf{DPS}_\prec$ and the sequence of variables as they have been eliminated provides a $\prec$-DP-simplicial elimination ordering. If $\mathrm{var}(F') \neq \emptyset$ then $F \notin \mathsf{DPS}_\prec$. □

Proposition 6. $\mathsf{BAC} \subsetneq \mathsf{DPS}_\forall \subsetneq \mathsf{DPS} = \bigcup_{\prec\in\Omega} \mathsf{DPS}_\prec$.

Proof. First we show that $\mathsf{BAC} \subsetneq \mathsf{DPS}_\forall$. Let $F \in \mathsf{BAC}$ and $\prec \in \Omega$. We use induction on the number of variables of F to show that $F \in \mathsf{DPS}_\prec$. The base case $|\mathrm{var}(F)| = 0$ is trivial. Let $|\mathrm{var}(F)| \geq 1$. Because $F \in \mathsf{BAC}$ and $\mathrm{var}(F) \neq \emptyset$, we find that F has at least one weakly simplicial variable. Recall that each weakly simplicial variable is DP-simplicial. Consequently, F has at least one DP-simplicial variable. Let x be the first DP-simplicial variable in the ordering $\prec$. By definition, x is a $\prec$-DP-simplicial variable. We consider $F' = \mathrm{DP}_x(F)$. Because a β-acyclic hypergraph remains β-acyclic under vertex and hyperedge deletion, $F' \in \mathsf{BAC}$. Because F' has fewer variables than F, we use the induction hypothesis to conclude that $F' \in \mathsf{DPS}_\prec$. Hence $\mathsf{BAC} \subseteq \mathsf{DPS}_\prec$ follows. Because $\prec \in \Omega$ was chosen arbitrarily, $\mathsf{BAC} \subseteq \mathsf{DPS}_\forall$ follows.

In order to see that $\mathsf{BAC} \neq \mathsf{DPS}_\forall$, we take a hypergraph H that is not β-acyclic and consider H as a CNF formula with only positive clauses. All variables of H are DP-simplicial and can be eliminated in an arbitrary order. Thus $H \in \mathsf{DPS}_\forall \setminus \mathsf{BAC}$.

Next we show that $\mathsf{DPS}_\forall \subsetneq \mathsf{DPS}$. Inclusion holds by definition. In order to show that the inclusion is strict, we consider the formula F of the example in Section 4.1. In that section we showed that y, b, b', b^*, c, z is a DP-simplicial elimination ordering of F. Hence, $F \in \mathsf{DPS}_\prec$ for any ordering $\prec$ with $y \prec b \prec b' \prec b^* \prec c \prec z$. We also showed that z is DP-simplicial in F but that F has no DP-simplicial ordering starting with z. Hence, $F \notin \mathsf{DPS}_{\prec'}$ for any ordering $\prec'$ with $z \prec' y$. We conclude that $F \in \mathsf{DPS} \setminus \mathsf{DPS}_\forall$. Finally, the equality $\mathsf{DPS} = \bigcup_{\prec\in\Omega} \mathsf{DPS}_\prec$ holds by definition. □

6.1 Grades of Tractability

What properties do we require from a class $\mathcal{C}$ of CNF formulas to be a "tractable class" for SAT? Clearly we want $\mathcal{C}$ to satisfy the property:

1. Given a formula $F \in \mathcal{C}$, we can decide in polynomial time whether F is satisfiable.

This alone is not enough, since even the class of all satisfiable CNF formulas has this property. Therefore we might wish that a tractable class $\mathcal{C}$ should also satisfy the property:

2. Given a formula F, we can decide in polynomial time whether $F \in \mathcal{C}$.

However, if $\mathcal{C}$ is not known to satisfy property 2, then it may still satisfy the property:

3. There exists a polynomial-time algorithm that either decides where a given a formula F is satisfiable or not, or else decides that F does not belong to $\mathcal{C}$.

The algorithm mentioned in property 3 may decide the satisfiability of some formulas outside of $\mathcal{C}$, hereby avoiding the recognition problem. Such algorithms are called *robust algorithms* [29]. In addition we would also assume from a tractable class $\mathcal{C}$ to be closed under isomorphisms, i.e., to satisfy the property:

4. If two formulas differ only in the names of their variables, then either both or none belong to $\mathcal{C}$.

This leaves us with two notions of a tractable class for SAT, a *strict* one where properties 1, 2, and 4 are required, and a *permissive* one where only properties 3 and 4 are required. Every strict class is permissive, but the converse does not hold in general. For instance, the class of Horn formulas is strictly tractable, but the class of extended Horn formulas is only known to be permissively tractable [27].

Where are the classes from our paper located within this classification? As a result of Theorem 1, we find that BAC is strictly tractable. By Theorem 2, DPS is not strictly tractable (unless $\mathsf{P} = \mathsf{NP}$). The classes $\mathsf{DPS}_{\prec}$ do not satisfy property 4. Hence they are not considered as tractable classes. However, $\mathsf{DPS}_{\forall}$ is permissively tractable, because an algorithm for $\mathsf{DPS}_{\prec}$ for an arbitrary ordering $\prec$ is a robust algorithm for $\mathsf{DPS}_{\forall}$. It remains open whether DPS is permissively tractable.

7 Comparisons

We compare the classes of our paper with other known (strictly or permissively) tractable classes. We say that two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of CNF formulas are *incomparable* if for every n larger than some fixed constant there exist formulas in $\mathcal{C}_1 \setminus \mathcal{C}_2$ and in $\mathcal{C}_2 \setminus \mathcal{C}_1$ with at least n variables.

We show that each of the classes mentioned in Proposition 6 is incomparable with a wide range of classes of CNF formulas, in particular with all the tractable classes considered in Speckenmeyer's survey [28], and classes based on graph width parameters [16]. For showing this it suffices to consider the classes BAC and DPS only, which are boundary classes as shown in Proposition 6.

The following four families of formulas will be sufficient for showing most of our incomparability results. Here, $n \geq 1$ is an integer, $x_1, \ldots, x_n$ and $y_1, \ldots, y_{2^n}$ are variables, and $C_1, \ldots, C_{2^n}$ are all possible clauses with variables $x_1, \ldots, x_n$.

$$
\begin{aligned}
F_a(n) &= \{C_1, \ldots, C_{2^n}\} \\
F_s(n) &= \{\{x_1, \ldots, x_{\lceil \frac{n}{2} \rceil}\}, \{x_{\lceil \frac{n}{2} \rceil}, \ldots, x_n\}\} \\
F_c(n) &= \{\, \{x_i, \overline{x}_{i+1}\} \mid 1 \leq i \leq n-1 \,\} \cup \{\{x_n, \overline{x}_1\}\} \\
F_{ac}(n) &= \{\, \{y_{j-1}, \overline{y}_j\} \cup C_j \mid 1 < j \leq 2^n \,\} \cup \{\{y_{2^n}, \overline{y}_1\} \cup C_1\} \cup \\
&\quad \{\, \{y_j, y_{j+1}\} \cup C_j \mid 1 \leq j \leq 2^n \,\} \cup \{\{y_{2^n}, y_1\} \cup C_{2^n}\}.
\end{aligned}
$$

We observe that every $I(F_a(n))$ is a complete bipartite graph with partition classes of size n and 2^n, respectively, and that every $I(F_s(n))$ is a tree. Because complete bipartite graphs and trees are chordal bipartite, we can apply Proposition 2 to obtain the following lemma.

Lemma 9. $F_a(n), F_s(n) \in \mathsf{BAC}$ *for all* $n \geq 1$.

By the following lemma, the other two classes of formulas do not intersect with DPS. Recall that two clauses C and D violate (*) if they have a resolvent that is neither a subset of C nor a subset of D.

Lemma 10. $F_c(n), F_{ac}(n) \notin \mathsf{DPS}$ *for all* $n \geq 3$.

Proof. Throughout the proof we compute indices of modulo n for the vertices x_i, and modulo 2^{n+1} for the vertices y_j.

First we show that $F_c(n) \notin \mathsf{DPS}$. The clauses $C = \{x_i, \overline{x}_{i+1}\}$ and $C' = \{x_{i-1}, \overline{x_i}\} \in F_c(n)$ have the x_i-resolvent $\{x_{i-1}, \overline{x}_{i+1}\}$ which is not a subset of C or C'. Hence, C and C' violate (*). Consequently, x_i is not DP-simplicial for any $1 \leq i \leq n$. Because $F_c(n)$ has no other resolvents, $F_c(n)$ has no DP-simplicial variables. Because $\text{var}(F_c(n)) \neq \emptyset$ either, we conclude that $F_c(n) \notin \mathsf{DPS}$ for all $n \geq 3$.

Next we show that $F_{ac}(n) \notin \mathsf{DPS}$. Let $1 \leq i \leq n$ for some $n \geq 3$. Let $1 \leq j_1, j_2 \leq 2^n$ such that $C_{j_1} \cap \overline{C_{j_2}} = \{x_i\}$. By definition, $F_{ac}(n)$ contains the clauses $C = \{y_{j_1}, y_{j_1+1}\} \cup C_{j_1}$ and $C' = \{y_{j_2}, y_{j_2+1}\} \cup C_{j_2}$, which have x_i-resolvent $C^* = \{y_{j_1}, y_{j_1+1}, y_{j_2}, y_{j_2+1}\} \cup (C_{j_1} \cup C_{j_2}) \setminus \{x_i, \overline{x}_i\}$. However, since $\{y_{j_1}, y_{j_1+1}\} \neq \{y_{j_2}, y_{j_2+1}\}$, we find that C^* is not a subset of C or C'. Hence, C and C' violate (*). Consequently, x_i is not DP-simplicial for any $1 \leq i \leq n$.

Let $1 \leq j \leq 2^n$ for some $n \geq 3$. Then $F_{ac}(n)$ contains the two clauses $C = \{y_j, y_{j+1}\} \cup C_j$ and $C' = \{y_{j-1}, \overline{y}_j\} \cup C_j$, which have y_j-resolvent $C^* = \{y_{j-1}, y_{j+1}\} \cup C_j$. However, $y_{j-1} \in C^* \setminus C$ and $y_{j+1} \in C^* \setminus C'$. Hence, C^* is not a subset of C or C'. Consequently y_j is not DP-simplicial for any $1 \leq j \leq 2^n$. Because $F_{ac}(n)$ has no other resolvents, $F_{ac}(n)$ has no DP-simplicial variables. Because $\text{var}(F_{ac}(n)) \neq \emptyset$ either, we conclude that $F_{ac}(n) \notin \mathsf{DPS}$ for all $n \geq 3$. □

Suppose that we want to show that BAC and DPS are incomparable with a class $\mathcal{C}$ of CNF formulas. Then, Proposition 6 combined with Lemmas 9 and 10 implies that we only have to show the validity of the following two statements:

(i) $F_a(n) \notin \mathcal{C}$ or $F_s(n) \notin \mathcal{C}$ for every n larger than some fixed constant;
(ii) $F_c(n) \in \mathcal{C}$ or $F_{ac}(n) \in \mathcal{C}$ for every n larger than some fixed constant.

7.1 Easy Classes

We use (i) and (ii) to show that BAC and DPS are incomparable with the classes considered by Speckenmeyer [28]. For example, consider the class of *2-CNF formulas*, i.e., CNF formulas where every clause contains at most two literals. For every $n \geq 3$, $F_a(n)$ is not a 2-CNF formula. This shows (i). Furthermore, (ii) follows from the fact

that $F_c(n)$ is a 2-CNF formula for every $n \geq 3$. Consequently, the class of 2-CNF formulas is incomparable with BAC and DPS.

As a second example we consider the class of *hitting formulas*, i.e., CNF formulas where $C \cap \overline{C'} \neq \emptyset$ holds for any two of their clauses [28]. Now, for every $n \geq 3$ the formula $F_s(n)$ is not a hitting formula. This shows (i). It is not difficult to see that for $n \geq 3$, $F_{ac}(n)$ is a hitting formula. This shows (ii). Consequently, the class of hitting formulas is incomparable with BAC and DPS.

The proofs for other classes of formulas considered in [28] are similar. In particular, for the classes *Horn, renameable Horn, extended Horn, CC-balanced, Q-Horn, SLUR, Matched, bounded deficiency, nested, co-nested, and BRLR$_k$ formulas* we can utilize the formulas $F_a(n)$ to show (i) and the formulas $F_c(n)$ to show (ii).

7.2 Classes of Bounded Width

It is known [16] that SAT is tractable for various classes of formulas that are defined by bounding certain width-measures of graphs associated with formulas. Besides the incidence graph $I(F)$ and the directed incidence graph $D(F)$, the other prominent graph associated with a CNF formula F is the *primal graph* $P(F)$ of F, which is the graph with vertex set var(F) and edge set $\{\, x, y \mid x, y \in \text{var}(C) \text{ for some } C \,\}$. We restrict our scope to the graph invariants *treewidth* (tw), and clique-width (cw). Recall that the latter notion has been defined in Section 2. For the definition of treewidth we refer to other sources [16], as we do not need it here.

For a graph invariant π, a graph representation $G \in \{P, I, D\}$ and an integer k, we consider the class $\mathsf{CNF}_k^G(\pi)$ of CNF formulas F with $\pi(G(F)) \leq k$. For every fixed $k \geq 0$, SAT can be solved in polynomial time for the classes $\mathsf{CNF}_k^P(\text{tw})$, $\mathsf{CNF}_k^I(\text{tw})$, and $\mathsf{CNF}_k^D(\text{cw})$ [16]. We show that these classes are incomparable with BAC and DPS.

Proposition 7. *For every $k \geq 2$, $\mathsf{CNF}_k^P(tw)$ is incomparable with* BAC *and* DPS.

Proof. We prove that (i) and (ii) hold with respect to $\mathsf{CNF}_k^P(\text{tw})$. Because $P(F_a(n))$ is the complete graph on n vertices, it has treewidth $n-1$ [1, 18]. Hence, $F_a(n) \notin \mathsf{CNF}_k^P(\text{tw})$ for all $n \geq k+2$. This proves (i). Because $P(F_c(n))$ is a cycle of length n, it has treewidth 2 [1, 18]. Hence, $F_c(n) \in \mathsf{CNF}_2^P(\text{tw})$. This proves (ii). □

Proposition 8. *For every $k \geq 2$, $\mathsf{CNF}_k^I(tw)$ is incomparable with* BAC *and* DPS.

Proof. We prove that (i) and (ii) hold with respect to $\mathsf{CNF}_k^I(\text{tw})$. Because $I(F_a(n))$ is a complete bipartite graph with partition classes of size n and 2^n, respectively, it has treewidth n [1, 18]. Hence, $F_a(n) \notin \mathsf{CNF}_k^I(\text{tw})$ for all $n \geq k+1$. This proves (i). Because $I(F_c(n))$ is a cycle of length $2n$, it has treewidth 2 [1, 18]. Hence, $F_c(n) \in \mathsf{CNF}_2^I(\text{tw})$. This proves (ii). □

Proposition 9. *For every $k \geq 4$, $\mathsf{CNF}_k^D(cw)$ is incomparable with* BAC *and* DPS.

Proof. First we show that $\mathsf{BAC} \setminus \mathsf{CNF}_k^D(\text{cw})$ contains formulas with an arbitrary large number of variables. For all $n \geq 1$, Brandstädt and Lozin [3] showed that there is a bipartite permutation graph $G(n)$ with clique-width n. We do not need the definition of

a bipartite permutation graph; it suffices to know that bipartite permutation graphs are chordal bipartite [29].

Let $G'(n) = (U_n \cup W_n, E_n)$ denote the graph obtained from $G(n)$ by deleting twin vertices as long as possible; two vertices are *twins* if they have exactly the same neighbors. The deletion of twins does not change the clique-width of a graph [6]. Hence, $G'(n)$ has clique-width n. It is well known and easy to see that the clique-width of a bipartite graph with partition classes of size r and s, respectively, is not greater than $\min(r, s) + 2$. Hence $|U_n| \geq n - 2$. Because we only deleted vertices, $G'(n)$ is also chordal bipartite.

Let $F(n) = \{ N(w) \mid w \in W_n \}$ where $N(w)$ denotes the set of neighbors of w in $G'(n)$. Then $G'(n)$ is the incidence graph of $F(n)$, because $G'(n)$ has no twins. Hence $F(n) \in \mathsf{BAC}$ follows from Proposition 2. Recall that the clique-width of $G'(n) = I(F(n))$ is n and that $|U_n| \geq n - 2$. Since all clauses of $F(n)$ are positive, $I(F(n))$ and $D(F(n))$ have the same clique-width. We conclude that $F(n)$ is a formula on at least $n - 2$ variables that belongs to $\mathsf{BAC} \setminus \mathsf{CNF}_k^D(\mathrm{cw})$ for $n \geq k + 1$.

For the converse direction we observe that $D(F_c(n))$ is an oriented cycle and clearly has clique-width at most 4. This means that $D(F_c(n)) \in \mathsf{CNF}_4^D(\mathrm{cw})$. By Lemma 10, we have that $D(F_c(n)) \notin \mathsf{DPS}$ for all $n \geq 3$. We then conclude that $\mathsf{CNF}_4^D(\mathrm{cw}) \setminus \mathsf{DPS}$ contains $D(F_c(n))$ for all $n \geq 3$. We are left to apply Proposition 6 to complete the proof of Proposition 9. □

Results similar to Propositions 7–9 also hold for the graph invariants branchwidth and rank-width, since a class of graphs has bounded branchwidth if and only if it has bounded treewidth [1], and a class of directed graphs has bounded rank-width if and only if it has bounded clique-width [12].

8 Parameterized Complexity

We stud the complexity of SAT for formulas that are "almost" β-acyclic. We define what it means to be almost β-acyclic in two different ways. We base the distance measure on the notion of a strong backdoor set in Section 8.1, and on the notion of β-hypertree width in Section 8.2. We start with a short introduction into Parameterized Complexity and refer to other sources [8, 11] for an in-depth treatment.

A parameterized problem can be considered as a set of pairs (I, k), the instances, where I is the main part and k is the parameter. The parameter is usually a non-negative integer. The complexity class XP consists of parameterized decision problems Π such that for each instance (I, k) it can be decided in $f(k)|I|^{g(k)}$ time whether $(I, k) \in \Pi$, where f and g are computable functions depending only on the parameter k, and $|I|$ denotes the size of I. So XP consists of parameterized decision problems which can be solved in polynomial time if the parameter is a constant. A parameterized decision problem is *fixed-parameter tractable* if there exists a computable function f such that instances (I, k) of size n can be decided in time $f(k)n^{O(1)}$. The class FPT denotes the class of all fixed-parameter tractable decision problems.

Parameterized complexity offers a completeness theory, similar to the theory of NP-completeness, that allows the accumulation of strong theoretical evidence that some

parameterized problems are not fixed-parameter tractable. This theory is based on a hierarchy of complexity classes FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ XP. Each class W[i] contains all parameterized decision problems that can be reduced to a certain fixed parameterized decision problem under *parameterized reductions*. These are many-to-one reductions where the parameter for one problem maps into the parameter for the other. More specifically, problem L reduces to problem L' if there is a mapping R from instances of L to instances of L' such that (i) (I, k) is a yes-instance of L if and only if $(I', k') = R(I, k)$ is a yes-instance of L', (ii) $k' = g(k)$ for a computable function g, and (iii) R can be computed in time $f(k)n^{O(1)}$ where f is a computable function and n denotes the size of (I, k). The class W[1] is considered as the parameterized analog to NP.

8.1 Strong Backdoor Sets

Let $\mathcal{C}$ be a class of CNF formulas. Consider a CNF formula F together with a set of variables $B \subseteq \text{var}(F)$. We say that B is a *strong backdoor set* of F with respect to $\mathcal{C}$ if for all truth assignments $\tau : B \to \{0, 1\}$ we have $F[\tau] \in \mathcal{C}$. In that case we also say that B is a *strong $\mathcal{C}$-backdoor set*. For every CNF formula F and every set $B \subseteq \text{var}(F)$ it holds that F is satisfiable if and only if $F[\tau]$ is satisfiable for at least one truth assignment $\tau : B \to \{0, 1\}$. Thus, if B is a strong $\mathcal{C}$-backdoor set of F, then determining whether F is satisfiable reduces to the SATISFIABILITY problem for at most $2^{|B|}$ reduced CNF formulas $F[\tau] \in \mathcal{C}$.

Now consider a strictly or permissively tractable class $\mathcal{C}$ of CNF formulas. Then, if we have found a strong $\mathcal{C}$-backdoor set of F of size k, deciding the satisfiability of F is fixed-parameter tractable for parameter k. Hence, the key question is whether we can find a strong backdoor set of size at most k if it exists. To study this question, we consider the following parameterized problem; note that this problem belongs to XP for every fixed strictly tractable class $\mathcal{C}$.

STRONG $\mathcal{C}$-BACKDOOR
Instance: A formula F and an integer $k > 0$.
Parameter: The integer k.
Question: Does F have a strong $\mathcal{C}$-backdoor set of size at most k?

It is known that STRONG $\mathcal{C}$-BACKDOOR is fixed-parameter tractable for the class $\mathcal{C}$ of Horn formulas and for the class $\mathcal{C}$ of 2CNF formulas [21]. Contrary to these results, we show that STRONG BAC-BACKDOOR is W[2]-hard.

Theorem 3. *The problem* STRONG BAC-BACKDOOR *is* W[2]*-hard.*

Proof. Let $\mathcal{S}$ be a family of finite sets $S_1, \ldots, S_m$. Then a subset $R \subseteq \bigcup_{i=1}^{m} S_i$ is called a *hitting set* of $\mathcal{S}$ if $R \cap S_i \neq \emptyset$ for $i = 1, \ldots, m$. The HITTING SET problem is defined as follows.

HITTING SET
Instance: A family $\mathcal{S}$ of finite sets $S_1, \ldots, S_m$ and an integer $k > 0$.
Parameter: The integer k.
Question: Does $\mathcal{S}$ have a hitting set of size at most k?

It is well known that HITTING SET is W[2]-complete [8]. We reduce from this problem to prove the theorem.

Let $\mathcal{S} = \{ S_1, \ldots, S_m \}$ and k be an instance of HITTING SET. We write $V(\mathcal{S}) = \bigcup_{i=1}^{m} S_i$ and construct a formula F as follows. For each $s \in V(\mathcal{S})$ we introduce a variable x_s, and we write $X = \{ x_s \mid s \in V(\mathcal{S}) \}$. For each S_i we introduce two variables h_i^1 and h_i^2. Then, for every $1 \leq i \leq m$, the formula F contains three clauses C_i, C_i^1, and C_i^2 such that:

- $C_i = \{ h_i^1, h_i^2 \}$;
- $C_i^1 = \{ h_i^1 \} \cup \{ x_s \mid s \in S_i \} \cup \{ \overline{x}_s \mid s \in V(\mathcal{S}) \setminus S_i) \}$;
- $C_i^2 = \{ h_i^2 \} \cup \{ \overline{x}_s \mid s \in V(\mathcal{S}) \}$.

We need the following claims. The first claim characterizes the induced cycles in $I(F)$ with length at least 6. We need it to prove the second claim.

Claim 1. Let D be an induced cycle in $I(F)$. Then $|V(D)| \geq 6$ if and only if $V(D) = \{h_i^1, h_i^2, x_s, C_i, C_i^1, C_i^2\}$ for some $1 \leq i \leq m$ and $s \in V(\mathcal{S})$.

We prove Claim 1 as follows. Suppose that D is an induced cycle in $I(F)$ with $|V(D)| \geq 6$. By construction, D contains at least one vertex from X. Because any two vertices in X have exactly the same neighbors in $I(F)$, D contains at most one vertex from X. Hence, D contains exactly one vertex from X, let x_s be this vertex. Let C_i^j and $C_{i'}^{j'}$ be the two neighbors of x_s on D. Because x_s is the only of D that belongs to X, we find that h_i^j and $h_{i'}^{j'}$ belong to D. By our construction, C_i and $C_{i'}$ then belong to D as well. If $C_i \neq C_{i'}$, then D contains at least two vertices from X, which is not possible. Hence $C_i = C_{i'}$, as desired. The reverse implication is trivial, and Claim 1 is proven.

Claim 2. Let B be a strong BAC-backdoor set that contains variable h_i^j. Then, for any $s^* \in S_i$, the set $(B \setminus \{h_i^j\}) \cup \{x_{s^*}\}$ is a strong BAC-backdoor set.

We prove Claim 2 as follows. Let $s^* \in S_i$ and define $B' = (B \setminus \{h_i^j\}) \cup \{x_{s^*}\}$. Suppose that B' is not a strong BAC-backdoor set. Then there is a truth assignment $\tau : B' \to \{0,1\}$ with $F[\tau] \notin \mathsf{BAC}$. This means that $I(F[\tau])$ contains an induced cycle D with $|V(D)| \geq 6$. Because B is a strong BAC-backdoor set, h_i^j must belong to $V(D)$. We apply Claim 1 and obtain $V(D) = \{h_i^1, h_i^2, x_s, C_i, C_i^1, C_i^2\}$ for some $x_s \in X$. Suppose $\tau(x_{s^*}) = 1$. Then $C_i^1 \notin F[\tau]$. Hence $\tau(x_{s^*}) = 0$, but then $C_i^2 \notin F[\tau]$. This contradiction proves Claim 2.

We are ready to prove the claim that $\mathcal{S}$ has a hitting set of size at most k if and only if F has a strong BAC-backdoor set of size at most k.

Suppose that $\mathcal{S}$ has a hitting set R of size at most k. We claim that $B = \{ x_s \mid s \in R \}$ is a strong BAC-backdoor set of F. Suppose not. Then there is a truth assignment τ with $F[\tau] \notin \mathsf{BAC}$. This means that $I(F[\tau])$ contains an induced cycle D with $|V(D)| \geq 6$. By Claim 1, we obtain $V(D) = \{h_i^1, h_i^2, x_s, C_i, C_i^1, C_i^2\}$ for some $1 \leq i \leq m$ and $s \in S$. Because C_i^1, C_i^2 are in $I(F[\tau])$, we find that $R \cap S_i = \emptyset$. This is not possible, because R is a hitting set of $\mathcal{S}$.

Conversely, suppose that F has a strong BAC-backdoor set B of size at most k. By Claim 2, we may without loss of generality assume that $B \subseteq X$. We claim that $R = \{ s \mid x_s \in B \}$ is a hitting set of $\mathcal{S}$. Suppose not. Then $R \cap S_i = \emptyset$ for some $1 \leq i \leq m$. This means that B contains no vertex from $\{ x_s \mid s \in S_i \}$. Let $\tau : B \to \{0,1\}$ be the truth assignment with $\tau(x_s) = 1$ for all $x_s \in B$. Then C_i^1 and C_i^2 are in $F[\tau]$. Let $s \in S_i$. Then the cycle D with $V(D) = \{h_i^1, h_i^2, x_s, C_i, C_i^1, C_i^2\}$ is an induced 6-vertex cycle in $I(F[\tau])$. This means that $F[\tau] \notin \mathsf{BAC}$, which is not possible. Hence, we have proven Theorem 3. □

We finish Section 8.1 by considering another type of backdoor sets. Let F be a formula and let $B \subseteq \text{var}(F)$ be a set of variables. Recall that $F - B$ denotes the formula obtained from F after removing all literals x and $\overline{x}$ with $x \in B$ from the clauses in F. We call B a *deletion backdoor set* with respect to a class $\mathcal{C}$ if $F - B \in \mathcal{C}$.

Deletion $\mathcal{C}$-backdoor sets can be seen as a relaxation of strong $\mathcal{C}$-backdoor sets if the class $\mathcal{C}$ is *clause-induced*, i.e., if for every $F \in \mathcal{C}$ and $F' \subseteq F$, we have $F' \in \mathcal{C}$. In that case every deletion $\mathcal{C}$-backdoor set B is also a strong $\mathcal{C}$-backdoor set. This is well known [22] and can easily be seen as follows. Let $\tau : B \to \{0,1\}$ be a truth assignment. Then by definition $F[\tau] \subseteq F - B$. Because B is a deletion $\mathcal{C}$-backdoor set, $F - B \in \mathcal{C}$. Because $\mathcal{C}$ is clause-induced and $F[\tau] \subseteq F - B$, this means that $F[\tau] \in \mathcal{C}$, as required.

Now let $\mathcal{C}$ be a clause-induced class. Let B be a smallest deletion $\mathcal{C}$-backdoor set and let B' be a smallest strong $\mathcal{C}$-backdoor set. Then, from the above, we deduce $|B'| \leq |B|$. The following example shows that $|B| - |B'|$ can be arbitrarily large for $\mathcal{C} = \mathsf{BAC}$, which is obviously clause-induced. Let F be the formula with $\text{var}(F) = \{x_1, \ldots, x_p, y_1, \ldots, y_p, z_1, \ldots, z_p\}$ for some $p \geq 1$ and clauses

$$C_1 = \{x_1, \ldots, x_p, y_1, \ldots, y_p\},$$
$$C_2 = \{\overline{y}_1, \ldots, \overline{y}_p, z_1, \ldots, z_p\},$$
$$C_3 = \{x_1, \ldots, x_p, z_1, \ldots, z_p\}.$$

Then $B = \{y_1\}$ is a smallest strong BAC-backdoor set. However, a smallest deletion BAC-backdoor set must contain at least p variables.

Analogously to the STRONG $\mathcal{C}$-BACKDOOR problem we define the following problem, where $\mathcal{C}$ is a fixed clause-induced class.

DELETION $\mathcal{C}$-BACKDOOR
Instance: A formula F and an integer $k > 0$.
Parameter: The integer k.
Question: Does F have a deletion $\mathcal{C}$-backdoor set of size at most k?

Determining the parameterized complexity of DELETION BAC-BACKDOOR is interesting, especially in the light of our W[2]-hardness result for STRONG BAC-BACKDOOR. In other words, is the problem of deciding whether a graph can be modified into a chordal bipartite graph by deleting at most k vertices fixed-parameter tractable in k? Marx [20] showed that the version of this problem in which the modified graph is required to be chordal instead of chordal bipartite is fixed-parameter tractable.

8.2 β-Hypertree Width

The hypergraph invariant *hypertree width* was introduced by Gottlob, Leone, and Scarcello [14]. It is defined via the notion of a *hypertree decomposition* of a hypergraph H, which is a triple $\mathcal{T} = (T, \kappa, \lambda)$ where T is a rooted tree and χ and λ are labelling functions with $\chi(t) \subseteq V(H)$ and $\lambda(t) \subseteq E(H)$, respectively, for every $t \in V(T)$, such that the following conditions hold:

1. For every $e \in E(H)$ there is a $t \in V(T)$ such that $e \subseteq \chi(t)$.
2. For every $v \in V(H)$, the set $\{\, t \in V(T) \mid v \in \chi(t) \,\}$ induces a connected subtree of T.
3. For every $t \in V(T)$, it holds that $\chi(t) \subseteq \bigcup_{e \in \lambda(t)} e$.
4. For every $t \in V(T)$, if a vertex v occurs in some hyperedge $e \in \lambda(t)$ and if $v \in \chi(t')$ for some node t' in the subtree below t, then $v \in \chi(t)$.

The *width* of a hypertree decomposition (T, χ, λ) is $\max\{\, |\lambda(t)| \mid t \in V(T) \,\}$. The *hypertree width*, denoted $\mathrm{hw}(H)$, of a hypergraph H is the minimum width over all its hypertree decompositions. Many NP-hard problems such as CSP or Boolean database queries can be solved in polynomial time for instances with associated hypergraphs of bounded hypertree width [13].

Gottlob and Pichler [15] defined β-hypertree width as a "hereditary variant" of hypertree width. The *β-hypertree width*, denoted $\beta\text{-hw}(H)$, of a hypergraph H is defined as the maximum hypertree width over all partial hypergraphs H' of H. Using the fact that α-acyclic hypergraphs are exactly the hypergraphs of hypertree width 1 [14], one deduces that the hypergraphs of β-hypertree width 1 are exactly the β-acyclic hypergraphs. Unfortunately, the complexity of determining the β-hypertree width of a hypergraph is not known [15]. However, we show the following. Here, a *β-hypertree decomposition* of width k of a hypergraph H is an oracle that produces for every partial hypergraph H' of H a hypertree decomposition of width at most k.

Theorem 4. SAT*, parameterized by an upper bound k on the β-hypertree width of a CNF formula F, is* W[1]*-hard even if a β-hypertree decomposition of width k for $H(F)$ is given.*

Proof. A *clique* in a graph is a subset of vertices that are mutually adjacent. A k-partite graph is *balanced* if its k partition classes are of the same size. A *partitioned clique* of a balanced k-partite graph $G = (V_1, \ldots, V_k, E)$ is a clique K with $|K \cap V_i| = 1$ for $i = 1 \ldots, k$. We devise a parameterized reduction from the following problem, which is W[1]-complete [25].

PARTITIONED CLIQUE
Instance: A balanced k-partite graph $G = (V_1, \ldots, V_k, E)$.
Parameter: The integer k.
Question: Does G have a partitioned clique?

Before we describe the reduction we introduce some auxiliary concepts. For any three variables z, x_1, x_2, let $F(z, x_1, x_2)$ denote the formula consisting of the clauses

$$\{z, x_1, \overline{x_2}\}, \{z, \overline{x_1}, x_2\}, \{z, \overline{x_1}, \overline{x_2}\}, \{\overline{z}, x_1, x_2\}, \{\overline{z}, \overline{x_1}, \overline{x_2}\}.$$

This formula has exactly three satisfying assignments, corresponding to the vectors 000, 101, and 110. Hence each satisfying assignment sets at most one out of x_1 and x_2 to true, and if one of them is set to true, then z is set to true as well ("$z = x_1 + x_2$"). Taking several instances of this formula we can build a "selection gadget." Let $x_1, \ldots, x_m$ and $z_1, \ldots, z_{m-1}$ be variables. We define $F^{=1}(x_1, \ldots, x_m; z_1, \ldots, z_{m-1})$ as the union of $F(z_1, x_1, x_2)$, $\bigcup_{i=2}^{m-1} F(z_i, z_{i-1}, x_{i+1})$, and $\{\{z_{m-1}\}\}$. Now each satisfying assignment of this formula sets exactly one variable out of $\{x_1, \ldots, x_m\}$ to true, and, conversely, for each $1 \leq i \leq m$ there exists a satisfying assignment that sets exactly x_i to true and all other variables from $\{x_1, \ldots, x_m\}$ to false.

Now we describe the reduction. Let $G = (V_1, \ldots, V_k)$ be a balanced k-partite graph for $k \geq 2$. We write $V_i = \{v_1^i, \ldots, v_n^i\}$. We construct a CNF formula F. As the variables of F we take the vertices of G plus new variables z_j^i for $1 \leq i \leq k$ and $1 \leq j \leq n-1$. We put $F = \bigcup_{i=0}^{k} F_i$ where the formulas F_i are defined as follows: F_0 contains for any $u \in V_i$ and $v \in V_j$ ($i \neq j$) with $uv \notin E$ the clause $C_{u,v} = \{\overline{u}, \overline{v}\} \cup \{w \mid w \in (V_i \cup V_j) \setminus \{u, v\}\}$; for $i > 0$ we define $F_i = F^{=1}(v_1^i, \ldots, v_n^i; z_1^i, \ldots, z_{n-1}^i)$. To prove Theorem 4 it suffices to show the following two claims.

Claim 1. β-hw$(H(F)) \leq k$.

We prove Claim 1 as follows. First we show that that β-hw$(H(F_0)) \leq k$. Let H_0' be a partial hypergraph of $H(F_0)$. Let I be the set of indices $1 \leq i \leq k$ such that some hyperedge of H_0' contains V_i. For each $i \in I$ we choose a hyperedge e_i of H_0' that contains V_i. The partial hypergraph H_0' admits a trivial hypertree decomposition (T_0, χ_0, λ_0) of width at most k with a single tree node t_0 where $\chi_0(t_0)$ contains all vertices of H_0' and $\lambda_0(t_0) = \{e_i \mid i \in I\}$. Second we observe that β-hw$(H(F_i)) = 1$ for $1 \leq i \leq k$: $H(F_i)$ is β-acyclic, and β-acyclic hypergraphs have β-hypertree width 1.

Now let H' be an arbitrarily chosen partial hypergraph of $H(F)$. For $i = 0, \ldots, k$, we let H_i' denote the (maximal) partial hypergraph of H' that is contained in $H(F_i)$. We let $\mathcal{T}_0 = (T_0, \chi_0, \lambda_0)$ be a hypertree decomposition of width at most k of H_0' as defined above. For $i = 1, \ldots, k$ we let $\mathcal{T}_i = (T_i, \chi_i, \lambda_i)$ be a hypertree decomposition of width 1 of H_i'. We combine these $k+1$ hypertree decompositions to a hypertree decomposition of width at most k for H'. We will do this by adding the decompositions $\mathcal{T}_1, \ldots, \mathcal{T}_k$ to $\mathcal{T}_0$ one by one and without increasing the width of $\mathcal{T}_0$.

Let $\mathcal{T}_i^* = (T_i^*, \chi_i^*, \lambda_i^*)$ denote the hypertree decomposition of width at most k obtained from $\mathcal{T}_0$ by adding the first i hypertree decompositions. For $i = 0$ we let $\mathcal{T}_0^* = \mathcal{T}_0$. For $i > 0$ we proceed as follows.

First we consider the case where there is a hyperedge $e \in H_0'$ with $V_{i+1} \subseteq e$. Observe that there exists a node $t \in V(T_i^*)$ with $e \subseteq \chi(t)$. We define $\mathcal{T}_{i+1}^* = (T_{i+1}^*, \chi_{i+1}^*, \lambda_{i+1}^*)$ as follows. We obtain T_{i+1}^* from the disjoint union of T_i^* and T_{i+1} by adding an edge between t and the root of T_{i+1}. As the root of T_{i+1}^* we choose the root of T_i^*. We set $\chi_{i+1}^*(t) = \chi_i^*(t)$ for every $t \in V(T_i^*)$, and $\chi_{i+1}^*(t) = \chi_{i+1}(t) \cup V_{i+1}$ for every $t \in V(T_{i+1})$; we set $\lambda_{i+1}^*(t) = \lambda_i^*(t)$ for every $t \in V(T_i^*)$, and $\lambda_{i+1}^*(t) = \lambda_{i+1}(t) \cup \{e\}$ for every $t \in V(T_{i+1})$ (hence $|\lambda_{i+1}^*(t)| \leq \max(2, k) = k$). Consequently $\mathcal{T}_{i+1}^*$ has width at most k.

It remains to consider the case where there is no hyperedge $e \in H'_0$ with $V_{i+1} \subseteq e$. We define $\mathcal{T}^*_{i+1}$ as follows. We obtain T^*_{i+1} from the disjoint union of T^*_i and T_{i+1} by adding an edge between an arbitrary node $t \in V(T^*_i)$ and the root of T_{i+1}. As the root of T^*_{i+1} we choose the root of T^*_i. We set $\chi^*_{i+1} = \chi^*_i \cup \chi_{i+1}$ and $\lambda^*_{i+1} = \lambda^*_i \cup \lambda_{i+1}$. Clearly $\mathcal{T}^*_{i+1}$ has width at most k. This completes the proof of Claim 1.

Claim 2. G has a partitioned clique if and only if F is satisfiable.

To prove Claim 2 we first suppose that G has a partitioned clique K. We define a partial truth assignment $\tau : V \to \{0,1\}$ by setting $\tau(v) = 1$ for $v \in K$, and $\tau(v) = 0$ for $v \notin K$. This partial assignment satisfies F_0, and it is easy to extend τ to a satisfying truth assignment of F. Conversely, suppose that F has a satisfying truth assignment τ. Because of the formulas F_i, $1 \leq i \leq k$, τ sets exactly one variable $v^i_{j_i} \in V_i$ to true. Let $K = \{v^1_{j_1}, \ldots, v^k_{j_k}\}$. The clauses in F_0 ensure that $v^i_{j_i}$ and $v^{i'}_{j_{i'}}$ are adjacent in G for each pair $1 \leq i < i' \leq k$, hence K is a partitioned clique of G. This proves Claim 2. □

We finish this section by showing some consequences of Theorem 4 with respect to the clique-width and rank-width of a formula. By definition, the clique-width of a CNF formula is always bounded by its directed clique-width. However, in general the directed clique-width can be much higher than the undirected one. It is well known that SAT is fixed-parameter tractable for the parameter directed clique-width [5, 10]. Fischer, Makowsky, and Ravve [10] developed a dynamic programming algorithm that counts the number of satisfying truth assignments in linear time for CNF formulas of bounded directed clique-width. They also conjectured that their method can be extended to work for formulas of bounded (undirected) clique-width. However, the reduction in the proof of Theorem 4 shows that this is not possible unless FPT = W[1].

Corollary 1. SAT*, parameterized by an upper bound k on the clique-width of the incidence graph of a formula F, is* W[1]*-hard even if a k-expression for $I(F)$ is given.*

Proof. We use the same parameterized reduction as in the proof of Theorem 4. Hence it remains to prove that the clique-width of the incidence graph of the formula F in the proof of Theorem 4 is at most $k' = O(k)$. In fact, we show that a $k+4$-expression for the incidence graph of F can be obtained in polynomial time.

We start with the following claim. Let $n \geq 3$, and for $i = 1, \ldots, k$, let T_i be the tree with vertices $C^i_1, \ldots, C^i_{n-1}$, $v^i_1, \ldots, v^i_n$, $z^i_1, \ldots, z^i_{n-1}$, and edges $C^i_1 v^i_1$, $C^i_1 v^i_2$, $C^i_1 z^i_1$, and $C^i_j v^i_{j+1}$, $C^i_j z^i_{j-1}$, $C^i_j z^i_j$ for $j = 2, \ldots, n-1$.

Claim 1. Every T_i allows a 5-expression resulting in a labeling in which every C^i_j has label d, every v^i_j has label i, z^i_{n-1} has label e, whereas every other z^i_j has label d.

Let $1 \leq i \leq k$. We prove Claim 1 by induction on n. Let $n = 3$. We get a desired 5-expression of T_i in the following way. We introduce v^i_1 and v^i_2, each with label i. Then we introduce C^i_1 with label b. We perform the operation $\eta_{b,i}$ resulting in edges between C^i_1 and v^i_1, v^i_2, respectively. We introduce z^i_1 with label c and perform the operation $\eta_{b,c}$ resulting in an edge between C^i_1 and z^i_1. We perform the operation $\rho_{b \to d}$ resulting in a change of label of C^i_1 from b to d. We introduce C^i_2 with label b and perform

the operation $\eta_{b,c}$ resulting in an edge between C_2^i and z_1^i. We perform the operation $\rho_{c\to d}$ resulting in a change of label of z_1^i from c to d. We introduce v_3^i with label c and perform the operation $\eta_{b,c}$ resulting in an edge between C_2^i and v_3^i. We perform the operation $\rho_{c\to i}$ resulting in a change of label of v_3^i from c to i. We introduce z_2^i with label e and perform the operation $\eta_{b,e}$ resulting in an edge between C_2^i and z_2^i. Hence, we have obtained T_3. What is left to do is to perform the operation $\rho_{b\to d}$ resulting in a change of label of C_2^i from b to d.

Let $n \geq 4$. Suppose that we have a labeling of T_{i-1} as in the statement of the claim. Then we do as follows. We introduce C_{n-1}^i with label b and perform the operation $\eta_{b,e}$ resulting in an edge between C_{n-1}^i and z_{n-2}^i. We perform the operation $\rho_{e\to d}$ resulting in a change of label of z_{n-2}^i from e to d. We introduce v_n^i with label c and perform the operation $\eta_{b,c}$ resulting in an edge between C_{n-1}^i and v_n^i. We perform the operation $\rho_{c\to i}$ resulting in a change of label of v_n^i from c to i. We introduce z_{n-1}^i with label e and perform the operation $\eta_{b,e}$ resulting in an edge between C_{n-1}^i and z_{n-1}^i. Hence, we have obtained T_n. What is left to do is to perform the operation $\rho_{b\to d}$ resulting in a change of label of C_{n-1}^i from b to d. This completes the proof of Claim 1.

Note that in the proof of Claim 1 we never performed an operation $\eta_{d,x}$ for some $x \in \{b, c, d, e, i\}$. Hence, we can consider the trees in order $T_1, \ldots, T_k$ to obtain a $(k+4)$-expression for their disjoint union where $v_1^i, \ldots, v_k^i$ are the (only) vertices of label i for $i = 1, \ldots, k$. Moreover, we may assume that all other vertices have label d because we can apply the operation $\rho_{e\to d}$ afterwards. For $s = 1$ and $t = 2$ we now introduce a new vertex $D_{s,t}$ with label b and perform the operations $\eta_{b,s}$, $\eta_{b,t}$ to connect $D_{s,t}$ to every v_i^s and every v_j^t, respectively. Afterwards we perform the operation $\rho_{b\to d}$ resulting in a change of label of $D_{s,t}$ from b to d. In this way, we can add a vertex $D_{s,t}$ for every other index pair $1 \leq s < t \leq k$ as well while using no new labels. We call the resulting graph I'.

We now return to the incidence graph $I(F)$ of the formula F in the proof of Theorem 4. Observe that $I(F)$ can be obtained from I' by adding a number of copies of the vertices C_j^i and $D_{s,t}$. This does not increase the clique-width of I' as explained in the proof of Proposition 9. Hence, the clique-width of $I(F)$ is at most $k+4$, as required. This completes the proof of Corollary 1. □

The already mentioned graph parameter rank-width was introduced by Oum and Seymour [23] for approximating the clique-width of graphs. A certain structure that certifies that a graph has rank-width at most k is called a rank-width decomposition of width k. Similar to clique-width, one can define the rank-width of a directed graph that takes the orientation of edges into account. The *directed* (or *signed*) *rank-width* of a CNF formula is the rank-width of its directed incidence graph. Ganian, Hliněný, and Obdržálek [12] developed an efficient dynamic programming algorithm that counts in linear time the number of satisfying assignments of a CNF formula of bounded *directed* rank-width. Because bounded undirected rank-width implies bounded undirected clique-width [23], the following is a direct consequence of Corollary 1.

Corollary 2. SAT*, parameterized by an upper bound k on the rank-width of the incidence graph of F, is* W[1]*-hard even if a rank-decomposition of width k for $I(F)$ is given.*

9 Conclusion

We have studied new classes of CNF formulas: the strictly tractable class BAC, the permissively tractable class $\mathsf{DPS}_\forall$, and the hard-to-recognize class DPS. Our results show that the classes are incomparable with previously studied classes. Moreover, they establish an interesting link between SAT and algorithmic graph theory: the formulas in BAC are exactly the formulas whose incidence graphs belong to the class of chordal bipartite graphs, a prominent and well-studied graph class. It would be interesting to study systematically other classes of bipartite graphs, e.g., the classes described by Brandstädt, Le and Spinrad [2], in order to determine the complexity of SAT restricted to CNF formulas whose incidence graphs belong to the class under consideration.

We have also established hardness results for two natural strategies for gradually extending BAC: extensions via strong backdoor sets and extensions via β-hypertree decompositions. The first extension is fixed-parameter intractable because it is W[2]-hard to find a strong backdoor set. The second extension is fixed-parameter intractable because SAT is W[1]-hard when parameterized by an upper bound on the β-hypertree width even if the β-hypertree decomposition is provided. It would be interesting to know whether SAT belongs to XP for CNF formulas of bounded β-hypertree width, if a β-hypertree decomposition is provided.

References

1. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1-2):1–45, 1998.
2. Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 1999.
3. Andreas Brandstädt and Vadim V. Lozin. On the linear structure and clique-width of bipartite permutation graphs. *Ars Combinatoria*, 67:273–281, 2003.
4. B. Courcelle, J. Engelfriet, and G. Rozenberg. Context-free handle-rewriting hypergraph grammars. In Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Graph-Grammars and their Application to Computer Science, 4th International Workshop, Bremen, Germany, March 5–9, 1990, Proceedings*, volume 532 of *Lecture Notes in Computer Science*, pages 253–268, 1991.
5. B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discr. Appl. Math.*, 108(1-2):23–52, 2001.
6. B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discr. Appl. Math.*, 101(1-3):77–114, 2000.
7. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.
8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer Verlag, New York, 1999.
9. Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983.
10. E. Fischer, J. A. Makowsky, and E. R. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.*, 156(4):511–529, 2008.
11. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

12. Robert Ganian, Petr Hlinený, and Jan Obdrzálek. Better algorithms for satisfiability problems for formulas of bounded rank-width. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 73–83. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
13. Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: a survey. In *Mathematical Foundations of Computer Science, 2001 (Mariánské Lázně)*, volume 2136 of *Lecture Notes in Computer Science*, pages 37–57. Springer, 2001.
14. Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *J. of Computer and System Sciences*, 64(3):579–627, 2002.
15. Georg Gottlob and Reinhard Pichler. Hypergraphs in model checking: acyclicity and hypertree-width versus clique-width. *SIAM J. Comput.*, 33(2):351–378, 2004.
16. Georg Gottlob and Stefan Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal*, 51(3):303–325, 2006. Survey paper.
17. Peter L. Hammer, Frederic Maffray, and Myriam Preismann. A characterization of chordal bipartite graphs. Technical report, Rutgers University, New Brunswick, NJ, 1989.
18. Ton Kloks and Hans Bodlaender. Approximating treewidth and pathwidth of some classes of perfect graphs. In *Algorithms and Computation (Nagoya, 1992)*, volume 650 of *Lecture Notes in Computer Science*, pages 116–125. Springer Verlag, 1992.
19. Oliver Kullmann and Horst Luckhardt. Algorithms for SAT/TAUT decision based on various measures, manuscript, 1999.
20. Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.
21. Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proceedings of SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada)*, pages 96–103, 2004.
22. Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.
23. Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.
24. Michael J. Pelsmajer, Jacent Tokazy, and Douglas B. West. New proofs for strongly chordal graphs and chordal bipartite graphs. Unpublished Manuscript, 2004.
25. Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. of Computer and System Sciences*, 67(4):757–771, 2003.
26. Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.
27. John S. Schlipf, Fred S. Annexstein, John V. Franco, and R. P. Swaminathan. On finding solutions for extended Horn formulas. *Information Processing Letters*, 54(3):133–137, 1995.
28. Ewald Speckenmeyer. Classes of easy expressions. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, chapter 13, Section 1.19, pages 27–31. IOS Press, 2009.
29. Jeremy P. Spinrad. *Efficient Graph Representations*. Fields Institute Monographs. AMS, 2003.
30. R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984.

31. Ryuhei Uehara. Linear time algorithms on chordal bipartite and strongly chordal graphs. In *Automata, languages and programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 993–1004. Springer, 2002.