Satisfiability of Acyclic and Almost Acyclic CNF Formulas (II)*

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. In the first part of this work (FSTTCS'10) we have shown that the satisfiability of CNF formulas with β-acyclic hypergraphs can be decided in polynomial time. In this paper we continue and extend this work. The decision algorithm for β-acyclic formulas is based on a special type of Davis-Putnam resolution where each resolvent is a subset of a parent clause. We generalize the class of β-acyclic formulas to more general CNF formulas for which this type of Davis-Putnam resolution still applies. We then compare the class of β-acyclic formulas and this superclass with a number of known polynomial formula classes.

1 Introduction

We continue our study [12] of 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.

Because many computationally hard problems can be solved efficiently on acyclic instances, it is a natural to consider SAT for CNF formulas with acyclic hypergraphs. There are several notions of acyclicity for hypergraphs as described by Fagin [6]: α-acyclicity, β-acyclicity, γ-acyclicity, and Berge-acyclicity, which are strictly ordered with respect to their generality, i.e., we have

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

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 [13], and that Berge-ACYC-SAT is solvable in polynomial time [7, 13]. In a recent paper [12] we completed the complexity classification of these four classes by showing that SAT can be solved in polynomial time for β-acyclic formulas, and consequently, for γ-acyclic formulas as well.

New results. The first aim of our paper is to generalize our polynomial-time algorithm for β-acyclic formulas [12]. This algorithm is based on the so-called Davis-Putnam Procedure [5], which successively eliminates variables using Davis-Putnam Resolution. In

* Ordyniak and Szeider's research was funded by the ERC (COMPLEX REASON, 239962). Paulusma's research was funded by EPSRC (EP/G043434/1).

general, this procedure is not efficient, because the number of clauses may increase after each application of Davis-Putnam Resolution. However, the special structure of β-acyclic formulas allows us to compute an elimination ordering of the variables, such that this does not happen. Hence, we can solve SAT in polynomial time for β-acyclic formulas. In fact, the elimination ordering produced this way has the special property that each obtained resolvent is a subset of a parent clause. This type of resolution is known as subsumption resolution [11]. In Section 3 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 4, 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 5, 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.

The second aim of our paper is to make a comparison between the class of β-acyclic formulas and other known polynomial classes of CNF formulas. We do this in Section 6, and our results show that the class of β-acyclic formulas is *incomparable* with all considered classes. Hence, β-acyclic formulas form a new "island of tractability" for SAT.

2 Preliminaries

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 $x \in C \cup \overline{C}$; 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 var$(F) = \bigcup_{C \in F}$ var(C).

Let F be a formula and $X \subseteq$ var(F). A *truth assignment* is a mapping $\tau : X \to \{\, 0, 1 \,\}$ defined on some set X of variables; we write var$(\tau) = X$. For $x \in$ var(τ) we define $\tau(\overline{x}) = 1 - \tau(x)$. 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 $1 \leq j < j' \leq i-1$. 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 [5] considers 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. 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.

3 Generalizing β-Acyclic Formulas

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. A hypergraph is α-*acyclic* if it can be reduced to the empty hypergraph $(\emptyset, \emptyset)$ by repeated application of the following reduction 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 it is α-acyclic and remains α-acyclic after removing an arbitrary number of hyperedges. Thus β-acyclicity is the hereditary variant of α-acyclicity. 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. It is known that SAT is NP-complete for the class of α-acyclic formulas [13]. However, β-acyclicity makes SAT polynomial.

Theorem 1 ([12]) SAT *can be solved in polynomial time for β-acyclic formulas.*

The proof of Theorem 1 is based on the following [12]. A vertex x of a hypergraph H is *weakly simplicial* if the hyperedges of H that contain x form a chain under set inclusion. A nontrivial β-acyclic hypergraph always contains a weakly simplicial vertex. After deletion of this vertex the hypergraph remains β-acyclic. Thus, by repeated deletion of weakly simplicial vertices we can eliminate all vertices of a β-acyclic hypergraph, producing a *weakly simplicial elimination ordering* of its vertices. Because we can find a weakly simplicial vertex in polynomial time, we can compute a weakly simplicial elimination ordering for a β-acyclic hypergraph in polynomial time. Once we have this ordering, we apply the Davis-Putnam procedure. This results in a sequence of formulas with a *non-increasing* number of clauses. As such, the Davis-Putnam procedure runs in polynomial time. Consequently, Theorem 1 holds.

Besides that it is possible to identify a "suitable" vertex in polynomial time, the other key observation in the proof of Theorem 1 is that the number of clauses must not increase by applying Davis-Putnam resolution. We can ensure this by requiring the following property that is more general than being weakly simplicial. Let F be a formula. We say that a variable $x \in \text{var}(F)$ is *DP-simplicial* in 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.

We observe that whenever an x-resolvent is a subset of a parent clause C then it is equal to $C \setminus \{x, \overline{x}\}$. If x is DP-simplicial in F, then $|\text{DP}_x(F)| \leq |F|$, as desired. 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, and we let BAC denote the class of all β-acyclic formulas. 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 3.1, the reverse is not true. Hence, DPS is a proper superclass of BAC.

Given an DP-simplicial ordering, the Davis-Putnam procedure runs in polynomial time. Hence we obtain the following result.

Proposition 1 *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.*

In fact, if a DP-simplicial elimination ordering of the variables in var(F) is given, we can even compute a certificate for the (un)satisfiability of F in polynomial time. This holds, because we can obtain a satisfying truth assignment of F from a satisfying truth assignment of $\text{DP}_x(F)$, and we can obtain a resolution refutation of F from a resolution refutation of $\text{DP}_x(F)$.

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

4 The NP-Completeness Result

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 SATISFIABILITY. In Section 4.1 we construct a CNF formula F' from a given CNF formula F. We also show a number of properties of F'. In Section 4.2 we use these properties to prove that F is satisfiable if and only if F' admits a DP-simplicial elimination ordering.

4.1 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 1 *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 2 *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, var(M') contains all b-variables. Because y_i and z_i are in 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 3 *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 4 *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 3, 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 4. □

Lemma 5 *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 5. □

4.2 The Reduction

We are now ready to prove the main result of Section 4.

Theorem 2 *The problem of deciding whether a given CNF formula admits a DP-simplicial elimination ordering 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 4 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 3 implies that M does not contain any bcD-clause or any of their resolution descendants. Then, by Lemma 5, 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 2, 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 1 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. □

5 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 2 $\mathsf{DPS}_\prec$ *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 $\prec$. We check whether x_i is DP-simplicial in F, for $i = 1, \ldots, n$. Each check is clearly polynomial. 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 3 $\mathsf{BAC} \subsetneq \mathsf{DPS}_\forall \subsetneq \mathsf{DPS} = \bigcup_{\prec \in \Omega} \mathsf{DPS}_\prec$.

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 3.1. 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. □

5.1 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 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 shows 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* [16]. 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 [14].

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 P = 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.

6 Comparisons

We compare the classes of our paper with other known (strictly or permissively) tractable classes. Due to Proposition 3, we only need to consider the boundary classes BAC and DPS. 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 BAC and DPS are each incomparable with a wide range of classes of CNF formulas, in particular with all the tractable classes considered in Speckenmeyer's survey [15], and classes based on graph width parameters [9].

We first introduce some terminology. The *incidence graph* $I(H)$ of a 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 $v \in e$. The *incidence graph* of a formula F is the bipartite graph $I(F)$ with vertex set $\mathrm{var}(F) \cup F$ and edge set $\{\, \{C, x\} \mid C \in F \text{ and } x \in \mathrm{var}(C) \,\}$. A graph is *chordal bipartite* if it is bipartite and has no induced cycle on 6 vertices or more. There exists a useful relationship between β-acyclic formulas and chordal bipartite graphs, due to Tarjan and Yannakakis [17]. They presented this relationship in terms of β-acyclic hypergraphs, whereas we use the formulation of our previous paper [12].

Proposition 4 ([17]) *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.*

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 $x_1, \ldots, x_n$.

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

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 4 to obtain the following lemma.

Lemma 6 $F_a(n), F_s(n) \in \mathsf{BAC}$ *for all* $n \geq 1$.

By the following lemma, the other two classes of formulas do not intersect with DPS.

Lemma 7 $F_c(n), F_{ac}(n) \notin \mathsf{DPS}$ *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 $\mathrm{var}(F_c(n)) \neq \emptyset$ 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 $\mathrm{var}(F_{ac}(n)) \neq \emptyset$ 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 3 combined with Lemmas 6 and 7 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.

6.1 Easy Classes

We use (i) and (ii) to show that BAC and DPS are incomparable with the classes considered by Speckenmeyer [15]. 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 [15]. 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 [15] 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).

6.2 Classes of Bounded Width

The SATISFIABILITY problem is tractable for various classes of formulas that are defined by bounding certain width-measures of graphs associated with formulas. Besides the incidence graph $I(F)$, the two other prominent graphs associated with a CNF formula F are the *primal graph* $P(F)$ and the *directed incidence graph* $D(F)$. The graph $P(F)$ has vertex set var(F) and edge set $\{\, \{x, y\} \mid x, y \in \text{var}(C) \text{ for some } C \,\}$. The graph $D(F)$ is the directed graph with vertex set 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 restrict our scope to the graph invariants *treewidth* (tw), and *clique-width* (cw). For their definitions we refer to other sources [9], as we do not need these definitions 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 $\pi(G(F)) \leq k$. It is known [9] that 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 $\mathsf{CNF}_k^D(\text{cw})$.

Proposition 5 *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, 10]. 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, 10]. Hence, $F_c(n) \in \mathsf{CNF}_2^P(\text{tw})$. This proves (ii). □

Proposition 6 *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, 10]. 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, 10]. Hence, $F_c(n) \in \mathsf{CNF}_2^I(\text{tw})$. This proves (ii). □

Proposition 7 *For every $k \geq 4$, $\mathsf{CNF}_k^D(cw)$ is incomparable with* BAC *and* DPS.

Proof. First we show that $\mathsf{BAC} \setminus \mathsf{CNF}_k^D(\mathrm{cw})$ 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 [16].

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 [4]. 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 4. 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 7, 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 3 to complete the proof of Proposition 7. □

Results similar to Propositions 5-7 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 [8].

7 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. Our results 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 consider other classes of bipartite graphs, e.g., the classes described by Brandstädt, Le and Spinrad [2], and determine the complexity of SAT restricted to CNF formulas whose incidence graphs belong to the class under consideration. Of particular interest are minimal super-classes of the class of chordal bipartite graphs.

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 and S. Olariu. Upper bounds to the clique-width of graphs. *Discr. Appl. Math.*, 101(1-3):77–114, 2000.
5. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.
6. Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983.
7. 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.
8. 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.
9. 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.
10. 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.
11. Oliver Kullmann and Horst Luckhardt. Algorithms for SAT/TAUT decision based on various measures, manuscript, 1999.
12. Sebastian Ordyniak, Daniël Paulusma, and Stefan Szeider. Satisfiability of acyclic and almost acyclic CNF formulas. 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 84–95. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
13. Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.
14. 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.
15. 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.
16. Jeremy P. Spinrad. *Efficient Graph Representations*. Fields Institute Monographs. AMS, 2003.
17. 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.