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@tuwien.ac.at, stefan@szeider.net
2 **School of Engineering and Computing Sciences**
Durham University, Durham, DH1 3LE England
daniel.paulusma@durham.ac.uk

Abstract

We study the propositional satisfiability problem (SAT) on classes of CNF formulas (formulas in Conjunctive Normal Form) that obey certain structural restrictions in terms of their hypergraph structure, by associating to a CNF formula the hypergraph obtained by ignoring negations and considering clauses as hyperedges on variables. We show that satisfiability of CNF formulas with so-called "β-acyclic hypergraphs" can be decided in polynomial time.

We also 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 (*Discr. Appl. Math.* 156, 2008).

Keywords and phrases Satisfiability, chordal bipartite graphs, β-acyclic hypergraphs, backdoor sets, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2010.84

1 Introduction

We study the propositional satisfiability problem (SAT) on classes of CNF formulas (formulas in Conjunctive Normal Form) that obey certain structural restrictions in terms of their hypergraph structure, associating to a CNF formula the hypergraph obtained from the formula by ignoring negations and considering clauses as hyperedges on variables.

Many otherwise hard problems become easy if restricted to acyclic instances. Hence it is natural to ask if SAT becomes polynomial-time tractable for CNF formulas with acyclic hypergraphs. However, in contrast to graphs, there are several notions of acyclicity for hypergraphs: α-acyclicity, β-acyclicity, γ-acyclicity, and Berge acyclicity, as described and discussed by Fagin [5]. We will provide definitions for the notions of acyclicity that are relevant to this paper in Section 2. It is known that the various notions of acyclicity are strictly ordered with respect to their generality, i.e., we have

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

where X-$\textsc{Acyc}$ denotes the class of X-acyclic hypergraphs. Let X-$\textsc{Acyc}$-$\textsc{Sat}$ denote the propositional satisfiability problem restricted to X-acyclic CNF formulas (i.e., CNF formulas

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

IARCS Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010).
Editors: Kamal Lodaya, Meena Mahajan; pp. 84–95

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

whose associated hypergraph is X-acyclic). It is not difficult to see that α-Acyc-Sat is NP-complete (see [19]), and that Berge-Acyc-Sat is solvable in polynomial time. For the latter, observe that if a CNF formula F is Berge-acyclic, then its incidence graph has treewidth 1, thus whether F is satisfiable can be decided in linear time [6, 19].

It is natural to ask where in the chain (1) the exact boundary between NP-completeness and polynomial-time tractability lies. In Section 3 we will answer this question and show that β-Acyc-Sat (and thus also γ-Acyc-Sat) can be solved in polynomial time. We establish this result by combining techniques from structural graph theory (a connection between β-acyclic hypergraphs and chordal bipartite graphs) with a fundamental technique for satisfiability solving (Davis-Putnam resolution).

In Sections 4 and 5 we will explore possibilities to gradually generalize the class of β-acyclic CNF formulas. We will study the parameterized complexity of deciding the satisfiability of formulas parameterized by their "distance" from the class of β-acyclic CNF formulas, with respect to two distance measures.

The first distance measure is based on the notion of *strong backdoor sets*: 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 clearly 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 [11]. 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 [11]. However, we show that even if we are given the CNF formula together with a hypertree decomposition of width k, deciding the satisfiability of F parameterized by k is W[1]-hard. 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 [6].

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 $x \in C \cup \overline{C}$; $\mathsf{var}(C)$ denotes the set of variables which occur in C. For a formula F we put $\mathsf{var}(F) = \bigcup_{C \in F} \mathsf{var}(C)$. Let F be a formula and $X \subseteq \mathsf{var}(F)$. We denote by F_X the set of clauses of F in which some variable of X occurs; i.e., $F_X := \{\, C \in F \mid \mathsf{var}(C) \cap X \neq \emptyset \,\}$.

A *truth assignment* is a mapping $\tau : X \to \{\, 0, 1 \,\}$ defined on some set X of variables; we write $\mathsf{var}(\tau) = X$. For $x \in \mathsf{var}(\tau)$ 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 if the clause contains some literal x with $\tau(x) = 1$; τ satisfies a formula F if it satisfies all clauses of F (i.e., if $F[\tau] = \emptyset$). 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 is to test whether a given CNF formula is satisfiable.

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 $|e| = 2$ for all $e \in E$ then H is also called 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 $\mathsf{var}(F)$ and hyperedge set $\{\, \mathsf{var}(C) \mid C \in F \,\}$. We say that F is *α-acyclic* or *β-acyclic* if $H(F)$ is α-acyclic or β-acyclic, respectively. The *incidence graph* of F is the graph $I(F)$ with vertex set $\mathsf{var}(F) \cup F$ and edge set $\{\, Cx \mid C \in F \text{ and } x \in \mathsf{var}(C) \,\}$. The *directed incidence graph* of F is the directed graph with vertex set $\mathsf{var}(F) \cup F$ 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.

Note that one can make a hypergraph α-acyclic by adding a universal hyperedge that contains all vertices (the first step of the above reduction removes all other hyperedges, the second step all vertices). Using this fact, it is easy to see that SAT is NP-complete for the class of α-acyclic CNF formulas [19]. 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 [9]. Thus SAT and CSP behave differently with respect to acyclicity (representing a clause with k literals as a relational constraint requires exponential space of order $k2^k$).

2.1 Parameterized Complexity

We define the basic notions of Parameterized Complexity and refer to other sources [4, 7] 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. 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 $\mathrm{W}[1], \mathrm{W}[2], \ldots, \mathrm{XP}$. Each class 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.

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

▶ **Theorem 1.** *Satisfiability of β-acyclic CNF formulas can be decided in polynomial time.*

The remainder of this section is devoted to a proof of this result. We need two known results on *chordal bipartite* graphs, which are bipartite graphs with no induced cycle on 6 vertices or more. The first one is Proposition 2. The equivalence between statement (i) and (ii) in this proposition is a result of Tarjan and Yannakakis [20]. The equivalence between statement (ii) and (iii) in Proposition 2 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.** *Let F be a CNF formula. Then the following three statements are equivalent:*

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

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 [21] showed the following (which also follows from results of Hammer, Maffray, and Preismann [12], see [17]). We call a bipartite graph *nontrivial* if it contains at least one edge.

▶ **Proposition 3** (Uehara [21], Hammer, Maffray, and Preismann [12])**.** *A graph is chordal bipartite if and only if every induced subgraph has a weakly simplicial vertex. Furthermore, a nontrivial chordal bipartite graph has a weakly simplicial vertex in each partite set.*

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. The above result immediately gives a polynomial-time procedure for the recognition of β-acyclic hypergraphs: delete weakly simplicial vertices from the hypergraph as long as possible (clearly one can recognize a weakly simplicial vertex in polynomial time). The hypergraph is β-acyclic if and only if we can eliminate in this way all its vertices.

The notion of *Davis-Putnam Resolution* allows us to use this elimination procedure to decide the satisfiability of β-acyclic CNF formulas.

Let C_1, C_2 be clauses such that $C_1 \cap \overline{C}_2 = \{x\}$ for some literal x. The clause $C = (C_1 \cup C_2) \setminus \{x, \overline{x}\}$ is called the *resolvent* of C_1 and C_2 with respect to x. Note that by definition any two clauses have at most one resolvent.

Consider a formula F and a variable x of F. We obtain a formula F' from F by adding all possible resolvents with respect to x, and by removing all clauses in which x occurs. We say that F' is obtained from F by *Davis-Putnam Resolution* with respect to x and we write $\mathrm{DP}_x(F) = F'$. It is well known (and easy to show) that F and $\mathrm{DP}_x(F)$ are equisatisfiable.

The so-called Davis-Putnam Procedure [3] successively eliminates variables in this manner until either the empty formula or a formula which contains the empty clause is obtained. 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. The key observation for our algorithm is that if x is a weakly simplicial variable of F, then the size of $\mathrm{DP}_x(F)$ is not greater than the size of F.

▶ **Lemma 4.** *If x is a weakly simplicial variable of a CNF 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, $\mathsf{var}(C_1) \subseteq \mathsf{var}(C_2)$ or $\mathsf{var}(C_2) \subseteq \mathsf{var}(C_1)$. 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. ◀

Now it is easy to establish Theorem 1. We extend the above elimination procedure. Assume we are given a β-acyclic CNF formula F. Then $I(F)$ is chordal bipartite due to Proposition 2. As long as possible, we select a weakly simplicial variable x and compute $\mathrm{DP}_x(F)$. Because $I(\mathrm{DP}_x(F))$ is an induced subgraph of $I(F)$, it follows that $\mathrm{DP}_x(F)$ is again β-acyclic. Moreover, $\mathrm{DP}_x(F)$ and F are equisatisfiable, and by Lemma 4, $|\mathrm{DP}_x(F)| \leq |F|$. Hence we can repeat this elimination procedure, and by induction, we will finally be left with a CNF formula F_0 that has no variables. If $F_0 = \emptyset$ then F is satisfiable, and if $F_0 = \{\emptyset\}$ then F is unsatisfiable. Because each reduction step can be carried out in polynomial time, Theorem 1 follows.

4 Backdoor Sets

Let $\mathcal{C}$ be a class of CNF formulas. Consider a CNF formula F together with a set of variables $B \subseteq \mathsf{var}(F)$. We say that B is a *strong backdoor set* of F with respect to *base class* $\mathcal{C}$ 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 $B \subseteq \mathsf{var}(F)$ 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 of at most $2^{|B|}$ reduced CNF formulas $F[\tau] \in \mathcal{C}$.

Now consider a base class $\mathcal{C}$ of CNF formulas for which SAT can be solved in polynomial time. 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 that is defined for every fixed base 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 [14]. Let BAC be the class of all β-acyclic CNF formulas. Contrary to the above results, we show that Strong BAC-Backdoor is $W[2]$-hard.

▶ **Theorem 5.** *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 [4]. 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 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 $(B \backslash \{h_i^j\}) \cup \{x_{s^*}\}$ is a strong BAC-backdoor set.

We prove Claim 2 as follows. Let $s^* \in S_i$ and define $B' = (B \backslash \{h_i^j\}) \cup \{x_{s^*}\}$. Suppose B' is not a strong BAC-backdoor set. Then there is a truth assignment $\tau : B' \to \{0, 1\}$ with $F[\tau] \notin$ 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 $\mathcal{S}$ 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 $F[\tau] \notin$ 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 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 $F[\tau] \notin$ BAC, which is not possible. Hence, we have proven Theorem 5. ◀

4.1 Deletion Backdoor Sets

Let F be a formula and let $B \subseteq \mathsf{var}(F)$ be a set of variables. Then $F - B$ 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 base 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 base 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 [15] 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 base 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.

We consider the base class BAC, which is obviously clause-induced. Let F be the formula with $\mathsf{var}(F) = \{x_1, \ldots, x_p, y_1, \ldots, y_p, z_1, \ldots, z_p\}$ for some $p \geq 1$ and clauses $C_1 = \{x_1, \ldots, x_p, y_1, \ldots, y_p\}, C_2 = \{\overline{y}_1, \ldots, \overline{y}_p, z_1, \ldots, z_p\}$ and $C_3 = \{x_1, \ldots, x_p, z_1, \ldots, z_p\}$ for some $p \geq 1$. 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 problem Deletion $\mathcal{C}$-Backdoor problem, where $\mathcal{C}$ is a fixed clause-induced base 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 [13] 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.

5 β-Hypertree Width and Clique-Width

Hypertree width is a hypergraph invariant introduced by Gottlob, Leone, and Scarcello [10]. 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 [9].

Gottlob and Pichler [11] 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 [10], 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 [11]. 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 6.** *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 [18].

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 $\{z, x_1, \overline{x_2}\}$, $\{z, \overline{x_1}, x_2\}$, $\{z, \overline{x_1}, \overline{x_2}\}$, $\{\overline{z}, x_1, x_2\}$, and $\{\overline{z}, \overline{x_1}, \overline{x_2}\}$. 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 $F_i = F^{=1}(v_1^i, \ldots, v_n^i; z_1^i, \ldots, z_{n-1}^i)$.

We will have proven Theorem 6 after showing 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.

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. ◀

Clique-width is a graph parameter that measures in a certain sense the structural complexity of a directed or undirected graph [2]. 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, and the insertion of all possible edges between vertices of label i and label j (either undirected 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. Such a construction of a graph can be represented by an algebraic term called a k-expression.

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*. Observe that 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 satisfiability decision is fixed-parameter tractable for the parameter directed clique-width [1, 6]. Fischer, Makowsky, and Ravve [6] 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 6 shows that this is not possible unless FPT = W[1].

▶ **Corollary 7.** *SAT, parameterized by an upper bound k on the clique-width of the incidence graph of F, is* W[1]*-hard even if a k-expression for $I(F)$ is given.*

Proof. It is easy to see that the clique-width of the incidence graph of the formula F in the proof of Theorem 6 is at most $k' = O(k)$, and a k'-expression can be found in polynomial time. Hence the result follows from the same reduction. ◀

Rank-width is a further graph parameter that has been considered for CNF formulas. This parameter was introduced by Oum and Seymour [16] 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 [8] 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 [16], the following is a direct consequence of Corollary 7.

▶ **Corollary 8.** *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.*

6 Conclusion

We have identified a new class of CNF formulas, the class BAC of β-acyclic formulas, for which recognition and satisfiability decision are solvable in polynomial time. Furthermore, we have established hardness results for two natural strategies for gradually extending this class: extensions via strong backdoor sets and extensions via β-hypertree decompositions. The first extension is fixed-parameter intractable because the backdoor sets are hard to find; the second extension is fixed-parameter intractable because satisfiability decision remains hard even if the β-hypertree decomposition is provided. It would be interesting to know whether the satisfiability of CNF formulas of β-hypertree width bounded by an arbitrary constant k can be decided in polynomial time (of order depending on k) if a β-hypertree decomposition is provided.

References

1 Bruno Courcelle, Johann A. Makowsky, and Udi 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.

2 Bruno Courcelle, Joost Engelfriet, and Grzegorz 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.

3 Martin Davis and Hillary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

4 Rod G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer Verlag, New York, 1999.

5 Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983.

6 Eldar Fischer, Johann A. Makowsky, and Elena V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.*, 156(4):511–529, 2008.

7 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

8 Robert Ganian, Petr Hliněný, and Jan Obdržálek. Better algorithms for satisfiability problems for formulas of bounded rank-width. Technical Report arXiv:1006.5621v1, CoRR, 2010.

9 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.

10 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *J. of Computer and System Sciences*, 64(3):579–627, 2002.

11 Georg Gottlob and Reinhard Pichler. Hypergraphs in model checking: acyclicity and hypertree-width versus clique-width. *SIAM J. Comput.*, 33(2):351–378, 2004.

12 Peter L. Hammer, Frederic Maffray, and Myriam Preismann. A characterization of chordal bipartite graphs. Technical report, Rutgers University, New Brunswick, NJ, 1989.

13 Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.

14 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.

15 Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

16 Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

17 Michael J. Pelsmajer, Jacent Tokazy, and Douglas B. West. New proofs for strongly chordal graphs and chordal bipartite graphs. Unpublished Manuscript, 2004.

18 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.

19 Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

20 Robert E. Tarjan and Mihalis 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.

21 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.