Backdoors to Satisfaction*

Serge Gaspers and Stefan Szeider

Institute of Information Systems, Vienna University of Technology,
A-1040 Vienna, Austria
gaspers@kr.tuwien.ac.at, stefan@szeider.net

Dedicated to Mike Fellows on the occasion of his 60th birthday.

Abstract. A backdoor set is a set of variables of a propositional formula such that fixing the truth values of the variables in the backdoor set moves the formula into some polynomial-time decidable class. If we know a small backdoor set we can reduce the question of whether the given formula is satisfiable to the same question for one or several easy formulas that belong to the tractable class under consideration. In this survey we review parameterized complexity results for problems that arise in the context of backdoor sets, such as the problem of finding a backdoor set of size at most k, parameterized by k. We also discuss recent results on backdoor sets for problems that are beyond NP.

1 Introduction

Satisfiability (SAT) is the classical problem of determining whether a propositional formula in conjunctive normal form (CNF) has a satisfying truth assignment. The famous Cook-Levin Theorem [22,61], stating that SAT is NP-complete, placed satisfiability as the cornerstone of complexity theory. Despite its seemingly specialised nature, satisfiability has proved to be extremely useful in a wide range of different disciplines, both from the practical as well as from the theoretical point of view. Satisfiability provides a powerful and general formalism for solving various important problems including hardware and software verification and planning [8,64,99,56]. Satisfiability is the core of many reasoning problems in automated deduction; for instance, the package dependency management for the OpenSuSE Linux distribution and the autonomous controller for NASA's Deep Space One spacecraft are both based on satisfiability [6,100]. Over the last two decades, SAT-solvers have become amazingly successful in solving formulas with hundreds of thousands of variables that encode problems arising from various application areas, see, e.g., [50]. Theoretical performance guarantees, however, are far from explaining this empirically observed efficiency. In fact, there is an enormous gap between theory and practice. To illustrate it with numbers, take the exponential factor 1.308^n of the currently fastest known

* Research supported by the European Research Council (ERC), project COMPLEX REASON 239962.

H.L. Bodlaender et al. (Eds.): Fellows Festschrift, LNCS 7370, pp. 287–317, 2012.

exact 3SAT algorithm [53]. Already for $n = 250$ variables this number exceeds by far the expected lifetime of the sun in nanoseconds.

Hidden Structure and Parameterized Complexity The discrepancy between theory and practice can be explained by the presence of a certain "hidden structure" in real-world problem instances. It is a widely accepted view that the structure of real-world problem instances makes the problems easy for heuristic solvers. However, classic worst-case analysis is not particularly well-suited to take this hidden structure into account. The classical model is one-dimensional, where only one aspect of the input (its size in bits, or the number of variables for a SAT formula) is taken into account, and it does not differentiate whether or not the instance is otherwise well-structured.

Parameterized Complexity, introduced by Mike Fellows together with Rod Downey offers a two-dimensional theoretical setting. The first dimension is the input size as usual, the second dimension (the parameter) allows to take structural properties of the problem instance into account. The result is a more fine-grained complexity analysis that has the potential of being more relevant to real-world computation while still admitting a rigorous theoretical treatment and firm algorithmic performance guarantees.

There are various ways of defining the "hidden structure" in a problem instance, yielding various ways to parameterize a problem.

Islands of Tractability. One way of coping with the high complexity of important problems within the framework of classical complexity is the identification of tractable sub-problems, i.e., of classes of instances for which the problem can be solved in polynomial time. Each class represents an "island of tractability" within an ocean of intractable problems. For the satisfiability problem, researchers have identified dozens of such islands – one could speak of an archipelago of tractability.

Usually it is quite unlikely that a real-world instance belongs to a known island of tractability, but it may be close to one. A very natural and humble way of parameterizing a problem is hence to take the distance to an island of tractability as a parameter. Guo *et al.* [52] called this approach "distance to triviality". For SAT, the distance is most naturally measured in terms of the smallest number variables that need to be instantiated or deleted such that the instance gets moved to an island of tractability. Such sets of variables are called *backdoor sets* because once we know a small backdoor set we can solve the instance efficiently. Thus backdoor sets provide a "clever reasoning shortcut" through the search space and can be used as an indicator for the presence of a hidden structure in a problem instance. Backdoor sets where independently introduced by Crama *et al.* [27] and by Williams *et al.* [101], the latter authors coined the term "backdoor".

The backdoor set approach to a problem consists of two steps: first a small backdoor set is computed (*backdoor detection*), second the backdoor set is used to solve the problem at hand (*backdoor evaluation*). It is hence natural to consider an upper bound on the size of a smallest backdoor set as a parameter for both backdoor detection and backdoor evaluation.

2 Satisfiability

The *propositional satisfiability problem* (SAT) was the first problem shown to be NP-hard [22,61]. Despite its hardness, SAT solvers are increasingly leaving their mark as a general-purpose tool in areas as diverse as software and hardware verification, automatic test pattern generation, planning, scheduling, and even challenging problems from algebra [50].

A *literal* is a propositional variable x or a negated variable $\neg x$. We also use the notation $x = x^1$ and $\neg x = x^0$. A *clause* is a finite set literals that does not contain a complementary pair x and $\neg x$. A propositional formula in conjunctive normal form, or *CNF formula* for short, is a set of clauses. An rCNF formula is a CNF formula where each clause contains at most r literals. For a clause C we write $\mathrm{var}(C) = \{\, x : x \in C \text{ or } \neg x \in C \,\}$, and for a CNF formula F we write $\mathrm{var}(F) = \bigcup_{C \in F} \mathrm{var}(C)$. An r-CNF formula is a CNF formula where each clause contains at most r literals. For a set S of literals we write $\overline{S} = \{\, x^{1-\epsilon} : x^\epsilon \in S \,\}$. We call a clause C *positive* if $C = \mathrm{var}(C)$ and *negative* if $\overline{C} = \mathrm{var}(C)$.

For a set X of propositional variables we denote by 2^X the set of all mappings $\tau : X \rightarrow \{0,1\}$, the truth assignments on X. For $\tau \in 2^X$ we let $\mathrm{true}(\tau) = \{\, x^{\tau(x)} : x \in X \,\}$ and $\mathrm{false}(\tau) = \{\, x^{1-\tau(x)} : x \in X \,\}$ be the sets of literals set by τ to 1 and 0, respectively. Given a CNF formula F and a truth assignment $\tau \in 2^X$ we define $F[\tau] = \{\, C \setminus \mathrm{false}(\tau) : C \in F,\ C \cap \mathrm{true}(\tau) = \emptyset \,\}$. If $\tau \in 2^{\{x\}}$ and $\epsilon = \tau(x)$, we simple write $F[x = \epsilon]$ instead of $F[\tau]$.

A CNF formula F is *satisfiable* if there is some $\tau \in 2^{\mathrm{var}(F)}$ with $F[\tau] = \emptyset$, otherwise F is *unsatisfiable*. Two CNF formulas are *equisatisfiable* if either both are satisfiable, or both are unsatisfiable. SAT is the NP-complete problem of deciding whether a given CNF formula is satisfiable [22,61].

Islands of Tractability and Backdoors. Backdoors are defined with respect to a fixed class $\mathcal{C}$ of CNF formulas, the *base class* (or *target class*, or more more figuratively, *island of tractability*). From a base class we require the following properties: (i) $\mathcal{C}$ can be recognized in polynomial time, (ii) the satisfiability of formulas in $\mathcal{C}$ can be decided in polynomial time, and (iii) $\mathcal{C}$ is closed under isomorphisms (i.e., if two formulas differ only in the names of their variables, then either both or none belong to $\mathcal{C}$).

Several base classes considered in this survey also satisfy additional properties. Consider a class $\mathcal{C}$ of CNF formulas. $\mathcal{C}$ is *clause-induced* if it is closed under subsets, i.e., if $F \in \mathcal{C}$ implies $F' \in \mathcal{C}$ for each $F' \subseteq F$. $\mathcal{C}$ is *clause-defined* if for each CNF formula F we have $F \in \mathcal{C}$ if and only if $\{C\} \in \mathcal{C}$ for all clauses $C \in F$. $\mathcal{C}$ is closed under *variable-disjoint union* if for any two CNF formulas $F_1, F_2 \in \mathcal{C}$ with $\mathrm{var}(F_1) \cap \mathrm{var}(F_2) = \emptyset$, also $F_1 \cup F_2 \in \mathcal{C}$. $\mathcal{C}$ is *self-reducible* if for any $F \in \mathcal{C}$ and any partial truth assignment τ, also $F[\tau] \in \mathcal{C}$.

A *strong $\mathcal{C}$-backdoor set* of a CNF formula F is a set B of variables such that $F[\tau] \in \mathcal{C}$ for each $\tau \in 2^B$. A *weak $\mathcal{C}$-backdoor set* of F is a set B of variables such that $F[\tau]$ is satisfiable and $F[\tau] \in \mathcal{C}$ holds for some $\tau \in 2^B$. A *deletion $\mathcal{C}$-backdoor set* of F is a set B of variables such that $F - B \in \mathcal{C}$, where $F - B = \{ C \setminus \{x^0, x^1 : x \in B\} : C \in F \}$.

If we know a strong $\mathcal{C}$-backdoor set of F of size k, we can reduce the satisfiability of F to the satisfiability of 2^k formulas in $\mathcal{C}$. Thus SAT becomes fixed-parameter tractable in k. If we know a weak $\mathcal{C}$-backdoor set of F, then F is clearly satisfiable, and we can verify it by trying for each $\tau \in 2^X$ whether $F[\tau]$ is in $\mathcal{C}$ and satisfiable. If $\mathcal{C}$ is clause-induced, any deletion $\mathcal{C}$-backdoor set of F is a strong $\mathcal{C}$-backdoor set of F. For several base classes, deletion backdoor sets are of interest because they are easier to detect than strong backdoor sets. The challenging problem is to find a strong, weak, or deletion $\mathcal{C}$-backdoor set of size at most k if it exists. For each class $\mathcal{C}$ of CNF formulas we consider the following decision problems.

Strong $\mathcal{C}$-Backdoor Set Detection
Instance: A CNF formula F and an integer $k \geq 0$.
Parameter: The integer k.
Question: Does F have a strong $\mathcal{C}$-backdoor set of size at most k?

The problems Weak $\mathcal{C}$-Backdoor Set Detection and Deletion $\mathcal{C}$-Backdoor Set Detection are defined similarly.

In fact, for the backdoor approach we actually need the functional variants of these problems, where if a backdoor set of size at most k exists, such a set is computed. However, for all cases considered in this survey, where backdoor detection is fixed-parameter tractable, the respective algorithms also compute a backdoor set.

We also consider these problems for formulas with bounded clause lengths. All such results are stated for 3CNF formulas, but hold, more generally, for rCNF formulas, where $r \geq 3$ is a fixed integer.

3 Base Classes

In this section we define the base classes for the SAT problem that we will consider in this survey.

3.1 Schaefer's Base Classes

In his seminal paper, Schaefer [93] classified the complexity of generalized satisfiability problems in terms of the relations that are allowed to appear in constraints. For CNF satisfiability, this yields the following five base classes[1].

1. *Horn formulas:* CNF formulas where each clause contains at most one positive literal.
2. *Anti-Horn formulas:* CNF formulas where each clause contains at most one negative literal.

[1] Affine Boolean formulas considered by Schaefer do not correspond naturally to a class of CNF formulas, hence we do not consider them here.

3. *2CNF formulas:* CNF formulas where each clause contains at most two literals.
4. *0-valid formulas:* CNF formulas where each clause contains at least one negative literal.
5. *1-valid formulas:* CNF formulas where each clause contains at least one positive literal.

We denote the respective classes of CNF formulas by $\textsc{Horn}$, $\textsc{Horn}^-$, 2CNF, 0-$\textsc{Val}$, and 1-$\textsc{Val}$, and we write $\mathsf{Schaefer} = \{\textsc{Horn}, \textsc{Horn}^-, \text{2CNF}, \text{0-}\textsc{Val}, \text{1-}\textsc{Val}\}$. We note that all these classes are clause-defined, and by Schaefer's Theorem, these are the only maximal clause-defined base classes. We also note that 0-$\textsc{Val}$ and 1-$\textsc{Val}$ are the only two base classes considered in this survey that are not self-reducible.

3.2 Base Classes Based on Subsolvers

State-of-the-art SAT-solvers are based on variants of the so-called Davis-Logemann-Loveland (DPLL) procedure [29,30] (see also [23]). The DPLL procedure searches systematically for a satisfying assignment, applying first *unit propagation* and *pure literal elimination* as often as possible. Then, DPLL branches on the truth value of a variable, and recurses. The algorithms stops if either there are no clauses left (the original formula is satisfiable) or all branches of the search lead to an empty clause (the original formula is unsatisfiable). Unit propagation takes as input a CNF formula F that contains a "unit clause" $\{x^\epsilon\}$ and outputs $F[x = \epsilon]$. Pure literal elimination takes as input a CNF formula F that has a "pure literal" x^ϵ, where $x \in \text{var}(F)$ and $x^{1-\epsilon} \notin \bigcup_{C \in F} C$, and outputs $F[x = \epsilon]$. In both cases F and $F[x = \epsilon]$ are equisatisfiable. If we omit the branching, we get an incomplete algorithm which decides satisfiability for a subclass of CNF formulas. Whenever the algorithm reaches the branching step, it halts and outputs "give up". This incomplete algorithm is an example of a "subsolver" as considered by Williams *et al.* [101]. The DPLL procedure gives rise to three non-trivial subsolvers: $\text{UP} + \text{PL}$ (unit propagation and pure literal elimination are available), UP (only unit propagation is available), PL (only pure literal elimination is available). We associate each subsolver with the class of CNF formulas for which it determines the satisfiability (this is well-defined, since unit propagation and pure literal elimination are confluent operations). Since the subsolvers clearly run in polynomial time, $\text{UP} + \text{PL}$, UP, and PL form base classes. We write $\mathsf{Subsolver} = \{\text{UP} + \text{PL}, \text{UP}, \text{PL}\}$.

3.3 Miscellaneous Base Classes

Renamable Horn Let X be a set of variables and F a CNF formula. We let $r_X(F)$ denote the CNF formula obtained from F by replacing for every variable $x \in X$, all occurrences of x^ϵ in F with $x^{1-\epsilon}$, for $\epsilon \in \{0, 1\}$. We call $r_X(F)$ a *renaming* of F. Clearly F and $r_X(F)$ are equisatisfiable. A CNF formula is called *renamable Horn* if it has a renaming which is Horn, and we denote the

Table 1. The parameterized complexity of WEAK, STRONG, and DELETION $\mathcal{C}$-BACKDOOR SET DETECTION for various base classes $\mathcal{C}$

Base Class	WEAK	STRONG	DELETION
$\mathcal{C} \in$ Schaefer	W[2]-h [70] (FPT)	FPT [70]	FPT [70]
$\mathcal{C} \in$ Subsolver	W[P]-c [96]	W[P]-c [96]	n/a
FOREST	W[2]-h [46] (FPT [46])	?† (?)	FPT
RHORN	W[2]-h	W[2]-h (?)	FPT [82]
CLU	W[2]-h [71] (FPT)	W[2]-h [71] (FPT [71])	FPT [71]

( ) It is indicated in parentheses if the complexity of the problem for 3CNF formulas is different from general CNF or unknown.
? It is open whether the problem is fixed-parameter tractable.
† Theorem 5 gives an fpt approximation for this problem.
n/a Deletion backdoor sets are undefined for base classes that are not clause-induced.

class of renamable Horn formulas as RHORN. It is easy to see that HORN is a strict subset of RHORN. One can find in polynomial time a Horn renaming of a given CNF formula, if it exists [62]. Hence RHORN is a further base class. In contrast to HORN, RHORN is not clause-defined.

Forests. Many NP-hard problems can be solved in polynomial time for problem instances that are in a certain sense acyclic. The satisfiability problem is no exception. There are various ways of defining a CNF formula to be acyclic. Here we consider acyclicity based on (undirected) incidence graphs: the *incidence graph* of a CNF formula F is the bipartite graph whose vertices are the variables and the clauses of F; a variable x and a clause C are joined by an edge if and only if $x \in \text{var}(C)$. Let FOREST denote the class of CNF formulas whose undirected incidence graphs are forests. It is well known that FOREST forms islands of tractability: the satisfiability of CNF formulas whose incidence graphs have bounded treewidth can be decided in linear time [43,91]. FOREST is the special case of formulas with treewidth at most 1.

Clusters. A CNF formula F is called a *hitting* if any two distinct clauses clash. Two clauses $C, C' \in F$ *clash* if they contain a complimentary pair of literals, i.e., $C \cap \overline{C'} \neq \emptyset$. A CNF formula is called a *clustering formula* if it is a variable disjoint union of hitting formulas. We denote by CLU the class of clustering formulas. Clustering formulas not only allow polynomial-time SAT decision, one can even count the number of satisfying truth assignments in polynomial time. This is due to the fact that each truth assignment invalidates at most one clause of a hitting formula [54,71].

4 Detecting Weak Backdoor Sets

It turns out that for all base classes $\mathcal{C}$ considered in this survey, WEAK $\mathcal{C}$-BACKDOOR SET DETECTION is W[2]-hard. In several cases, restricting the input formula to 3CNF helps, and makes Weak $\mathcal{C}$-Backdoor Set Detection fixed-parameter tractable.

In the proof of the following proposition we use a general approach that entails previously published proofs (such as in [46,70,71]) as special cases.

Proposition 1. Weak $\mathcal{C}$-Backdoor Set Detection *is* W[2]*-hard for all base classes* $\mathcal{C} \in \mathsf{Schaefer} \cup \{\text{RHorn}, \text{Forest}, \text{Clu}\}$.

Proof. We show W[2]-hardness for $\mathcal{C} \in \{$2CNF, Horn, 0-Val, RHorn, Forest, Clu$\}$. The hardness proofs for the remaining two classes 1-Val and Horn$^-$ are symmetric to the proofs for 0-Val and Horn, respectively.

Let G be a CNF formula with a set $X \subseteq \text{var}(G)$ of its variables marked as *external*, all other variables of G are called *internal*. We call G an *or-gadget* for a base class $\mathcal{C}$ if G has the following properties:

1. $G \notin \mathcal{C}$.
2. $G \in$ 1-Val.
3. $G[x = 1] \in \mathcal{C}$ holds for all $x \in X$.
4. For each clause $C \in G$ either $X \subseteq C$ or $\text{var}(C) \cap X = \emptyset$.
5. $\text{var}(G) \setminus X \neq \emptyset$.
6. G can be constructed in time polynomial in $|X|$.

First, we show the following meta-result, and then we define or-gadgets for the different base-classes.

Claim 1: If $\mathcal{C}$ is clause-induced, closed under variable-disjoint union, and has an or-gadget for any number ≥ 1 of external variables, then Weak $\mathcal{C}$-Backdoor Set Detection is W[2]-hard.

We prove the claim by giving a parameterized reduction from the W[2]-complete problem Hitting Set (HS) [33]. Let $(\mathcal{S}, k)$, $\mathcal{S} = \{S_1, \ldots, S_m\}$, be an instance of HS. Let $I = \{1, \ldots, m\} \times \{1, \ldots, k+1\}$. For each S_i we construct $k+1$ or-gadgets $G_i^1, \ldots, G_i^{k+1}$ whose external variables are exactly the elements of S_i, and whose internal variables do not appear in any of the other gadgets $G_{i'}^{j'}$ for $(i', j') \in I \setminus \{(i, j)\}$. Let $F = \bigcup_{(i,j) \in I} G_i^j$. From Property 6 it follows that F can be constructed from $\mathcal{S}$ in polynomial time. We show that $\mathcal{S}$ has a hitting set of size k if and only if F has a weak $\mathcal{C}$-backdoor set of size k.

Assume $B \subseteq \bigcup_{i=1}^m S_i$ is a hitting set of $\mathcal{S}$ of size k. Let $\tau \in 2^B$ the truth assignment that sets all variables from B to 1. By Properties 2 and 3, $G_i^j[\tau]$ is satisfiable and belongs to $\mathcal{C}$ for each $(i, j) \in I$. By Property 4, $\text{var}(G_i^j[\tau]) \cap \text{var}(G_{i'}^{j'}[\tau]) = \emptyset$ for any two distinct pairs $(i, j), (i', j') \in I$. Consequently $F[\tau]$ is satisfiable, and since $\mathcal{C}$ is closed under variable-disjoint union, $F[\tau]$ belongs to $\mathcal{C}$. Thus B is a weak $\mathcal{C}$-backdoor set of F of size k.

Conversely, assume that $B \subseteq \text{var}(F)$ is a weak $\mathcal{C}$-backdoor set of F of size k. Hence, there exists a truth assignment $\tau \in 2^B$ such that $F[\tau]$ is satisfiable and belongs to $\mathcal{C}$. Clearly for each $(i, j) \in I$, $G_i^j[\tau]$ is satisfiable (since $G_i^j[\tau] \subseteq F$), and $G_i^j[\tau] \in \mathcal{C}$ (since $\mathcal{C}$ is clause-induced). However, since $G_i^j \notin \mathcal{C}$ by Property 1, $B \cap \text{var}(G_i^j) \neq \emptyset$ for each $(i, j) \in I$. Let $1 \leq i \leq m$. By construction, F contains

$k+1$ copies $G_i^1 \dots, G_i^{k+1}$ of the same gadget. From Property 5 it follows that all the $k+1$ copies are different. Since $|B| \le k$, there must be some $x_i \in B$ such that there are $1 \le j' < j'' \le k+1$ with $x_i \in \text{var}(G_i^{j'}) \cap \text{var}(G_i^{j''})$. It follows that x_i is an external variable of $G_i^{j'}$, hence $x_i \in B \cap S_i$. Consequently, B is a hitting set of $\mathcal{S}$.

Hence we have indeed a parameterized reduction from HS to WEAK $\mathcal{C}$-BACKDOOR SET DETECTION, and Claim 1 is shown true. We define for each class $\mathcal{C} \in \{$2CNF, HORN, 0-VAL, RHORN, FOREST, CLU$\}$ an or-gadget $F(\mathcal{C})$ where $X = \{x_1, \dots, x_s\}$ is the set of external variables; internal variables are denoted z_i.

- G(2CNF) $= \{X \cup \{z_1, z_2\}\}$.
- G(HORN) $= G$(0-VAL) $= \{X \cup \{z_1\}\}$.
- G(RHORN) $= \{X \cup \{\neg z_1, \neg z_2\}, \{z_1, \neg z_2\}, \{\neg z_1, z_2\}, \{z_1, z_2\}\}$.
- G(FOREST) $= \{X \cup \{\neg z_1, \neg z_2\}, \{z_1, z_2\}\}$.
- G(CLU) $= \{X \cup \{z_1\}, \{z_1\}\}$.

Since the considered classes $\mathcal{C}$ are clearly clause-induced and closed under variable-disjoint union, the proposition now follows from Claim 1. □

For base classes based on subsolvers, weak backdoor set detection is even W[P]-hard. This is not surprising, since the subsolvers allow a propagation through the formula which is similar to the propagation in problems like MINIMUM AXIOM SET or DEGREE 3 SUBGRAPH ANNIHILATOR [33]. The proof of the following theorem is based on a reduction from the W[P]-complete problem CYCLIC MONOTONE CIRCUIT ACTIVATION.

Theorem 1 ([96]). WEAK $\mathcal{C}$-BACKDOOR SET DETECTION *is* W[P]*-complete for all base classes* $\mathcal{C} \in \mathsf{Subsolver}$. *This even holds if the input formula is in 3CNF.*

In summary, we conclude that WEAK $\mathcal{C}$-BACKDOOR SET DETECTION is at least W[2]-hard for all considered base classes. If we restrict our scope to 3CNF formulas, we obtain mixed results.

Proposition 2. *For every clause-defined class* $\mathcal{C}$*,* WEAK $\mathcal{C}$-BACKDOOR SET DETECTION *is fixed-parameter tractable for input formulas in 3CNF.*

Proof. The result follows by a standard bounded search tree argument, sketched as follows. Assume we are given a CNF formula $F \notin \mathcal{C}$ and an integer k. We want to decide whether F has a weak $\mathcal{C}$-backdoor set of size $\le k$. Since $\mathcal{C}$ is clause-defined, F contains a clause C such that $\{C\} \notin \mathcal{C}$. Hence some variable of $\text{var}(C)$ must belong to any weak $\mathcal{C}$-backdoor set of F. There are at most 3 such variables, each of which can be set to true or to false. Hence we branch in at most 6 cases. By iterating this case distinction we build a search tree T, where each node t of T corresponds to a partial truth assignment τ_t. We can stop building the tree at nodes of depth k and at nodes t where $F[\tau_t] \in \mathcal{C}$. It is now easy to see that F has a weak $\mathcal{C}$-backdoor set of size at most k if and only if T has a leaf t such that $F[\tau_t] \in \mathcal{C}$ and $F[\tau_t]$ is satisfiable. For each leaf we can check in polynomial time whether these properties hold. □

In particular, Weak $\mathcal{C}$-Backdoor Set Detection is fixed-parameter tractable for $\mathcal{C} \in \mathsf{Schaefer}$ if the input formula is in 3CNF.

The proof of Proposition 2 can be extended to the class Clu of clustering formulas. Nishimura et al. [71] have shown that a CNF formula is a clustering formula if and only if it does not contain (i) two clauses C_1, C_2 that overlap ($C_1 \cap C_2 \neq \emptyset$) but do not clash ($C_1 \cap \overline{C_2} = \emptyset$), or (ii) three clauses D_1, D_2, D_3 where D_1 and D_2 clash, D_2 and D_3 clash, but D_1 and D_3 do not clash. $\{C_1, C_2\}$ is called an overlap obstruction, $\{D_1, D_2, D_2\}$ is called a clash obstruction. Each weak Clu-backdoor set of a CNF formula F must contain at least one variable from each overlap and each clash obstruction. However, if F is a 3CNF formula, the number of variables of an overlap obstruction is at most 5, and the number of variables of a clash obstruction is at most 7. Hence we can find a weak Clu-backdoor set of size at most k with a bounded search tree, which gives the following result.

Proposition 3. Weak Clu-Backdoor Set Detection *is fixed-parameter tractable for 3CNF formulas.*

Proposition 4. Weak RHorn-Backdoor Set Detection *is* W[2]*-hard, even for 3CNF formulas.*

Proof. Similarly to the proof of Proposition 1 we reduce from HS. As gadgets we use formulas of the form $G = \{\{z_1, \neg x_1, \neg z_2\}, \{z_2, \neg x_2, \neg z_3\}, \ldots, \{z_s, \neg x_s, \neg z_{s+1}\}, \{\neg z_1, z_{s+1}\}, \{\neg z_1, \neg z_{s+1}\}, \{z_1, z_{s+1}\}\}$, where $x_1, \ldots, x_s$ are external variables and $z_1, \ldots, z_{s+1}$ are internal variables. G can be considered as being obtained form the complete formula $\{\,\{z_1^{\epsilon}, z_{s+1}^{\delta}\} : \epsilon, \delta \in \{0,1\}\,\}$ by "subdividing" the clause $\{z_1, \neg z_{s+1}\}$. $G \notin$ RHorn but $G[x_i = 0] \in$ RHorn. In fact, $r_X(G[x_i = 0]) \in$ Horn for $X = \{z_{i+1}, \ldots, z_{s+1}\}$, hence no external variable needs to be renamed. Moreover, we can satisfy $G[x_i = 0]$ by setting all external variables and z_1 to 0, and by setting z_{s+1} to 1.

Let $(\mathcal{S}, k)$, $\mathcal{S} = \{S_1, \ldots, S_m\}$, be an instance of HS. For each S_i we construct $k+1$ gadgets $G_i^1, \ldots, G_i^{k+1}$, each having S_i as the set of its external variables, and the internal variables are new variables only used inside a gadget. We let F to be the union of all such gadgets G_i^j for $1 \leq i \leq m$ and $1 \leq j \leq k+1$.

Similar to the proof of Proposition 1 we can easily show that $\mathcal{S}$ has a hitting set of size k if and only if F has a weak RHorn-backdoor set of size k. The proposition follows. □

According to Propositions 2 and 3, Weak $\mathcal{C}$-Backdoor Set Detection is fixed-parameter tractable for certain base classes $\mathcal{C}$ and input formulas in 3CNF. For the classes $\mathcal{C}$ covered by Propositions 2 and 3 it holds that for every 3CNF formula $F \notin \mathcal{C}$ we can find a set of variables of bounded size, an "obstruction", from which at least one variable must be in any weak $\mathcal{C}$-backdoor set of F. Hence a weak $\mathcal{C}$ backdoor set of size at most k can be found by means of a bounded search tree algorithm. The next result shows that fixed-parameter tractability also prevails for the base class FOREST. However, the algorithm is considerably more complicated, as in this case we do not have obstructions of bounded size.

Theorem 2 ([46]). WEAK FOREST-BACKDOOR SET DETECTION *is fixed-parameter tractable for 3CNF formulas.*

Proof (Sketch). We sketch the fpt algorithm from [46] deciding whether a 3CNF formula has a weak FOREST-backdoor set of size k. We refer to [46] for the full details and the correctness proof. Let G denote the incidence graph of F. The first step of the algorithm runs an fpt algorithm (with parameter k') by Bodlaender [9] that either finds $k' = 2k + 1$ vertex-disjoint cycles in G or a feedback vertex set of G of size at most $12k'^2 - 27k' + 15$.

In case a feedback vertex set X is returned, a tree decomposition of $G \setminus X$ of width 1 is computed and X is added to each bag of this tree decomposition. As the WEAK FOREST-BACKDOOR SET DETECTION problem can be defined in Monadic Second Order Logic, a meta-theorem by Courcelle [26] can use this tree decomposition to conclude.

In case Bodlaender's algorithm returns k' vertex-disjoint cycles, the algorithm finds a set S^* of $O(4^k k^6)$ variables such that any weak FOREST-backdoor set of size k contains at least one variable from S^*. In this case, the algorithm recurses by considering all possibilities of assigning a value to a variable from S^*.

Let $C_1, \ldots, C_{k'}$ denote the variable-disjoint cycles returned by Bodlaender's algorithm. Consider a variable $x \in \text{var}(F)$ and a cycle C. We say that x *kills* C *internally* if $x \in C$. We say that x *kills* C *externally* if $x \notin C$ and C contains a clause $u \in F$ such that $x \in \text{var}(u)$.

As our k' cycles are all vertex-disjoint, at most k cycles may be killed internally. The algorithm goes through all choices of k cycles among $C_1, \ldots, C_{k'}$ that may be killed internally. All other cycles, say $C_1, \ldots, C_{k+1}$, are not killed internally and need to be killed externally. The algorithm now computes a set $S \subseteq \text{var}(F)$ of size $O(k^6)$ such that any weak FOREST-backdoor set of size k, which is a subset of $\text{var}(F) \setminus \bigcup_{i=1}^{k+1} \text{var}(C_i)$, contains at least one variable from S. The union of all such S, taken over all choices of cycles to be killed internally, forms then the set S^* that was to be computed.

For each cycle from $C_1, \ldots, C_{k+1}$, compute its set of external killers in $\text{var}(F) \setminus \bigcup_{i=1}^{k+1} \text{var}(C_i)$. Only these external killers are considered from now on. If one such cycle has no such external killer, then there is no solution with the current specifications and the algorithm backtracks. For each $i, 1 \leq i \leq k+1$, let x_i denote an external killer of C_i with a maximum number of neighbors in C_i. The algorithm executes the first applicable from the following rules.

Multi-Killer Unsupported. If there is an index $i, 1 \leq i \leq k+1$ such that x_i has $\ell \geq 4k$ neighbors in C_i and at most $4k^2 + k$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i, then include all these external killers in S.

Multi-Killer Supported. If there is an index $i, 1 \leq i \leq k+1$ such that x_i has $\ell \geq 4k$ neighbors in C_i and more than $4k^2 + k$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i, then set $S = \{x_i\}$.

Large Overlap. If there are two cycles $C_i, C_j, 1 \leq i \neq j \leq k+1$, with at least $16k^4 + k$ common external killers, then set $S = \emptyset$.

Small Overlap. Otherwise, include in S all vertices that are common external killers of at least two cycles from $C_1, \ldots, C_{k+1}$.

The algorithm recursively checks for each $s \in S^*$ whether the formulas $F[s=0]$ and $F[s=1]$ have a weak FOREST-backdoor set of size $k-1$ and returns YES if any such recursive call was successful and NO otherwise. □

5 Detecting Strong Backdoor Sets

Proposition 5 ([70]). STRONG $\mathcal{C}$-BACKDOOR SET DETECTION *is fixed-parameter tractable for every base class* $\mathcal{C} \in$ Schaefer. *For* $\mathcal{C} \in \{0\text{-VAL}, 1\text{-VAL}\}$, *the problem is even solvable in polynomial time.*

Proof. Consider a CNF formula F. Strong HORN-backdoor sets of F are exactly the vertex covers of the positive primal graph of F, whose vertex set is $\text{var}(F)$, two variables are joined by an edge if they appear together positively in a clause. Strong HORN$^-$-backdoor sets can be characterized symmetrically. Strong 2CNF-backdoor sets of F are exactly the hitting sets of the hypergraph whose vertex set is $\text{var}(F)$ and whose hyperedges are all the subsets $e \subseteq \text{var}(F)$ of size three such that $e \subseteq \text{var}(C)$ for a clause $C \in F$. Thus STRONG $\mathcal{C}$-BACKDOOR SET DETECTION for $\mathcal{C} \in$ {HORN, HORN$^-$, 2CNF} can be accomplished by fpt algorithms for VERTEX COVER [19] and 3-HITTING SET [40]. The smallest strong 1-VAL-backdoor set of F is exactly the union of $\text{var}(C)$ for all negative clauses $C \in F$, the smallest strong 0-VAL-backdoor set of F is exactly the union of $\text{var}(C)$ for all positive clauses $C \in F$. □

Proposition 6. STRONG RHORN-BACKDOOR SET DETECTION *is* W[2]*-hard.*

Proof. The proof uses a reduction from HS similar to the proof of Proposition 1. An instance $(\mathcal{S}, k)$, $\mathcal{S} = \{S_1, \ldots, S_m\}$, of HS is reduced to a formula F which is the union of certain gadgets G_i^j for $1 \leq i \leq m$ and $1 \leq j \leq k+1$. Let $V = \bigcup_{i=1}^m S_i$. A gadget G_i^j contains the four clauses $S_i \cup \{z_1, z_2\}$, $\{z_1, \neg z_2\}$, $\{\neg z_1, z_2\}$, and $\overline{V} \cup \{\neg z_1, \neg z_2\}$, where z_1, z_2 are internal variables that do not occur outside the gadget. Let $B \subseteq V$ be a hitting set of $\mathcal{S}$ and let $\tau \in 2^B$. If τ sets at least one variable to 0, then τ removes from each gadget the only negative clause, hence $r_{\text{var}(F)}(F[\tau]) \in$ HORN. On the other hand, if τ sets all variables from B to 1, then it removes from each gadget the only positive clause (B is a hitting set). Hence, $F[\tau] \in$ HORN in this case. Consequently B is a strong RHORN-backdoor set of F. Conversely, assume B is a strong RHORN-backdoor set of F. Let $\tau \in 2^B$ be the all-1-assignment. For the sake of contradiction, assume there is a set S_i such that $B \cap S_i = \emptyset$. Since $|B| = k$, $B \cap \text{var}(G_i^j) = \emptyset$ for some $1 \leq j \leq k+1$. Now $F[\tau]$ contains the subset $G_i^j[\tau] = \{S_i \cup \{z_1, z_2\}, \{z_1, \neg z_2\}, \{\neg z_1, z_2\}, \{\neg z_1, \neg z_2\}\}$ which is not renamable Horn, hence B is not a strong RHORN-backdoor set of F, a contradiction. Hence B is a hitting set of $\mathcal{S}$. □

It is not known whether Strong Forest-Backdoor Set Detection is fixed-parameter tractable nor whether Strong RHorn-Backdoor Set Detection is fixed-parameter tractable for 3CNF formulas. For the former problem, however, we know at least an fpt approximation [46]; see Theorem 5 below.

The following result is shown by a reduction from Cyclic Monotone Circuit Activation, similarly to Theorem 1.

Theorem 3 ([96]). Strong $\mathcal{C}$-Backdoor Set Detection *is* W[P]*-complete for every base class* $\mathcal{C} \in \mathsf{Subsolver}$*, even for formulas in 3CNF.*

The bounded search tree method outlined above for Weak Clu-Backdoor Set Detection for 3CNF formulas can clearly be adapted for strong backdoors. Hence we get the following result.

Proposition 7. Strong Clu-Backdoor Set Detection *is fixed-parameter tractable for 3CNF formulas.*

5.1 Empty Clause Detection

Dilkina *et al.* [31] suggested to strengthen the concept of strong backdoor sets by means of *empty clause detection.* Let $\mathcal{E}$ denote the class of all CNF formulas that contain the empty clause. For a base class $\mathcal{C}$ we put $\mathcal{C}^{\{\}} = \mathcal{C} \cup \mathcal{E}$; we call $\mathcal{C}^{\{\}}$ the base class obtained from $\mathcal{C}$ by adding empty clause detection. Formulas often have much smaller strong $\mathcal{C}^{\{\}}$-backdoor sets than strong $\mathcal{C}$-backdoor sets [31]. Dilkina *et al.* show that, given a CNF formula F and an integer k, determining whether F has a strong $\text{Horn}^{\{\}}$-backdoor set of size k, is both NP-hard and co-NP-hard (here k is considered just as part of the input and not as a parameter). Thus, the non-parameterized search problem for strong Horn-backdoor sets gets harder when empty clause detection is added. It turns out that also the parameterized problem gets harder when empty clause detection is added.

Theorem 4 ([97]). *For every clause-induced base class* $\mathcal{C}$ *such that at least one satisfiable CNF formula does not belong to* $\mathcal{C}$ *the problem* strong $\mathcal{C}^{\{\}}$-backdoor set *is* W[1]*-hard.*

The theorem clearly applies to all base classes in $\mathsf{Schaefer} \cup \{\text{RHorn}, \text{Forest}\}$. The proof from [97] relies on a reduction from [39], where a reduction to 3CNF formulas is also given. Thus, Theorem 4 also holds for 3CNF formulas.

6 Detecting Deletion Backdoor Sets

In this section we consider the parameterized complexity of Deletion $\mathcal{C}$-Backdoor Set Detection for the various base classes $\mathcal{C}$ from above. For most of the classes the complexity is easily established as follows. For Schaefer classes, strong and deletion backdoor sets coincide, hence the FPT results carry over. The subsolver classes are not clause-induced, hence it does not make sense to consider deletion backdoor sets. Deletion Forest-Backdoor Set Detection can be solved by algorithms for a slight variation of the feedback vertex set problem, and is therefore FPT. One has only to make sure that the feedback vertex set contains only variables and no clauses. This, however, can be achieved by using algorithms for Weighted Feedback Vertex Set [81,17].

It is tempting to use Chen *et al.*'s FPT algorithm for directed feedback vertex set [20] for the detection of deletion backdoor sets. The corresponding base class would contain all CNF formulas with acyclic *directed* incidence graphs (the orientation of edges indicate whether a variable occurs positively or negatively). Unfortunately this class is not suited as a base class since it contains formulas where each clause contains either only positive literals or only negative literals, and SAT is well known to be NP-hard for such formulas [45].

Hence we are left with the classes Clu and RHorn.

For the detection of deletion Clu-backdoor sets we can use overlap obstructions and clash obstructions, as defined before Proposition 3. With each obstruction, we associate a deletion pair which is a pair of sets of variables. With an overlap obstruction $\{C_1, C_2\}$, we associate the deletion pair

$$\{\text{var}(C_1 \cap C_2), \text{var}((C_1 \setminus C_2) \cup (C_2 \setminus C_1))\},$$

and with a clash obstruction $\{D_1, D_2, D_3\}$, we associate the deletion pair

$$\{\text{var}((D_1 \setminus D_3) \cap \overline{D_2}), \text{var}((D_3 \setminus D_1) \cap \overline{D_2})\}.$$

For a formula F, let G_F denote the graph with vertex set $\text{var}(F)$ that has an edge xy if and only if there is a deletion pair $\{X, Y\}$ of F with $x \in X$ and $y \in Y$. Nishimura *et al.* [71] have shown that a set $X \subseteq \text{var}(F)$ is a deletion Clu-backdoor set of F if and only if X is a vertex cover of G_F. Thus, the detection of a deletion Clu-backdoor set of size k can be reduced to the problem of checking whether G_F has a vertex cover of size k, for which there exist very fast algorithms (see for example [19]).

Proposition 8 ([71]). Deletion Clu-Backdoor Set Detection *is fixed-parameter tractable.*

The remaining case is the class RHorn. As noted by Gottlob and Szeider [51] without proof (see also [82]), one can show fixed-parameter tractability of Deletion RHorn-Backdoor Set Detection by reducing it to the problem 2SAT Deletion. The latter problem takes as input a 2CNF formula and an integer k (the parameter), and asks whether one can make the formula satisfiable by deleting at most k clauses. 2SAT Deletion was shown fixed-parameter tractable by Razgon and O'Sullivan [82]. Here we give the above mentioned reduction.

Lemma 1. *There is a parameterized reduction from* Deletion RHorn-Backdoor Set Detection *to* 2SAT Deletion.

Proof. Let (F, k) be a given instance of Deletion RHorn-Backdoor Set Detection. We construct a graph $G = (V, E)$ by taking as vertices all literals

x^ϵ, for $x \in \text{var}(F)$ and $\epsilon \in \{0,1\}$, and by adding two groups of edges. The first group consists of all edges x^0, x^1 for $x \in \text{var}(F)$, the second group consists of all edges $x^\epsilon y^\delta$ for $x, y \in \text{var}(F)$, $\epsilon, \delta \in \{0,1\}$, such that $x^\epsilon, y^\delta \in C$ for some $C \in F$. Observe that the edges of the first group form a perfect matching M of the graph G.

Claim 1. F has a deletion RHORN-backdoor set of size at most k if and only if G has a vertex cover with at most $|M| + k$ vertices.

($\Rightarrow$) Let B be a deletion RHORN-backdoor set of F of size at most k and $X \subseteq \text{var}(F) \setminus B$ such that $r_X(F - B) \in$ HORN. Let $N = \{\, x^0 : x \in \text{var}(F) \setminus X \,\} \cup \{\, x^1 : x \in X \,\}$. Let $K = \{\, x^0, x^1 : x \in B \,\} \cup N$. By definition, $|K| = |M| + |B| \leq |M| + k$. We show that K is a vertex cover of G. Consider an edge $e = x^0 x^1 \in M$ of the first group. If $x \in X$, then $x^1 \in N \subseteq K$ and if $x \notin X$, then $x^0 \in N \subseteq K$. Hence e is covered by K. It remains to consider an edge $f = x^\epsilon y^\delta$ of the second group. If $x \in B$ or $y \in B$, then this edge is covered by K. Hence assume $x, y \notin B$. By construction of G, there is a clause $C \in F$ with $x^\epsilon, y^\delta \in C$. Since $x, y \notin B$, there is also a clause $C' \in F - B$ with $x^\epsilon, y^\delta \in C$. Since C' corresponds to a Horn clause $C'' \in r_X(F - B)$, at least one of the literals x^ϵ, y^δ belongs to N, and hence K covers the edge f. Hence the first direction of Claim 1 follows.

($\Leftarrow$) Let K be a vertex cover of G with at most $|M|+k$ vertices. Let $B \subseteq \text{var}(F)$ be the set of all variables x such that both $x^0, x^1 \in K$. Clearly $|B| \leq k$. Let $X \subseteq \text{var}(F) \setminus B$ such that $x^1 \in K$. We show that $r_X(F - B) \in$ HORN. Let x^δ, y^ϵ be two literals that belong to a clause C'' of $r_X(F - B)$. We show that $\epsilon = 0$ or $\delta = 0$. Let $C' \in F - B$ the clause that corresponds to C'', and let $x^{\epsilon'}, y^{\delta'} \in C'$. It follows that $x^{\epsilon'} y^{\delta'} \in E$, and since K is a vertex cover of G, $x^{\epsilon'} \in K$ or $y^{\delta'} \in K$. If $x^{\epsilon'} \in K$ then $\epsilon = 0$, if $y^{\delta'} \in K$ then $\delta = 0$. Since $x^\delta, y^\epsilon \in C'' \in r_X(F - B)$ were chosen arbitrarily, we conclude that $r_X(F - B) \in$ HORN. Hence Claim 1 is shown.

Mishra *et al.* [68] already observed that a reduction from [16] can be adapted to show that this above-guarantee vertex cover problem can be reduced to 2SAT DELETION. For completeness, we give a reduction here as well.

We construct a 2CNF formula F_2 from G. For each vertex x^ϵ of G we take a variable x_ϵ. For each edge $x^0 x^1 \in M$ we add a negative clause $\{\neg x_0, \neg x_1\}$, and for each edge $x^\epsilon y^\delta \in E \setminus M$ we add a positive clause $\{x_\epsilon, y_\delta\}$.

Claim 2. G has a vertex cover with at most $|M| + k$ vertices if and only if we can delete at most k negative clauses from F_2 to obtain a satisfiable formula.

($\Rightarrow$) Let K be a vertex cover of G. We delete from F_2 all negative clauses $\{\neg x_0, \neg x_1\}$ where both $x_0, x_1 \in K$ (there are at most k such clauses) and obtain a 2CNF formula F_2'. We define a truth assignment $\tau \in 2^{\text{var}(F_2')}$ by setting a variable to 1 if and only if it belongs to K. It remains to show that τ satisfies F_2'. The negative clauses are satisfied since τ sets exactly one literal of a negative clause $\{\neg x_0, \neg x_1\} \in F_2'$ to 1 and exactly one to 0. The positive clauses are satisfied since each positive clause $\{x_\epsilon, y_\delta\}$ corresponds to an edge $x_\epsilon y_\delta \in E$, and since K is a vertex cover, τ sets at least one of the variables x_ϵ, y_δ to 1.

($\Leftarrow$) Let F_2' be a satisfiable formula obtained from F_2 by deleting at most k negative clauses. Let $D = \{\, x \in \text{var}(F) : \{\neg x_0, \neg x_1\} \in F_2 \setminus F_2' \,\}$. Let τ

be a satisfying truth assignment of F_2'. We define a set K of vertices of G by setting $K = \{ x^0, x^1 : x \in D \} \cup \{ x^{\tau(x)} : x \in \text{var}(F) \setminus D \}$, and we observe that $|K| \leq |M| + k$. It remains to show that K is a vertex cover of G. Consider an edge $e = x^0x^1 \in M$ of the first group. If $x \in D$ then $x^0, x^1 \in K$; if $x \notin D$ then $x^{\tau(x)} \in K$, hence e is covered by K. Now consider an edge $f = x^\epsilon y^\delta \in E \setminus M$ of the second group. If $x \in D$ or $y \in D$ then f is clearly covered by K. Hence assume $x, y \notin D$. By definition, there is a positive clause $\{x_\epsilon, y_\delta\} \in F_2' \subseteq F_2$. Since τ satisfies F_2', it follows that $\tau(x_\epsilon) = 1$ or $\tau(y_\delta) = 1$. Consequently $x^\epsilon \in K$ or $y^\delta \in K$, thus K covers f. Hence Claim 2 is shown.

Next we modify F_2 by replacing each positive clause $C = \{x_\epsilon, y_\delta\}$ with $2k+2$ "mixed" clauses $\{x_\epsilon, z_C^i\}, \{\neg z_C^i, y_\delta\}$, for $1 \leq i \leq k+1$, where the z_C^i are new variables. Let F_2^* denote the 2CNF formula obtained this way from F_2.

Claim 3. We can delete at most k negative clauses from F_2 to obtain a satisfiable formula if and only if we can delete at most k clauses from F_2^* to obtain a satisfiable formula.

The claim follows easily from the following considerations. We observe that each pair of mixed clauses $\{x_\epsilon, z_C^i\}, \{\neg z_C^i, y_\delta\}$ is semantically equivalent with $C = \{x_\epsilon, y_\delta\}$. Hence, if F_2 can be made satisfiable by deleting some of the negative clauses, we can also make F_2^* satisfiable by deleting the same clauses. However, deleting some of the mixed clauses does only help if we delete at least one from each of the $k+1$ pairs that correspond to the same clause C. Hence also Claim 3 is shown true. Claims 1–3 together establish the lemma. ☐

Razgon and O'Sullivan's result [82] together with Lemma 1 immediately give the following.

Proposition 9. DELETION RHORN-BACKDOOR SET DETECTION *is fixed-parameter tractable.*

7 Permissive Problems

We consider any function p that assigns nonnegative integers to CNF formulas as a *satisfiability parameter*. In particular we are interested in such satisfiability parameters p for which the following parameterized problem is fixed-parameter tractable:

SAT(p)
Instance: A CNF formula F and an integer $k \geq 0$.
Parameter: The integer k.
Task: Determine whether F is satisfiable or determine that $p(F) > k$.

Note that an algorithm that solves the problem has the freedom of deciding the satisfiability of some formulas F with $p(F) > k$, hence the exact recognition of formulas F with $p(F) \leq k$ can be avoided. Thus SAT(p) is not a usual decision problem, as there are three different outputs, not just two. If SAT(p) is fixed-parameter tractable then we call p an fpt satisfiability parameter, and we say that "the satisfiability of CNF formulas of bounded p is fixed-parameter tractable" (cf. [95]). We write 3SAT(p) if the input is restricted to 3CNF formulas.

Backdoor sets provide a generic way to define satisfiability parameters. Let $\mathcal{C}$ be a base class and F a CNF formula. We define $\mathrm{wb}_{\mathcal{C}}(F)$, $\mathrm{sb}_{\mathcal{C}}(F)$ and $\mathrm{db}_{\mathcal{C}}(F)$ as the size of a smallest weak, strong, and deletion $\mathcal{C}$-backdoor set of F, respectively.

Of course, if the detection of the respective $\mathcal{C}$-backdoor set is fixed-parameter tractable, then $\mathrm{wb}_{\mathcal{C}}$, $\mathrm{sb}_{\mathcal{C}}$, and $\mathrm{db}_{\mathcal{C}}$ are fpt satisfiability parameters. However, it is possible that $\mathrm{wb}_{\mathcal{C}}$, $\mathrm{sb}_{\mathcal{C}}$, or $\mathrm{db}_{\mathcal{C}}$ are fpt satisfiability parameters but the corresponding $\mathcal{C}$-backdoor set detection problem is W[1]-hard. The problems SAT($\mathrm{wb}_{\mathcal{C}}$), SAT($\mathrm{sb}_{\mathcal{C}}$), and SAT($\mathrm{db}_{\mathcal{C}}$) can therefore be considered as more "permissive" versions of the "strict" problems Weak, Strong, and Deletion $\mathcal{C}$-Backdoor Set Detection, the latter require to find a backdoor set even if the given formula is trivially seen to be satisfiable or unsatisfiable. The distinction between permissive and strict versions of problems have been considered in a related context by Marx and Schlotter [66,67] for parameterized k-neighborhood local search. Showing hardness for permissive problems SAT(p) seems to be a much more difficult task than for the strict problems. So far we could establish only few such hardness results.

Proposition 10. SAT($\mathrm{wb}_{\mathcal{C}}$) *is* W[1]*-hard for all* $\mathcal{C} \in \mathsf{Schaefer} \cup \{\mathrm{RHORN}\}$.

Proof. We will show a more general result, that W[1]-hardness holds for all base classes that contain all anti-monotone 2CNF formulas. A CNF formula is *anti-monotone* if all its clauses are negative. Let $\mathcal{C}$ be a base class that contains all anti-monotone 2CNF formulas.

We show that SAT($\mathrm{wb}_{\mathcal{C}}$) is W[1]-hard by reducing from Partitioned Clique, also known as Multicolored Clique. This problem takes as input a k-partite graph and asks whether the graph has a clique on k vertices. The integer k is the parameter. The problem is well-known to be W[1]-complete [78].

Let $H = (V, E)$ with $V = \bigcup_{i=1}^{k} V_i$ be an instance of this problem. We construct a CNF formula F as follows. We consider the vertices of H as variables and add clauses $\{\neg u, \neg v\}$ for any two distinct vertices such that $uv \notin E$. For each $1 \leq i \leq k$, we add the clause V_i. This completes the construction of F.
We show that the following statements are equivalent:

(1) F is satisfiable
(2) H contains a k-clique.
(3) F has a weak $\mathcal{C}$-backdoor set of size at most k.

(1)⇒(2). Let τ be a satisfying assignment of F. Because of the clause V_i, τ sets at least one variable of V_i to 1, for each $1 \leq i \leq k$. As each V_i is an independent set, F contains a clause $\{\neg u, \neg v\}$ for every two distinct vertices in V_i. Thus, τ sets exactly one variable of V_i to 1, for each $1 \leq i \leq k$. The clauses of F also imply that $v_i v_j \in E$ for each $1 \leq i < j \leq k$, since otherwise τ would falsify the clause $\{\neg v_i, \neg v_j\}$. Hence $v_1, \ldots, v_k$ induce a clique in H.

(2)⇒(3). Assume $v_1, \ldots, v_k$ induce a clique in H, with $v_i \in V_i$. We show that $B = \{v_1, \ldots, v_k\}$ is a weak $\mathcal{C}$-backdoor set of F. Let $\tau \in 2^B$ be the truth assignment that sets all variables of B to 1. This satisfies all the clauses $V_i, 1 \leq i \leq k$. Thus, $F[\tau]$ is an anti-monotone 2CNF formula. Therefore it is in $\mathcal{C}$ and it is satisfiable as it is 0-valid. Hence B is a weak $\mathcal{C}$-backdoor set of F.

(3)⇒(1). Any formula that has a weak backdoor set is satisfiable.

Since all three statements are equivalent, we conclude that $\text{SAT}(\text{wb}_\mathcal{C})$ is W[1]-hard. This shows the proposition for the base classes HORN, 2CNF, 0-VAL, and RHORN, as they contain all anti-monotone 2CNF formulas. The hardness for HORN^- and 1-VAL follows by symmetric arguments from the hardness of HORN and 0-VAL, respectively. □

In general, if we have an *fpt approximation algorithm* [14,21,34] for a strict backdoor set detection problem, then the corresponding permissive problem $\text{SAT}(p)$ is fixed-parameter tractable. For instance, if we have an fpt algorithm that, for a given pair (F, k) either outputs a weak, strong, or deletion $\mathcal{C}$-backdoor set of F of size at most $f(k)$ or decides that F has no such backdoor set of size at most k, then clearly $\text{wb}_\mathcal{C}$, $\text{sb}_\mathcal{C}$, and $\text{db}_\mathcal{C}$, respectively, is an fpt satisfiability parameter.

This line of reasoning is used in the next theorem to show that $\text{sb}_{\text{FOREST}}$ is an fpt satisfiability parameter. This result labels FOREST as the first nontrivial base class $\mathcal{C}$ for which $\text{sb}_\mathcal{C}$ is an fpt satisfiability parameter and $\text{sb}_\mathcal{C} \neq \text{db}_\mathcal{C}$. Hence the additional power of strong FOREST-backdoor sets over deletion FOREST-backdoor sets is accessible.

Theorem 5 ([46]). STRONG FOREST-BACKDOOR SET DETECTION *admits a* 2^k *fpt-approximation. Hence* $\text{SAT}(\text{sb}_{\text{FOREST}})$ *is fixed-parameter tractable.*

Proof (Sketch). We sketch the fpt-approximation algorithm from [46] which either concludes that a CNF formula F has no strong FOREST-backdoor set of size k or returns one of size at most 2^k. We refer to [46] for the full details and the correctness proof. Let G denote the incidence graph of F. The first step of the algorithm runs, similarly to the proof of Theorem 2, the fpt algorithm (with parameter k') by Bodlaender [9] that either finds $k' = k^2 2^{k-1} + k + 1$ vertex-disjoint cycles in G or a feedback vertex set of G of size at most $12k'^2 - 27k' + 15$.

In case a feedback vertex set X is returned, a tree decomposition of $G \setminus X$ of width 1 is computed and X is added to each bag of this tree decomposition. As the STRONG FOREST-BACKDOOR SET DETECTION problem can be defined in Monadic Second Order Logic, a meta-theorem by Courcelle [26] can be used to decide the problem in linear time using this tree decomposition.

In case Bodlaender's algorithm returns k' vertex-disjoint cycles, the algorithm finds a set S^* of $O(k^{2k} 2^{k^2-k})$ variables such that every strong FOREST-backdoor set of size k contains at least one variable from S^*. In this case, the algorithm recurses by considering all possibilities of including a variable from S^* in the backdoor set.

Let $C_1, \ldots, C_{k'}$ denote the variable-disjoint cycles returned by Bodlaender's algorithm. Consider a variable $x \in \text{var}(F)$ and a cycle C. We say that x *kills* C *internally* if $x \in C$. We say that x *kills* C *externally* if $x \notin C$ and C contains two clause $u, v \in F$ such that $x \in u$ and $\neg x \in v$. We say in this case that x kills C externally in u and v.

The algorithm goes through all $\binom{k'}{k}$ ways to choose k cycles among $C_1, \ldots, C_{k'}$ that may be killed internally. All other cycles, say $C_1, \ldots, C_{k''}$ with $k'' = k' - k$, are not killed internally. We refer to these cycles as C''-cycles. The algorithm now computes a set $S \subseteq \text{var}(F)$ of size at most 2 such that any strong FOREST-backdoor set of size k, which is a subset of $\text{var}(F) \setminus \bigcup_{i=1}^{k''} \text{var}(C_i)$, contains at least one variable from S. The union of all such S, taken over all choices of cycles to be killed internally, forms then the set S^* that was to be computed.

From now on, consider only killers in $\text{var}(F) \setminus \bigcup_{i=1}^{k''} \text{var}(C_i)$. For each C''-cycle C_i, consider vertices x_i, u_i, v_i such that x_i kills C_i externally in u_i and v_i and there is a path P_i from u_i to v_i along the cycle C_i such that if any variable kills C_i externally in two clauses u'_i and v'_i such that $u'_i, v'_i \in P_i$, then $\{u_i, v_i\} = \{u'_i, v'_i\}$. Note that any variable that does not kill C_i internally, but kills the cycle $Cx_i = P_i \cup \{x_i\}$ also kills the cycle C_i externally in u_i and v_i. We refer to such external killers as *interesting*.

The algorithm executes the first applicable from the following rules.

No External Killer. If there is an index $i, 1 \leq i \leq k''$, such that Cx_i has no external killer, then set $S := \{x_i\}$.

Killing Same Cycles. If there are variables y and z and at least $2^{k-1} + 1$ C''-cycles such that both y and z are interesting external killers of each of these C''-cycles, then set $S := \{y, z\}$.

Killing Many Cycles. If there is a variable y that is an interesting external killer of at least $k \cdot 2^{k-1} + 1$ C''-cycles, then set $S := \{y\}$.

Too Many Cycles Otherwise, set $S = \emptyset$.

For each $s \in S^*$ the algorithm calls itself recursively to compute a strong FOREST-backdoor set for $F[s = 0]$ and for $F[s = 1]$ with parameter $k - 1$. If both recursive calls return backdoor sets, the union of these two backdoor sets and $\{s\}$ is a strong FOREST-backdoor set for F. It returns the smallest such backdoor set obtained for all choices of s, or NO if for each $s \in S^*$ at least one recursive call returned NO. □

Very recently, Theorem 5 has been extended to the base classes NESTED [47] and $\mathcal{W}_{\leq t}$, for every fixed $t \geq 0$ [48]. The class NESTED was introduced by Knuth [59]. It is the class of all CNF formulas whose variables can be linearly ordered such that no pair of clauses straddle each other; a clause c *straddles* a clause c' if there are variables $x, y \in \text{var}(c)$ and $z \in \text{var}(c')$ such that $x < z < y$ in the linear ordering under consideration. The class $\mathcal{W}_{\leq t}$ contains all CNF formulas whose incidence graph has treewidth at most t. These results generalize Theorem 5 since $\mathcal{W}_{\leq 1}$ = FOREST $\subseteq$ NESTED $\subseteq \mathcal{W}_{\leq 3}$. The overall outline of the algorithms from [47,48] resembles the algorithm presented in the proof of Theorem 5, but the case where the incidence graph has large treewidth requires significantly more involved arguments.

8 Comparison of Parameters

Satisfiability parameters can be compared with respect to their generality. Let p, q be satisfiability parameters. We say that p is *at least as general* as q, in symbols $p \preceq q$, if there exists a function f such that for every CNF formula F we have $p(F) \leq f(q(F))$. Clearly, if $p \preceq q$ and SAT(p) is fpt, then so is SAT(q). If $p \preceq q$ but not $q \preceq p$, then p is *more general* than q. If neither $p \preceq q$ nor $q \preceq p$ then p and q are *incomparable*.

As discussed above, each base class $\mathcal{C}$ gives rise to three satisfiability parameters $\mathrm{wb}_{\mathcal{C}}(F)$, $\mathrm{sb}_{\mathcal{C}}(F)$ and $\mathrm{db}_{\mathcal{C}}(F)$. If $\mathcal{C}$ is clause-induced, then $\mathrm{sb}_{\mathcal{C}} \preceq \mathrm{db}_{\mathcal{C}}$; and if $\mathcal{C} \subseteq \mathcal{C}'$, then $\mathrm{sb}_{\mathcal{C}'} \preceq \mathrm{sb}_{\mathcal{C}}$ and $\mathrm{db}_{\mathcal{C}'} \preceq \mathrm{db}_{\mathcal{C}}$.

By associating certain graphs with CNF formulas one can use graph parameters to define satisfiability parameters. The most commonly used graphs are the primal, dual, and incidence graphs. The primal graph of a CNF formula F has as vertices the variables of F, and two variables are adjacent if they appear together in a clause. The dual graph has as vertices the clauses of F, and two clauses C, C' are adjacent if they have a variable in common (i.e., if $\mathrm{var}(C) \cap \mathrm{var}(C') \neq \emptyset$). The incidence graph, as already defined above, is a bipartite graph, having as vertices the variables and the clauses of F; a variable x and a clause C are adjacent if $x \in \mathrm{var}(C)$. The directed incidence graph is obtained from the incidence graph by directing an edge xC from x to C if $x \in C$ and from C to x if $\neg x \in C$.

The treewidth of the primal, dual, and incidence graph gives fpt satisfiability parameters, respectively. The treewidth of the incidence graph is more general than the other two satisfiability parameters [60]. The clique-width of the three graphs provides three more general satisfiability parameters. However, these satisfiability parameters are unlikely fpt: It is easy to see that SAT remains NP-hard for CNF formulas whose primal graphs are cliques, and for CNF formulas whose dual graphs are cliques. Moreover, SAT, parameterized by the clique-width of the incidence graph is W[1]-hard, even if a decomposition is provided [72]. However, the clique-width of directed incidence graphs is an fpt satisfiability parameter which is more general than the treewidth of incidence graphs [25,43].

How do fpt satisfiability parameters based on decompositions and fpt satisfiability parameters based on backdoor sets compare to each other?

Each base class $\mathcal{C}$ considered above, except for the class Forest, contains CNF formulas whose directed incidence graphs have arbitrarily large clique-width. Hence none of the decomposition based parameters is at least as general as the parameters $\mathrm{sb}_{\mathcal{C}}$ and $\mathrm{db}_{\mathcal{C}}$. On the other hand, taking the disjoint union of n copies of a CNF formula multiplies the size of backdoor sets by n but does not increase the width. Hence no backdoor based parameter is more general than decomposition based parameters.

Thus, almost all considered backdoor based fpt satisfiability parameters are incomparable with almost all considered decomposition based fpt satisfiability parameters. A notable exception is the satisfiability parameter $\mathrm{db}_{\textsc{Forest}}$. It is easy to see that the treewidth of the incidence graph of a CNF formula is no greater than the size of a smallest deletion Forest-backdoor set plus one, as the latter forms a feedback vertex set of the incidence graph. Thus the treewidth of incidence graphs is a more general satisfiability parameter than the size of a smallest deletion Forest-backdoor sets. However, one can construct CNF formulas F with $\mathrm{sb}_{\textsc{Forest}}(F) = 1$ whose directed incidence graph has arbitrarily large clique-width. Just take a formula whose incidence graph is a subdivision of a large square grid, and add a further variable x such that on each path which is a subdivision of one edge of the grid there is a clause containing x and a clause containing $\neg x$. Thus, the satisfiability parameter sb_{FOREST}, which is fpt by Theorem 5, is incomparable to all the decomposition based satisfiability parameters considered above.

Figure 1 shows the relationship between some of the discussed fpt satisfiability parameters.

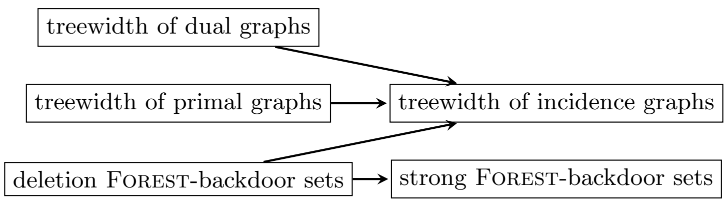

Fig. 1. Relationship between some fpt satisfiability parameters. An arrow from A to B means that B is more general than A. If there is now arrow between A and B then A and B are incomparable.

9 Kernels

The use of strong or deletion backdoor sets for SAT decision, with respect to a base class $\mathcal{C}$, involves two tasks:

1. *backdoor detection*, to find a strong (or deletion) backdoor set of size at most k, or to report that such a backdoor set does not exist,
2. *backdoor evaluation*, to use a given strong (or deletion) backdoor set of size at most k to determine whether the CNF formula under consideration is satisfiable.

In each case where backdoor detection is fixed-parameter tractable, one can now ask whether the detection problem admits a polynomial kernel. For instance, for the classes HORN and 2CNF, backdoor detection can be rephrased as VERTEX COVER or as 3-HITTING SET problems, as discussed above, and therefore admits polynomial kernels [18,1].

Backdoor evaluation is trivially fixed-parameter tractable for any base class, but it is unlikely that it admits a polynomial kernel.

Proposition 11 ([98]). $\mathcal{C}$-BACKDOOR SET EVALUATION *does not admit a polynomial kernel for any self-reducible base class* $\mathcal{C}$ *unless* NP $\subseteq$ co-NP/poly.

This proposition is a trivial consequence of the well-known result that SAT parameterized by the number of variables has no polynomial kernel unless NP $\subseteq$ co-NP/poly [10,44], and the fact that var(F) is always a strong $\mathcal{C}$-backdoor set of F if $\mathcal{C}$ is self-reducible.

Less immediate is the question whether $\mathcal{C}$-Backdoor Set Evaluation admits a polynomial kernel if the inputs are restricted to 3CNF formulas, as 3SAT parameterized by the number of variables has a cubic kernel by trivial reasons. However, for Horn and 2CNF this question can be answered negatively.

Proposition 12 ([98]). $\mathcal{C}$-Backdoor Set Evaluation *does not admit a polynomial kernel for* $\mathcal{C} \in \{\text{Horn}, \text{2CNF}\}$ *unless* NP $\subseteq$ co-NP/poly*, even if the input formula is in 3CNF.*

10 Backdoor Trees

Backdoor trees are binary decision trees on backdoor variables whose leaves correspond to instances of the base class. Every strong backdoor set of size k gives rise to a backdoor tree with at least $k+1$ and at most 2^k leaves. It is reasonable to rank the hardness of instances in terms of the number of leaves of backdoor trees, thus gaining a more refined view than by just comparing the size of backdoor sets.

Consider the CNF formula F with variables $x_1, \ldots, x_{2n}$ and $y_1, \ldots, y_n$ consisting of all clauses of the form

$$\{y_i, \neg x_1, \ldots, \neg x_{2i-2}, x_{2i-1}, \neg x_{2i}, \ldots, \neg x_{2n}\},$$
$$\{y_i, \neg x_1, \ldots, \neg x_{2i-1}, x_{2i}, \neg x_{2i+1}, \ldots, \neg x_{2n}\},$$

for $1 \leq i \leq n$. The set $B = \{y_1, \ldots, y_n\}$ is a strong Horn-backdoor set (in fact, B is the smallest possible). However, every Horn-backdoor tree T with $\text{var}(T) = \{y_1, \ldots, y_n\}$ has 2^n leaves. On the other hand, the formula F has a Horn-backdoor tree T' with only $2n+1$ leaves where $\text{var}(T') = \{x_1, \ldots, x_{2n}\}$. Thus, when we want to minimize the number of leaves of backdoor trees, we must not restrict ourselves to variables of a smallest strong backdoor set.

The problem $\mathcal{C}$-Backdoor Tree Detection now takes as input a CNF formula F, a parameter k, and asks whether F has a $\mathcal{C}$-backdoor tree with at most k leaves.

A base class $\mathcal{C}$ is said to admit a *loss-free kernelization* if there exists a polynomial-time algorithm that, given a CNF formula F and an integer k, either correctly decides that F has no strong $\mathcal{C}$-backdoor set of size at most k, or computes a set $X \subseteq \text{var}(F)$ such that the following conditions hold: (i) X contains all minimal strong $\mathcal{C}$-backdoor sets of F of size at most k; and (ii) the size of X is bounded by a computable function that depends on k only.

Samer and Szeider [88] have shown that $\mathcal{C}$-Backdoor Tree Detection is fixed-parameter tractable for every base class $\mathcal{C}$ that admits a loss-free kernelization. Since Buss-type kernelization is loss-free, the two classes Horn and 2CNF admit a loss-free kernelization. Hence $\mathcal{C}$-Backdoor Tree Detection is fixed-parameter tractable for $\mathcal{C} \in \{\text{2CNF}, \text{Horn}\}$.

11 Backdoors for Problems beyond NP

The backdoor approach has been successfully applied to obtain fixed-parameter tractability for problems whose unparameterized worst-case complexity lies beyond NP. In particular, FPT results have been obtained for the $\#P$-complete problem Propositional Model Counting, the PSPACE-complete QBF-SAT problem, and problems of nonmonotonic reasoning and abstract argumentation that are located on the second level of the Polynomial Hierarchy. In this section we briefly survey these results.

11.1 Propositional Model Counting

The #SAT problem asks to compute for a given CNF formula F the number of assignments $\tau \in 2^{\text{var}(F)}$ that satisfy F. This problem arises in several areas of Artificial Intelligence, in particular in the context of probabilistic reasoning [3,86]. The problem is #P-complete and remains #P-hard even for monotone 2CNF formulas and Horn 2CNF formulas. It is NP-hard to approximate the number of satisfying assignments of a CNF formula with n variables within $2^{n^{1-\epsilon}}$ for any $\epsilon > 0$. This approximation hardness holds also for monotone 2CNF formulas and Horn 2CNF formulas [86]. However, if #SAT can be solved in polynomial time $O(n^c)$ for the formulas of a base class $\mathcal{C}$, and if we know a strong $\mathcal{C}$-backdoor set of a formula F of size k, then we can compute the number of satisfying assignments of F in time $O(2^k n^c)$ [71,90]. For some applications in probabilistic reasoning one is interested in the weighted model counting (WMC) problem, which is more general than #SAT (see, e.g., [92,15]). Since the backdoor set approach applies also to the more general problem, we will use it for the following discussions.

A *weighting* w of a CNF formula F is a mapping w that assigns each variable $x \in \text{var}(F)$ a rational number $0 \leq w(x) \leq 1$; this generalizes to literals by $w(\overline{x}) = 1 - w(x)$ and to truth assignments $\tau \in 2^X$ by $w(\tau) = \prod_{x \in X} w(x^{\tau(x)})$. We define $\#_w(F)$ as the sum of the weights of all assignments $\tau \in 2^{\text{var}(F)}$ that satisfy F. The WMC problem asks to compute $\#_w(F)$ for a given CNF formula F and weighting w. WMC is clearly at least as hard as computing $\#(F)$ as we can reduce #SAT to WMC by using the weight $1/2$ for all n variables and multiplying the result by 2^n. A strong $\mathcal{C}$-backdoor set X of a CNF formula F can be used to compute $\#_w(F)$ via the equation

$$\#_w(F) = \sum_{\tau \in 2^X} w(\tau) \cdot \#_w(F[\tau]).$$

It is easy to see that WMC is polynomial for the base classes CLU and FOREST as the corresponding algorithms for deciding satisfiability for these classes as discussed above allow a straightforward generalization to WMC. From Theorem 5 and Proposition 8 we conclude that WMC is fixed-parameter tractable parameterized by $\text{sb}_{\text{FOREST}}$ and db_{CLU}.

11.2 Quantified Boolean Formulas

Many important computational tasks like planning, verification, and several questions of knowledge representation and automated reasoning can be naturally encoded as the evaluation problem of *quantified Boolean formulas (QBF)* [74,84,87]. A QBF consists of a propositional CNF formula F (the "matrix") and a quantifier prefix. For instance $\mathcal{F} = \forall y \, \forall z \, \exists x \, \exists w \, F$ with $F = \{\{\neg x, y, \neg w\}, \{x, \neg y, w\}, \{\neg y, z\}, \{y, \neg z\}\}$ is a QBF. The evaluation of quantified Boolean formulas constitutes a PSPACE-complete problem and is therefore believed to be computationally harder than the NP-complete propositional satisfiability problem [58,76,94]. Only a few tractable classes of quantified Boolean formulas are known where the number of *quantifier alternations* is unbounded. For example, the time needed to solve QBF formulas whose primal graph has bounded treewidth grows non-elementarily in the number of quantifier alternations [75]. Two prominent tractable classes with *unbounded* quantifier alternations are QHorn and Q2CNF which are QBFs where the matrix is a Horn or 2CNF formula, respectively. QHorn formulas and Q2CNF formulas can be evaluated in polynomial time due to well-known results of Kleine Büning *et al.* [13] and of Aspvall *et al.* [2], respectively.

In order to evaluate a QBF formula with a small strong Horn- or 2CNF-backdoor set X efficiently, we require that X is closed under variable dependencies. That is, if x depends on y and $x \in X$, then also $y \in X$, where we say that x depends on y if the quantifier for y appears to the left of the quantifier for x, and one cannot move the quantifier for y to the right of x without changing the validity of the QBF. In general deciding whether a variable depends on the other is PSPACE complete, but there are "over-approximations" of dependencies that can be computed in polynomial time. Such over-approximations can be formalized in terms of *dependency schemes*. Indeed, it is fixed-parameter tractable to detect strong Horn or 2CNF-backdoor sets of size at most k that are closed with respect to any fixed polynomial-time decidable dependency scheme [89]. This fpt result allows an unbounded number of quantifier alternations for each value of the parameter, in contrast to the results for parameter treewidth.

11.3 Nonmonotonic Reasoning

Answer-Set Programming (ASP) is an increasingly popular framework for declarative programming [65,69]. ASP allows to describe a problem by means of rules and constraints that form a disjunctive logic program P over a finite universe U of atoms. A rule r is of the form $(x_1 \vee \cdots \vee x_l \leftarrow y_1, \ldots, y_n, \neg z_1, \ldots, \neg z_m)$. We write $\{x_1, \ldots, x_l\} = H(r)$ (the *head* of r) and $\{y_1, \ldots, y_n, z_1, \ldots, z_m\} = B(r)$ (the *body* of r), $B^+(r) = \{y_1, \ldots, y_n\}$ and $B^-(r) = \{z_1, \ldots, z_n\}$. A set M of atoms *satisfies* a rule r if $B^+(r) \subseteq M$ and $M \cap B^-(r) = \emptyset$ implies $M \cap H(r) \neq \emptyset$. M is a *model* of P if it satisfies all rules of P. The *GL reduct* of a program P under a set M of atoms is the program P^M obtained from P by first removing all rules r with $B^-(r) \cap M \neq \emptyset$ and second removing all $\neg z$ where $z \in B^-(r)$ from all remaining rules r [49]. M is an *answer set* of a program P if M it is a minimal model of P^M.

For instance, from the program $P = \{(\text{tweety-flies} \leftarrow \text{tweety-is-a-bird}, \neg\text{tweety-is-a-penguin}), (\text{tweety-is-a-bird} \leftarrow)\}$ we may conclude that tweety-flies, since this fact is contained in the only answer set $\{\text{tweety-is-a-bird}, \text{tweety-flies}\}$

of P. If we add the fact tweety-is-a-penguin to the program and obtain $P' = P \cup \{(\text{tweety-is-a-penguin} \leftarrow)\}$, then we have to retract our conclusion tweety-flies since this fact is not contained in any answer set of P' (the only answer set of P' is $\{\text{tweety-is-a-bird}, \text{tweety-is-a-penguin}\}$). This *nonmonotonic* behaviour that adding a fact may allow fewer conclusions is typical for many applications in Artificial Intelligence. The main computational problems for ASP (such as deciding whether a program has a solution, or if a certain atom is contained in at least one or in all answer sets) are of high worst-case complexity and are located at the second level of the Polynomial Hierarchy [38].

Also for ASP several islands of tractability are known, and it is possible to develop a backdoor approach [42]. Similar to SAT one can define partial truth assignments τ on a set of atoms and solve a disjunctive logic program P by solving all the reduced programs $P[\tau]$. However, the situation is trickier than for satisfiability. Although every answer set of P corresponds to an answer set of $P[\tau]$ for some truth assignment τ, the reverse direction is not true. Therefore, one needs to run a check for each answer set of $P[\tau]$ whether it gives rise to an answer set of P. Although this correctness check is polynomial, we must ensure that we do not need to carry it out too often. A sufficient condition for bounding the number of checks is that we can compute all answer sets of a program $P \in \mathcal{C}$ in polynomial time ("$\mathcal{C}$ is enumerable"). In particular, this means that $P \in \mathcal{C}$ has only a polynomial number of answer sets, and so we need to run the correctness check only a polynomial number of times.

Several enumerable islands of tractability have been identified and studied regarding the parameterized complexity of backdoor set detection [42]. For instance, programs where each rule head contains exactly one atom and each rule body is negation-free are well-known to have exactly one answer set. Such programs are called Horn programs, and similar to satisfiability, one can use vertex covers to compute backdoor sets with respect to Horn. Further enumerable islands of tractability can be defined by forbidding cycles in graphs, digraphs, and mixed graphs associated with disjunctive logic programs. Now, one can use feedback vertex set (fvs) algorithms for the considered graphs to compute backdoor sets: undirected fvs [33], directed fvs [20], and mixed fvs [12]. One can get even larger enumerable islands of tractability by labeling some of the vertices or edges and by only forbidding "bad" cycles, namely cycles that contain at least one labeled edge or vertex. For the undirected case one can use subset feedback vertex set algorithms to compute backdoor sets [28,57]. Currently it is open whether this problem is fixed-parameter tractable for directed or mixed graphs. Even larger islands can be obtained by only forbidding bad cycles with an even number of labeled vertices or edges [41]. This gives rise to further challenging feedback vertex set problems.

11.4 Abstract Argumentation

The study of arguments as abstract entities and their interaction in form of *attacks* as introduced by Dung [35] has become one of the most active research branches within Artificial Intelligence, Logic and Reasoning [5,7,80]. Abstract argumentation provides suitable concepts and formalisms to study, represent, and process various reasoning problems most prominently in defeasible reasoning (see, e.g., [79,11]) and agent interaction (see, e.g., [77]). An abstract argumentation system can be considered as a directed graph, where the vertices are called "arguments" and a directed edge from a to b means that argument a "attacks" argument b.

A main issue for any argumentation system is the selection of acceptable sets of arguments, called extensions. Whether or not a set of arguments is accepted is considered with respect to certain properties of sets of arguments, called semantics [4]. For instance, the *preferred semantics* requires that an extension is a maximal set of arguments with the properties that (i) the set is independent, and (ii) each argument outside the set which attacks some argument in the set is itself attacked by some argument in the set. Property (i) ensures that the set is conflict-free, property (ii) ensures that the set defends itself against attacks.

Important computational problems are to determine whether an argument belongs to some extension (credulous acceptance) or whether it belongs to all extensions (skeptical acceptance) [32,37]. For most semantics, including the preferred semantics, the problems are located on the second level of the Polynomial Hierarchy [36].

It is known that the acceptance problems can be solved in polynomial time if the directed graph of the argumentation framework is *acyclic*, *noeven* (contains no even cycles), *symmetric*, or *bipartite* [35,4,24,36]. Thus, these four properties give rise to islands of tractability for abstract argumentation, and one can ask whether a backdoor approach can be developed to solve the acceptance problems for instances that are close to an island. Here it is natural to consider deletion backdoor sets, i.e., we delete arguments to obtain an instance that belongs to the considered class. For the islands of acyclic, symmetric, and bipartite argumentation frameworks we can find a backdoor using the fixed-parameter algorithms for directed feedback vertex set [20], vertex cover [33] and for graph bipartization [83], respectively. For finding a vertex set of size k that kills all directed cycles of even length we only know an XP algorithm which is based on a deep result [85].

However, it turns out that using the backdoor set is tricky and quite different from satisfiability and answer set programming [73]. The acceptance problems remain (co-)NP-hard for instances that can be made symmetric or bipartite by deleting one single argument. On the other hand, if an instance can be made acyclic or noeven by deleting k arguments, then the acceptance problems can be solved in time $3^k n^c$. The base 3 of the running time comes from the fact that the evaluation algorithm considers three different cases for the arguments in the backdoor set: (1) the argument is in the acceptable set, (2) the argument is not in the set and is attacked by at least one argument from the set, and (3) the argument is not in the set but is not attacked by any argument from the set.

12 Conclusion

Backdoor sets aim at exploiting hidden structures in real-world problem instances. The effectiveness of this approach has been investigated empirically in [31,42,63,88] and in many cases, small backdoor sets were found for large industrial instances.

As several backdoor set problems reduce to well-investigated core problems from parameterized complexity, such as VERTEX COVER, 3-HITTING SET, FEEDBACK VERTEX SET, and their variants, a few decades of focused research efforts can be used to detect backdoor sets efficiently. Nevertheless, several questions remain open. In particular, the parameterized complexity classification of several permissive problems seems challenging. As discussed at the end of Subsection 11.3, the classification of variants of the FEEDBACK VERTEX SET problem would also shed some light on backdoor set detection problems in nonmonotonic reasoning.

We believe that more research in this direction is necessary if we want to explain the good practical performance of heuristic SAT solvers. Directions for future research could involve multivariate parameterizations of backdoor problems and the consideration of backdoors to combinations of different base classes.

Acknowledgment. We thank Ryan Williams for his comments on an earlier version of this survey.

References

1. Abu-Khzam, F.N.: A kernelization algorithm for d-hitting set. J. of Computer and System Sciences 76(7), 524–531 (2010)
2. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Information Processing Letters 8(3), 121–123 (1979)
3. Bacchus, F., Dalmao, S., Pitassi, T.: Algorithms and complexity results for #SAT and Bayesian inference. In: 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003), pp. 340–351 (2003)
4. Baroni, P., Giacomin, M.: Semantics of abstract argument systems. In: Rahwan, I., Simari, G. (eds.) Argumentation in Artificial Intelligence, pp. 25–44. Springer (2009)
5. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artificial Intelligence 171(10-15), 619–641 (2007)
6. Berre, D.L., Parrain, A.: On SAT technologies for dependency management and beyond. In: Thiel, S., Pohl, K. (eds.) Proceedings of 12th International Conference Software Product Lines Workshops, SPLC 2008, Limerick, Ireland, September 8-12, vol. 2, pp. 197–200. Lero Int. Science Centre, University of Limerick, Ireland (2008)
7. Besnard, P., Hunter, A.: Elements of Argumentation. The MIT Press (2008)
8. Bjesse, P., Leonard, T., Mokkedem, A.: Finding Bugs in an Alpha Microprocessor Using Satisfiability Solvers. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 454–464. Springer, Heidelberg (2001)
9. Bodlaender, H.L.: On disjoint cycles. International Journal of Foundations of Computer Science 5(1), 59–68 (1994)

10. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. of Computer and System Sciences 75(8), 423–434 (2009)
11. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93(1-2), 63–101 (1997)
12. Bonsma, P., Lokshtanov, D.: Feedback Vertex Set in Mixed Graphs. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 122–133. Springer, Heidelberg (2011)
13. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. Information and Computation 117(1), 12–18 (1995)
14. Cai, L., Huang, X.: Fixed-parameter approximation: Conceptual framework and approximability results. Algorithmica 57(2), 398–412 (2010)
15. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artificial Intelligence 172(6-7), 772–799 (2008)
16. Chen, J., Kanj, I.A.: On approximating minimum vertex cover for graphs with perfect matching. Theoretical Computer Science 337(1-3), 305–318 (2005)
17. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. J. of Computer and System Sciences 74(7), 1188–1198 (2008)
18. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: further observations and further improvements. J. Algorithms 41(2), 280–301 (2001)
19. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theoretical Computer Science 411(40–42), 3736–3756 (2010)
20. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. of the ACM 55(5), Art. 21, 19 (2008)
21. Chen, Y.-J., Grohe, M., Grüber, M.: On Parameterized Approximability. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 109–120. Springer, Heidelberg (2006)
22. Cook, S.A.: The complexity of theorem-proving procedures. In: Proc. 3rd Annual Symp. on Theory of Computing, pp. 151–158. Shaker Heights, Ohio (1971)
23. Cook, S.A., Mitchell, D.G.: Finding hard instances of the satisfiability problem: a survey. In: Satisfiability problem: theory and applications, Piscataway, NJ. American Mathematical Society, pp. 1–17 (1997)
24. Coste-Marquis, S., Devred, C., Marquis, P.: Symmetric Argumentation Frameworks. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 317–328. Springer, Heidelberg (2005)
25. Courcelle, B., Makowsky, J.A., Rotics, U.: On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. Discr. Appl. Math. 108(1-2), 23–52 (2001)
26. Courcelle, B.: Graph rewriting: an algebraic and logic approach. In: Handbook of Theoretical Computer Science, vol. B, pp. 193–242. Elsevier Science Publishers, North-Holland (1990)
27. Crama, Y., Ekin, O., Hammer, P.L.: Variable and term removal from Boolean formulae. Discr. Appl. Math. 75(3), 217–230 (1997)
28. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Subset Feedback Vertex Set Is Fixed-Parameter Tractable. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6755, pp. 449–461. Springer, Heidelberg (2011)
29. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. of the ACM 7(3), 201–215 (1960)

30. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM 5, 394–397 (1962)
31. Dilkina, B.N., Gomes, C.P., Sabharwal, A.: Tradeoffs in the Complexity of Backdoor Detection. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 256–270. Springer, Heidelberg (2007)
32. Dimopoulos, Y., Torres, A.: Graph theoretical structures in logic programs and default theories. Theoretical Computer Science 170(1-2), 209–244 (1996)
33. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
34. Downey, R.G., Fellows, M.R., McCartin, C.: Parameterized Approximation Problems. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 121–129. Springer, Heidelberg (2006)
35. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–357 (1995)
36. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. Artificial Intelligence 171(10-15), 701–729 (2007)
37. Dunne, P.E., Bench-Capon, T.J.M.: Coherence in finite argument systems. Artificial Intelligence 141(1-2), 187–203 (2002)
38. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. Ann. Math. Artif. Intell. 15(3-4), 289–323 (1995)
39. Fellows, M.R., Szeider, S., Wrightson, G.: On finding short resolution refutations and small unsatisfiable subsets. Theoretical Computer Science 351(3), 351–359 (2006)
40. Fernau, H.: A top-down approach to search-trees: Improved algorithmics for 3-hitting set. Algorithmica 57(1), 97–118 (2010)
41. Fichte, J.K.: The good, the bad, and the odd: Cycles in answer-set programs. In: ESSLII 2011 (2011)
42. Fichte, J.K., Szeider, S.: Backdoors to tractable answer-set programming. Technical Report 1104.2788, Arxiv.org (2012), Extended and updated version of a paper that appeared in the proceedings of IJCAI 2011. The 22nd International Joint Conference on Artificial Intelligence (2012)
43. Fischer, E., Makowsky, J.A., Ravve, E.R.: Counting truth assignments of formulas of bounded tree-width or clique-width. Discr. Appl. Math. 156(4), 511–529 (2008)
44. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, pp. 133–142. ACM (2008)
45. Garey, M.R., Johnson, D.R.: Computers and Intractability. W. H. Freeman and Company, New York (1979)
46. Gaspers, S., Szeider, S.: Backdoors to acyclic SAT. In: Proceedings of ICALP 2012 (Track A: Algorithms, Complexity and Games), the 39th International Colloquium on Automata, Languages and Programming, University of Warwick, UK, July 9-13. LNCS. Springer (to appear, 2012)
47. Gaspers, S., Szeider, S.: Strong backdoors to nested satisfiabiliy. In: Proceedings of SAT 2012, the 15th International Conference on Theory and Applications of Satisfiability Testing, Trento, Italy, June 17-20, 2012. LNCS. Springer (to appear, 2012)
48. Gaspers, S., Szeider, S.: Strong backdoors to bounded treewidth SAT. Technical report 1204.6233, Arxiv.org (2012)

49. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
50. Gomes, C.P., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: Handbook of Knowledge Representation. Foundations of Artificial Intelligence, vol. 3, pp. 89–134. Elsevier (2008)
51. Gottlob, G., Szeider, S.: Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. The Computer Journal 51(3), 303–325 (2006); survey paper
52. Guo, J., Hüffner, F., Niedermeier, R.: A Structural View on Parameterizing Problems: Distance from Triviality. In: Downey, R., Fellows, M., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 162–173. Springer, Heidelberg (2004)
53. Hertli, T.: 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. In: Ostrovsky, R. (ed.) Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011). IEEE (2011)
54. Iwama, K.: CNF-satisfiability test by counting and polynomial average time. SIAM J. Comput. 18(2), 385–391 (1989)
55. Kakimura, N., Kawarabayashi, K., Kobayashi, Y.: Erdös-Pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In: Rabani, Y. (ed.) Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, pp. 1726–1736. SIAM (2012)
56. Kautz, H.A., Selman, B.: Planning as satisfiability. In: Proceedings ECAI 1992, pp. 359–363 (1992)
57. Kawarabayashi, K., Kobayashi, Y.: Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. Technical report, University of Tokyo, Japan (2010); see also [55]
58. Büning, H.K., Lettman, T.: Propositional logic: deduction and algorithms. Cambridge University Press, Cambridge (1999)
59. Knuth, D.E.: Nested satisfiability. Acta Informatica 28(1), 1–6 (1990)
60. Kolaitis, P.G., Vardi, M.Y.: Conjunctive-query containment and constraint satisfaction. J. of Computer and System Sciences 61(2), 302–332 (2000); Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, WA (1998)
61. Levin, L.: Universal sequential search problems. Problems of Information Transmission 9(3), 265–266 (1973)
62. Lewis, H.R.: Renaming a set of clauses as a Horn set. J. of the ACM 25(1), 134–135 (1978)
63. Li, Z., van Beek, P.: Finding Small Backdoors in SAT Instances. In: Butz, C., Lingras, P. (eds.) Canadian AI 2011. LNCS, vol. 6657, pp. 269–280. Springer, Heidelberg (2011)
64. Gupta, A., Prasad, M., Biere, A.: A survey of recent advances in SAT-based formal verification. Software Tools for Technology Transfer 7(2), 156–173 (2005)
65. Marek, V.W., Truszczynski, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: a 25-Year Perspective, pp. 169–181. Springer (1999)
66. Marx, D., Schlotter, I.: Parameterized complexity and local search approaches for the stable marriage problem with ties. Algorithmica 58(1), 170–187 (2010)
67. Marx, D., Schlotter, I.: Stable assignment with couples: parameterized complexity and local search. Discrete Optim. 8(1), 25–40 (2011)

68. Mishra, S., Raman, V., Saurabh, S., Sikdar, S., Subramanian, C.R.: The Complexity of Finding Subgraphs Whose Matching Number Equals the Vertex Cover Number. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 268–279. Springer, Heidelberg (2007)
69. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. 25(3-4), 241–273 (1999); Logic programming with non-monotonic semantics: representing knowledge and its computation
70. Nishimura, N., Ragde, P., Szeider, S.: 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, Vancouver, BC, Canada, May 10-13, pp. 96–103 (2004)
71. Nishimura, N., Ragde, P., Szeider, S.: Solving #SAT using vertex covers. Acta Informatica 44(7-8), 509–523 (2007)
72. Ordyniak, S., Paulusma, D., Szeider, S.: Satisfiability of acyclic and almost acyclic CNF formulas. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, Chennai, India, December 15-18. LIPIcs, vol. 8, pp. 84–95. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
73. Ordyniak, S., Szeider, S.: Augmenting tractable fragments of abstract argumentation. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, pp. 1033–1038. AAAI Press (2011)
74. Otwell, C., Remshagen, A., Truemper, K.: An effective QBF solver for planning problems. In: Proceedings of MSV/AMCS, pp. 311–316. CSREA Press (2004)
75. Pan, G., Vardi, M.Y.: Fixed-parameter hierarchies inside PSPACE. In: Proceedings of 21th IEEE Symposium on Logic in Computer Science (LICS 2006), Seattle, WA, USA, August 12-15, pp. 27–36. IEEE Computer Society Press (2006)
76. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
77. Parsons, S., Wooldridge, M., Amgoud, L.: Properties and complexity of some formal inter-agent dialogues. J. Logic Comput. 13(3), 347–376 (2003)
78. Pietrzak, K.: 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)
79. Pollock, J.L.: How to reason defeasibly. Artificial Intelligence 57(1), 1–42 (1992)
80. Rahwan, I., Simari, G.R. (eds.): Argumentation in Artificial Intelligence. Springer (2009)
81. Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. ACM Transactions on Algorithms 2(3), 403–415 (2006)
82. Razgon, I., O'Sullivan, B.: Almost 2-SAT is fixed parameter tractable. J. of Computer and System Sciences 75(8), 435–450 (2009)
83. Reed, B., Smith, K., Vetta, A.: Finding odd cycle transversals. Oper. Res. Lett. 32(4), 299–301 (2004)
84. Rintanen, J.: Constructing conditional plans by a theorem-prover. J. Artif. Intell. Res. 10, 323–352 (1999)
85. Robertson, N., Seymour, P.D., Thomas, R.: Permanents, Pfaffian orientations, and even directed circuits. Ann. of Math (2) 150(3), 929–975 (1999)
86. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence 82(1-2), 273–302 (1996)

87. Sabharwal, A., Ansotegui, C., Gomes, C.P., Hart, J.W., Selman, B.: QBF Modeling: Exploiting Player Symmetry for Simplicity and Efficiency. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 382–395. Springer, Heidelberg (2006)
88. Samer, M., Szeider, S.: Backdoor trees. In: Twenty-Third Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, July 13–17, pp. 363–368. AAAI Press (2008)
89. Samer, M., Szeider, S.: Backdoor sets of quantified Boolean formulas. Journal of Automated Reasoning 42(1), 77–97 (2009)
90. Samer, M., Szeider, S.: Fixed-parameter tractability. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, vol. ch. 13, pp. 425–454. IOS Press (2009)
91. Samer, M., Szeider, S.: Algorithms for propositional model counting. J. Discrete Algorithms 8(1), 50–64 (2010)
92. Sang, T., Beame, P., Kautz, H.A.: Performing bayesian inference by weighted model counting. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, Pittsburgh, Pennsylvania, USA, July 9-13, pp. 475–482. AAAI Press / The MIT Press (2005)
93. Schaefer, T.J.: The complexity of satisfiability problems. In: Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, San Diego, Calif., pp. 216–226. ACM (1978)
94. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: Proc. Theory of Computing, pp. 1–9. ACM (1973)
95. Szeider, S.: On Fixed-Parameter Tractable Parameterizations of SAT. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 188–202. Springer, Heidelberg (2004)
96. Szeider, S.: Backdoor sets for DLL subsolvers. Journal of Automated Reasoning 35(1-3), 73–88 (2005); Reprinted as Giunchiglia, E., Walsh, T.(eds.): SAT 2005 - Satisfiability Research in the Year 2005, ch. 4. Springer Verlag (2006)
97. Szeider, S.: Matched formulas and backdoor sets. J. on Satisfiability, Boolean Modeling and Computation 6, 1–12 (2008)
98. Szeider, S.: Limits of preprocessing. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, AAAI 2011, pp. 93–98. AAAI Press, Menlo Park (2011)
99. Velev, M.N., Bryant, R.E.: Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. J. Symbolic Comput. 35(2), 73–106 (2003)
100. Weld, D.S.: Recent advances in AI planning. AI Magazine 20(2), 93–123 (1999)
101. Williams, R., Gomes, C., Selman, B.: Backdoors to typical case complexity. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003, pp. 1173–1178. Morgan Kaufmann (2003)