# Fixed-Parameter Algorithms for Artificial Intelligence, Constraint Satisfaction, and Database Problems

Georg Gottlob[1], Stefan Szeider[2]

[1] *Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, England, UK.* [2] *Durham University, Department of Computer Science, Science Labs, South Road, DH1 3LE Durham, England, UK.*

*Email: georg.gottlob@comlab.ox.ac.uk, stefan.szeider@durham.ac.uk*

**We survey the parameterized complexity of problems that arise in artificial intelligence, database theory, and automated reasoning. In particular, we consider various parameterizations of the constraint satisfaction problem, the evaluation problem of Boolean conjunctive database queries, and the propositional satisfiability problem. Furthermore, we survey parameterized algorithms for problems arising in the context of the stable model semantics of logic programs, for a number of other problems of non-monotonic reasoning, and for the computation of cores in data exchange.**

## 1. INTRODUCTION

Many problems that arise in artificial intelligence, database theory, and automated reasoning are at least NP-hard. In such cases, one often applies heuristics or approximation algorithms in order to obtain suboptimal solutions in a reasonable amount of time. The framework of *parameterized complexity*, initiated by Rod Downey and Mike Fellows, provides a means for developing optimal algorithms which are feasible in practice. The main idea is to associate with a problem instance a *parameter*, usually a non-negative integer $k$, which is small for instances of relevance, and to develop algorithms with running times that are possibly exponential in the parameter but uniformly polynomial in the instance size. In particular, an algorithm is a *fixed-parameter algorithm* if it solves instances of size $n$ and parameter $k$ in time $O(f(k)n^c)$; here $f$ denotes a computable function and $c$ denotes a constant which is independent of the parameter $k$. Although the function $f$ can be exponential (and *is* exponential for non-trivial cases), a fixed-parameter algorithm remains feasible for large instances provided that the parameter $k$ is small. This feature of fixed-parameter algorithms makes them often preferable to algorithms with a running time of the form $O(n^{f(k)})$, since the latter become impractical for large instances even when $f(k) = k$ and $k$ is small.

Example 1.1. A *vertex cover* of a graph $G$ is a set $S$ of vertices of $G$ such that every edge of $G$ has at least one of its ends in $S$. Given a graph $G$ and a non-negative integer $k$, the problem of deciding whether $G$ admits a vertex cover of size at most $k$, is a classical NP-complete problem [49]. However, several fixed-parameter algorithms are known that solve the following parameterized version of the problem:

> **VC**
> *Instance:* A graph $G$ and a non-negative integer $k$.
> *Parameter:* $k$.
> *Question:* Does $G$ admit a vertex cover of size at most $k$?

Indeed, **VC** is probably the best studied problem in parameterized complexity with a long history of improvements [18]. The currently fastest fixed-parameter algorithm for **VC** is due to Chen, Kanj, and Xia [19] and requires time $O(1.273^k + nk)$ and polynomial space for graphs of order $n$. □

In this paper we focus on *positive results*, i.e., the existence of fixed-parameter algorithms for the problems under consideration. The presented negative results (i.e., hardness results) have merely the purpose of carving out territories that are very likely to be inaccessible to fixed-parameter algorithms.

The majority of combinatorial problems studied in the framework of parameterized complexity offer a "natural parameter," e.g., it is natural to parameterize the vertex cover problem by the size of the vertex covers as in **VC** above. However, the problems considered in this survey lack of a natural parameter; there are numerous possibilities for parameters. An important

task is the identification of parameters that are as general as possible (i.e., for many instances of practical importance one can expect that the parameter is small), and which are still accessible to fixed-parameter algorithms. Thus, problems of that kind open up a race between more and more general parameters.

We have tried to cover many relevant results under a unified framework. However, there are certainly several interesting fixed-parameter algorithms known which are not covered in this survey. We apologize to all the authors whose results are not included.

The remainder of the paper is organized as follows. We close this section with a brief review of the basics of parameterized complexity. In Section 2 we consider various parameterizations of the constraint satisfaction problem and the evaluation problem for Boolean conjunctive database queries. In Section 3 we consider several parameterizations of the propositional satisfiability problem. Finally, in Section 4, we consider problems arising in the context of the stable model semantics of logic programs, problems related to other forms of non-monotonic reasoning, and we discuss complexity problems related to the computation of cores in data exchange.

### 1.1. Parameterized complexity in a nutshell

Next we give a brief and rather informal review of the most important concepts of parameterized complexity. For an in-depth treatment of the subject we refer the reader to other sources [29, 30, 35, 41, 45, 81].

An instance of a parameterized decision problem is a pair $(I, k)$ where $I$ is the *main part* and $k$ is the *parameter*; the latter consists usually of one or several non-negative integers. A parameterized decision problem is *fixed-parameter tractable* if it can be solved by a fixed-parameter algorithm, i.e., if instances $(I, k)$ can be decided in time $O(f(k)\|I\|^c)$ where $f$ is a computable function, $c$ is a constant, and $\|I\|$ represents the *size* of $I$ with respect to a reasonable encoding. FPT denotes the class of all fixed-parameter tractable decision problems.

The framework of parameterized complexity offers a *completeness theory*, similar to the theory of NP-completeness, that allows the accumulation of strong theoretical evidence that a parameterized problem is *not* fixed-parameter tractable. This completeness theory is based on the *weft hierarchy* of complexity classes $\mathrm{W}[1], \mathrm{W}[2], \ldots, \mathrm{W}[\mathrm{P}]$. Each class is the equivalence class of certain parameterized problems under *fpt-reductions*. Such a reduction is a straightforward extension of a polynomial-time many-one reduction that ensures a parameter for one problem maps into the parameter for another (see [30] for details). All reductions considered in this survey are of this type. The class XP consists of parameterized problems that can be solved in polynomial time if the parameter is considered as a constant. The above classes form the chain

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{P}] \subseteq \mathrm{XP}$$

where all inclusions are assumed to be proper; it is known that $\mathrm{FPT} \neq \mathrm{XP}$ [30, 45]. The following parameterized clique-problem is W[1]-complete; this problem is the basis for several hardness results considered below.

**CLIQUE**
*Instance:* A graph $G$ and a non-negative integer $k$.
*Parameter:* $k$.
*Question:* Does $G$ contain a clique on $k$ vertices?

A good reason for assuming that FPT and W[1] are distinct is provided by an analog of Cook's Theorem: FPT = W[1] implies the existence of a fixed-parameter algorithm for deciding whether a non-deterministic Turing machine accepts an input within $k$ non-deterministic computation steps [30]. Although XP contains problems which are very unlikely to be fixed-parameter tractable, it is often a significant improvement to show that a problem belongs to this class, in contrast to, e.g., $k$-SAT which is NP-hard for every constant $k \geq 3$.

## 2. CONSTRAINT SATISFACTION AND CONJUNCTIVE DATABASE QUERIES

### 2.1. Preliminaries

*Constraint Satisfaction* is a central problem of artificial intelligence and provides a powerful and general formalism for the encoding of various special-purpose problems. An instance of the constraint satisfaction problem (CSP) consist of a set of variables that range over a finite domain of possible values, together with a collection of constraints. Each constraint imposes a restriction on the combination of values for specified variables. The question is whether one can assign values to variables complying with all the restrictions imposed by the constraints.

Constraint satisfaction can be rephrased as the evaluation problem for *Boolean conjunctive queries* over relational databases. Actually, constraint satisfaction and Boolean conjunctive query evaluation are essentially the same problem [74]. Below we will state concepts and results in terms of constraint satisfaction; we will occasionally point out aspects that arise from the database theoretic point of view. Furthermore, the evaluation problem for Boolean conjunctive queries can be considered as the model checking problem for first-order formulas in prenex form whose only connectives are $\exists$ and $\wedge$. We note in passing that first-order model checking problems provide an alternative way of defining the complexity classes of the weft hierarchy; this approach was initiated by Downey,

Fellows, and Regan [31] and refined and extended by Flum and Grohe [44].

Formally, a *constraint satisfaction instance* is a triple $(V, D, \mathcal{C})$ where $V$ is a set of variables, $D$ is a set of values, the *domain*, and $\mathcal{C}$ is a set of *constraints*. Each constraint in $\mathcal{C}$ is a pair $(S, R)$ where $S$, the *constraint scope*, is a sequence of distinct variables of $V$, and $R$, the *constraint relation*, is a relation over $D$ whose arity matches the length of $S$ (that is, $R \subseteq D^r$). A constraint satisfaction instance with domain $\{0, 1\}$ is called *Boolean*. We denote by $\mathrm{var}(C)$ the set of variables that occur in the scope of a constraint $C$. We define the *size* of an instance $I = (V, D, \mathcal{C})$ as the length of a string that encodes the instance; more specifically, the size $\|C\|$ of a constraint $C = (S, R)$ is given by $|\mathrm{var}(C)| \cdot (|R| + 1)$; the size of the instance $I$ is given by $|V| + |D| + \sum_{C \in \mathcal{C}} \|C\|$.

Consider a constraint satisfaction instance $I = (V, D, \mathcal{C})$. An assignment $\alpha : V \to D$ *satisfies* a constraint $((x_1, \ldots, x_r), R) \in \mathcal{C}$ if $(\alpha(x_1), \ldots, \alpha(x_r)) \in R$. An assignment satisfies $I$ if it satisfies all the constraints of $I$ simultaneously; in that case we also say that $\alpha$ is a *solution* of $I$. A constraint satisfaction instance is *satisfiable* or *consistent* if it has at least one solution. **CSP** is the problem of deciding whether a given instance is consistent.

Example 2.1. Graph 3-colorability can be expressed as a constraint satisfaction problem: for a given graph $G = (V, E)$ we form the constraint satisfaction instance $I = (V, \{\text{red, green, blue}\}, \mathcal{C})$ where for every edge $uv \in E$ the set $\mathcal{C}$ contains the constraint $((u, v), R)$ with $R = \{$(red,green), (red,blue), (green,red), (green,blue), (blue,red), (blue,green)$\}$. □

This example shows that the constraint satisfaction problem is NP-hard.

Example 2.2. Similarly to Example 2.1, the popular *Sudoku* puzzle can be described as a constraint satisfaction problem [98]. As variables we take the 81 squares of the game, the domain is the set of integers from 1 to 9. Each clue number gives raise to a unary constraint that fixes the value for a variable. For each set of nine variables $x_1, \ldots, x_9$ that form a row, column, or a $3 \times 3$ subgrid we take a constraint $((x_1, \ldots, x_9), R)$ where $R$ contains all tuples that correspond to a permutation of the integers $1, \ldots, 9$. □

The next two examples show how the clique problem can be encoded as as a constraint satisfaction problem. These two examples are the base for several hardness results considered below.

Example 2.3 ([87]). Let $G = (V, E)$ be a graph and let $k$ be a positive integer. We consider the constraint satisfaction instance $I = (\{x_1, \ldots, x_k\}, V, \{\, ((x_i, x_j), R) : 1 \leq i < j \leq k \,\})$ where $R = \{\, (u, v) : uv \in E \,\}$. It is easy to verify that $G$ contains a clique on $k$ vertices if and only if $I$ is consistent. □

Example 2.4 ([93]). Let $G = (\{v_1, \ldots, v_n\}, E)$ be a graph and let $k$ be a positive integer. Each edge $e = v_i v_{i'}$ of $G$ can be encoded by a tuple $t_e = (d_1, \ldots, d_n, d'_1, \ldots, d'_n)$ where $d_i = d'_{i'} = 1$ and all other values are 0. We consider the *Boolean* constraint satisfaction instance $I$ with $kn$ variables $x_i^j$ $(1 \leq i \leq n,\ 1 \leq j \leq k)$ and $\binom{k}{2}$ many constraints $C^{j,j'}$ $(1 \leq j < j' \leq k)$. The constraints are defined by $C^{j,j'} = ((x_1^j, \ldots, x_n^j, x_1^{j'}, \ldots, x_n^{j'}), R)$, where $R$ contains all tuples $t_{v_i v_{i'}}$ for $v_i v_{i'} \in E$ and $i < i'$. Again, it is easy to verify that $G$ contains a clique on $k$ vertices if and only if $I$ is consistent. □

Example 2.5. Consider a relational database with the following relational schemes.

```
works(Emp#,Proj#)
manages(Emp#,Proj#)
relative(Emp1,Emp2)
```

The following Boolean conjunctive query checks whether an employee works in a project managed by his or her relative.

```
ans ⟵ works(E,P) ∧ manages(M,P)
       ∧ relative(E,M).
```

Formulating this query as a constraint satisfaction instance we get $I = (\{x_E, x_M, x_P\}, D, \{((x_E, x_P), R_w), ((x_M, x_P), R_m), ((x_E, x_M), R_r)\})$, where $D$ is the set of all persons contained in the database, and $R_w, R_m, R_r$ are the binary relations that correspond to the database records. □

### 2.2. Parameterized constraint satisfaction

A *constraint satisfaction parameter* is a computable function $p$ that assigns to every constraint satisfaction instance $I$ a non-negative integer $p(I)$. For constraint satisfaction parameters $p_1, \ldots, p_r$ we consider the following generic parameterized problem:

> **CSP**$(p_1, \ldots, p_r)$
> *Instance:* A constraint satisfaction instance $I$ and non-negative integers $k_1, \ldots, k_r$ with $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$.
> *Parameters:* $k_1, \ldots, k_r$.
> *Question:* Is $I$ consistent?

Note that we formulate this problem as a "promise problem" in the sense that for solving the problem we do not need to verify the assumption $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$. However, for most of the cases considered below the verification of the assumption $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$ is fixed-parameter tractable. For a constraint satisfaction instance $I = (V, D, \mathcal{C})$ we have the basic parameters $\mathbf{vars}(I) = |V|$, the number of variables of $I$, $\mathbf{length}(I) = \sum_{C \in \mathcal{C}} |\mathrm{var}(C)|$, the sum

of arities of constraints, and $\mathbf{dom}(I) = |D|$, the domain size.

In the context of Boolean conjunctive queries, the parameter **length** is known as the *query size*. For queries it is natural to assume a small query size in contrast to a large size of the database.

If we parameterize by the domain size, then the problem remains NP-hard, since, e.g., 3-colorability can be expressed as a constraint satisfaction problem with constant domain (see Example 2.1 above). Thus $\mathbf{CSP}(\mathbf{dom})$ is not fixed-parameter tractable unless P = NP. On the other hand, if we parameterize by the number of variables *and* the domain size, i.e., $\mathbf{CSP}(\mathbf{vars}, \mathbf{dom})$, then we have a trivially fixed-parameter tractable problem: we can decide the consistency of an instance $I$ by checking all $\mathbf{dom}(I)^{\mathbf{vars}(I)}$ possible assignments.

Somewhat surprisingly, if we do not bound the domain size, we get a W[1]-complete problem.

THEOREM 2.1 ([87]). *The problems* $\mathbf{CSP}(\mathbf{vars})$ *and* $\mathbf{CSP}(\mathbf{length})$ *are* W[1]*-complete.*

The hardness part of this theorem follows from Example 2.3 which gives a reduction from **CLIQUE**.

Let $\mathcal{A}$ be a deterministic polynomial-time algorithm that applies simplification and propagation rules to a constraint satisfaction instance $I$ without changing consistency. For example, arc consistency is an important simplification and propagation rule used in constraint solvers. Let $\mathcal{A}(I)$ denote the instance obtained form $I$ by applying algorithm $\mathcal{A}$, and let $\mathbf{kern}_{\mathcal{A}}(I)$ denote the size of $\mathcal{A}(I)$ (if the size of $\mathcal{A}(I)$ depends on a particular ordering of variables and constraints, let $\mathbf{kern}_{\mathcal{A}}(I)$ denote the largest size of $\mathcal{A}(I)$ over all such orderings).

It follows now that the problem $\mathbf{CSP}(\mathbf{kern}_{\mathcal{A}})$ is fixed-parameter tractable, since after the polynomial-time preprocessing we are left with an instance $\mathcal{A}(I)$ whose size is bounded in terms of the parameter, and therefore any brute-force algorithm applied to $\mathcal{A}(I)$ yields fixed-parameter tractability. In other words, $\mathbf{CSP}(\mathbf{kern}_{\mathcal{A}})$ is fixed-parameter tractable by definition.

Next we will consider nontrivial constraint satisfaction parameters that give rise to tractable classes of constraint satisfaction instances. There are two main approaches for obtaining tractable classes. The first possibility for achieving tractability is to restrict the constraint relations to a certain class of relations, a *constraint language*. For example, Schaefer's Dichotomy Theorem [96] is a classical result which classifies Boolean constraint languages. For further results on tractable constraint languages, see, e.g., Cohen and Jeavon's survey [21].

The second possibility is to impose *structural restrictions* on the way how variables and constraints interact. To this end one associates with a constraint satisfaction instance certain (hyper)graphs that model the overall structure of the instance.

As observed by Feder and Vardi [39], the constraint satisfaction problem can be rephrased as the question of whether there exists a homomorphism from a relational structure $\mathbf{A}$ into a relational structure $\mathbf{B}$. This point of view reveals an interesting symmetry between the above two possibilities for achieving tractability: structural restrictions correspond to restrictions on the structure $\mathbf{A}$, whereas restrictions on the constraint relations correspond to restrictions on the structure $\mathbf{B}$.

In this survey we will focus mainly on parameters that impose structural restrictions. To this end we associate the following (hyper)graphs with a constraint satisfaction instance $I$.

The *primal graph* $G_I$ is the graph whose vertices are the variables of $I$; two variables are joined by an edge if and only if the variables occur together in the scope of a constraint of $I$. The *incidence graph* $G_I^*$ is the bipartite graph whose vertices are the variables and the constraints of $I$; a variable $x$ and a constraint $C$ are joined by an edge if and only if $x \in \mathrm{var}(C)$. Finally, the *constraint hypergraph* $H_I$ is the hypergraph whose vertices are, as for primal graphs, the variables of $I$; the hyperedges of $H_I$ are the sets $\mathrm{var}(C)$ for constraints $C$ of $I$.

### 2.3. Instances of bounded treewidth

The graph parameter *treewidth* measures in a certain sense the "tree-likeness" of a graph. Treewidth is famous for its central role in Robertson and Seymour's Graph Minor Project; however, the concept was independently discovered by other authors. Many otherwise NP-hard graph problems such as Hamiltonicity and 3-colorability are fixed-parameter tractable if parameterized by the treewidth of the input graph. It is generally believed that many practically relevant problems actually do have low treewidth [11]. Treewidth has also been fruitfully applied to several areas of AI and automated reasoning [59]. Before we discuss applications of treewidth to constraint satisfaction we briefly recall its definition.

Let $G$ be a graph and let $T$ be a tree. To improve readability we will refer to the vertices of $T$ as *nodes*. Let $\chi$ be a labeling of the nodes of $T$ by sets vertices of $G$; the sets $\chi(t)$ are called *bags*. The pair $(T, \chi)$ is a *tree decomposition* of $G$ if the following conditions hold:

(i) every vertex of $G$ belongs to $\chi(t)$ for some node $t$ of $T$;
(ii) for every edge $vw$ of $G$ there is some node $t$ of $T$ such that $v, w \in \chi(t)$;
(iii) for any three nodes $t_1, t_2, t_3$ of $T$, if $t_2$ lies on the path from $t_1$ to $t_3$, then $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$.

The *width* of a tree decomposition $(T, \chi)$ is the maximum $|\chi(t)| - 1$ over all nodes $t$ of $T$; the *treewidth* $\mathrm{tw}(G)$ of $G$ is the minimum width over all its tree decompositions. Determining the treewidth of a graph is an NP-hard problem, as shown by

Arnborg, Corneil, and Proskurowski [5]. However, as shown by Bodlaender [15], if $k$ is fixed, then for a given graph $G$ one can find in linear time a tree decomposition of width at most $k$, or decide that the treewidth of $G$ exceeds $k$. Bodlaender's algorithm is not feasible in practice as its linear running time involves a huge constant factor. For practical applications other algorithms are preferred [14, 16].

The importance of treewidth lies in the fact that many hard graph problems can be solved in polynomial time for graphs of bounded treewidth [12, 23]. So it is obvious to consider constraint satisfaction instances with primal or incidence graphs of bounded treewidth. Indeed, it was observed by Freuder [48] that instances whose primal graphs have bounded treewidth can be solved in (non-uniformal) polynomial time.

We define the constraint satisfaction parameters $\mathbf{tw}(I) = \mathrm{tw}(G_I)$ and $\mathbf{tw}^*(I) = \mathrm{tw}(G_I^*)$ where $G_I$ and $G_I^*$ are the primal and incidence graphs of $I$, respectively. Since $\mathbf{tw}(I) \leq \mathbf{vars}(I)$ holds by trivial reasons for every constraint satisfaction instance $I$, the W[1]-hardness of $\mathbf{CSP}(\mathbf{tw})$ is a direct consequence of Theorem 2.1.

COROLLARY 2.1. *The problem* $\mathbf{CSP}(\mathbf{tw})$ *is* W[1]*-hard.*

However, as observed by Gottlob, Scarcello, and Sideri [60], additionally bounding the domain size renders the problem fixed-parameter tractable:

THEOREM 2.2 ([60]). *The problem* $\mathbf{CSP}(\mathbf{tw}, \mathbf{dom})$ *is fixed-parameter tractable.*

Fixed-parameter tractability of $\mathbf{CSP}(\mathbf{tw}, \mathbf{dom})$ can be established by means of dynamic programming along tree decompositions. Given an instance $I = (V, D, \mathcal{C})$ we first obtain a tree decomposition $((V_T, E_T), \chi)$ of width $k$ of the primal graph $G_I$. Next we produce an instance $I' = (V, D, \mathcal{C}')$ that is solution-equivalent to $I$, taking for every node $t$ of $T$ a new constraint $C_t = (\chi(t), R_t)$ whose relation $R_t$ contains those tuples over $D$ that are consistent with all constraints $C$ of $I$ with $\mathrm{var}(C) \subseteq \chi(t)$. Since the domain size is bounded, the size of $C_t$ is bounded as well. Now $I'$ is an "acyclic" instance that can be solved by a bottom-up traversal of $T$. This procedure for acyclic instances was first described by Yannakakis [109].

Bounding the treewidth of incidence graphs instead of the treewidth of primal graphs yields a more general constraint satisfaction parameter $\mathbf{tw}^*$. Namely, there are classes of constraint satisfaction instances with bounded $\mathbf{tw}^*$ and arbitrarily large $\mathbf{tw}$. However, the treewidth of the incidence graph $G_I^*$ exceeds the treewidth of the primal graph $G_I$ at most by one [74]. Hence we have the following corollary to Theorem 2.1.

COROLLARY 2.2. *The problem* $\mathbf{CSP}(\mathbf{tw}^*)$ *is* W[1]*-hard.*

However, unlike for $\mathbf{CSP}(\mathbf{tw})$, bounding the domain size by an additional parameter does not yield fixed-parameter tractability. This follows from the reduction from **CLIQUE** as described in Example 2.4. In fact, for Boolean instances $I$ that encode the existence of a clique of size $k$ in a graph, as described in Example 2.4, we have $\mathbf{tw}^*(I) \leq \binom{k}{2}$. This can be seen by taking a tree decomposition $(T, \chi)$ of the incidence graph where $T$ is a path (i.e., $(T, \chi)$ is a path decomposition) where bags contain all the $\binom{k}{2}$ constraints plus one of the variables. Hence we have a reduction from **CLIQUE** to $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom})$.

THEOREM 2.3 ([93]). *The problem* $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom})$ *is* W[1]*-hard.*

This result implies W[1]-hardness of $\mathbf{CSP}(p, \mathbf{dom})$ for every constraint satisfaction parameter $p$ that is more general than $\mathbf{tw}^*$, i.e., if $p(I) \leq f(\mathbf{tw}^*(I))$ for some computable function $f$.

Next we present a decomposition method which is based on feedback vertex sets and was proposed by Dechter [27]. This method is less general than the method of bounded treewidth. However, it is often preferred since it is significantly simpler to implement and requires less space. This method provides an example for the "backdoor set approach" to constraint solving.

A set $S$ of vertices of a graph $G$ is a *cycle cut set* or *feedback vertex set* of $G$ if $G - S$ is acyclic (here $G - S$ denotes the graph obtained from $G$ by removing all vertices in $S$ and all the edges that are incident with vertices in $S$). Although it is NP-hard to determine, given $G$ and $k$, whether $G$ has a feedback vertex set of size at most $k$ [49], the following parameterized problem is fixed-parameter tractable.

**FVS**

*Instance:* A graph $G$, a non-negative integer $k$.

*Parameter:* $k$.

*Question:* Does $G$ have a feedback vertex set of size at most $k$?

Downey and Fellows [30] have proposed two fixed-parameter algorithms for **FVS**, one is based on bounded search trees, one is based on a structure theorem due to Bodlaender [13]. An efficient fixed-parameter algorithm for **FVS** based on the iterative compression technique is due to Guo et al. [64].

Assume that $S$ is a feedback vertex set of the primal graph of a constraint satisfaction instance $I = (V, D, \mathcal{C})$. For each partial assignment $\alpha : S \to D$ we obtain from $I$ the instance $I[\alpha] = (V \setminus S, D, \mathcal{C}')$, the *restriction of* $I$ *to* $\alpha$, as follows: First we remove all constraints from $I$ that are satisfied by $\alpha$. Next we remove from the remaining constraints all tuples that disagree in at least one variable with $\alpha$, and we eliminate the variables of $S$ from the scopes and update the relations accordingly.

It is not difficult to see that $I$ is consistent if and only if $I[\alpha]$ is consistent for at least one partial assignment $\alpha : S \to D$. However, since $S$ is a feedback vertex set of the primal graph of $I$, the primal graphs of all the instances $I[\alpha]$ are acyclic. Whence each of the $|D|^{|S|}$ instances $I[\alpha]$ can be solved in polynomial time. This renders the problem fixed-parameter tractable if we take $|D|$ and $|S|$ as parameters.

Therefore, denoting by $\mathbf{fvs}(I)$ the size of a smallest feedback vertex set of the primal graph of $I$, we have the following result.

THEOREM 2.4. *The problem* $\mathbf{CSP}(\mathbf{fvs}, \mathbf{dom})$ *is fixed-parameter tractable.*

As noted above, this result is less general than Theorem 2.2 since graphs with small feedback vertex sets have small treewidth, but the converse is not the case.

Partial assignments and restrictions as considered above provide a general framework for parameterizing the constraint satisfaction problem. Let $\Gamma$ be a class of constraint satisfaction instances for which consistency can be decided in polynomial time. Consider an arbitrary constraint satisfaction instance $I = (V, D, \mathcal{C})$ and a subset $B \subseteq V$. The set $B$ is called a *strong* $\Gamma$-*backdoor set* of $I$ if for all partial assignments $\alpha : B \to D$, the restriction $I[\alpha]$ belongs to the base class $\Gamma$; this notion of backdoor sets was introduced by Williams, Gomes, and Selman [106]. Let $\mathbf{b}_\Gamma(I)$ denote the size of a smallest strong $\Gamma$-backdoor set of $I$. If we know a strong $\Gamma$-backdoor set $B$ of an instance $I = (V, D, \mathcal{C})$, then we can solve $I$ by considering all $|D|^{|B|}$ restrictions $I[\alpha]$ for $\alpha : B \to D$. Hence, if the detection of a strong $\Gamma$-backdoor set of size at most $k$ is fixed parameter tractable with respect to the parameter $k$, then $\mathbf{CSP}(\mathbf{b}_\Gamma, \mathbf{dom})$ is fixed-parameter tractable. Note that, if $\Gamma_0$ denotes the class of constraint satisfaction instances with acyclic primal graphs, then feedback vertex sets of primal graphs, as considered above, are strong $\Gamma_0$-backdoor sets.

The identification of base classes $\Gamma$ with fixed-parameter tractable backdoor set detection is an interesting research objective. In the next section we will consider the concept of backdoor sets with respect to propositional satisfiability.

Grohe [61] has established an interesting result under the complexity theoretic assumption FPT $\neq$ W[1]; see also the conference paper of Grohe, Schwentick, and Segoufin [63]: if a class of polynomial time solvable constraint satisfaction instances is solely characterized in terms of primal graphs, then the primal graphs of the instances in the class have bounded treewidth. This result rests on Robertson and Seymour's deep Excluded Grid Theorem [90]. For stating Grohe's theorem more specifically, we introduce the following notation. For a class $\mathcal{G}$ of graphs let $\mathbf{CSP}[\mathcal{G}]$ denote the constraint satisfaction problem restricted to instances whose primal graphs belongs to $\mathcal{G}$. We say that $\mathcal{G}$ has *bounded treewidth* if there exists a constant $k$ such that all graphs in $\mathcal{G}$ have treewidth at most $k$.

THEOREM 2.5 ([61]). *Unless* FPT = W[1]*, the following statements are equivalent for every recursively enumerable class* $\mathcal{G}$ *of graphs:*

(i) *The problem* $\mathbf{CSP}(\mathbf{length})[\mathcal{G}]$ *is fixed-parameter tractable.*
(ii) *The problem* $\mathbf{CSP}[\mathcal{G}]$ *can be solved in polynomial time.*
(iii) $\mathcal{G}$ *has bounded treewidth.*

### 2.4. Beyond treewidth

Constraints of large arity can be helpful, as illustrated by the following example.

EXAMPLE 2.6. Consider constraint satisfaction instance $I = (V, D, \mathcal{C})$ that contains a constraint $C_{\text{big}}$ with $\text{var}(C_{\text{big}}) = V$. We can solve the instance in polynomial time, since we only need to check whether one of the tuples of the constraint relation of $C_{\text{big}}$ defines a solution. □

A constraint whose scope contains $n$ variables produces a clique on $n$ vertices in the primal graph. The primal graph, however, does not provide enough information to conclude from the presence of a clique on $n$ vertices that the corresponding $n$ variables occur together in the scope of a constraint (the clique could be the result of, say, several binary constraints). Therefore it is beneficial to consider the *constraint hypergraph*, as defined above, which contains this additional structural information.

Let us fix some hypergraph terminology before we discuss properties of constraint hypergraphs. A *hypergraph* $H$ is a pair $(V, E)$ where $V$ is a set of vertices and $E$ is a set of hyperedges; a hyperedge is a subset of $V$. The *primal graph* or *2-section* of a hypergraph $H = (V, E)$ is the graph $G = (V, E_2)$ where $E_2$ contains all pairs of distinct vertices that occur together in some hyperedge of $H$. Thus the primal graph of a constraint satisfaction instance $I$ is the primal graph of the constraint hypergraph of $I$.

A constraint satisfaction instance is *acyclic* if its constraint hypergraph is acyclic, where a hypergraph is called acyclic if it can be reduced to the empty hypergraph $(\emptyset, \emptyset)$ by multiple applications of the following rules (we refer here to the most general concept of hypergraph acyclicity, also known as $\alpha$-acyclicity [36]).

(i) Remove a vertex that belongs to at most one hyperedge.
(ii) Remove a hyperedge that is empty or is a subset of another hyperedge.

Several equivalent definitions of hypergraph acyclicity are known [8]. For example, a hypergraph is acyclic if and only if there exists a tree decomposition of

its primal graph such that the bags of the tree decomposition are hyperedges of the hypergraph. Such a tree decomposition, rooted at one of its nodes, is also known as a *join tree* of the acyclic hypergraph. The following is a well-known result of Yannakakis [109].

THEOREM 2.6 ([109]). *Acyclic constraint satisfaction instances can be solved in polynomial time.*

Actually, in our above sketch of the algorithm that establishes Theorem 2.2 we have already encountered a join tree: the tree decomposition of the given instance $I$ is nothing but a join tree of the constraint hypergraph of the equivalent instance $I'$.

Next we state a related fixed-parameter tractability result that is based on a mixture of restrictions on both the structure and the constraint relations. Often constraint satisfaction instances involve several *non-equality* constraints, i.e., constraints that impose the restriction that two variables cannot receive the same value. For a constraint satisfaction instance $I$ let $I_{\neq}$ denote the instance obtained from $I$ by removing all non-equality constraints. Furthermore, let $\mathbf{CSP}_{\neq}^{\text{acyc}}$ denote the constraint satisfaction problem restricted to instances $I$ for which $I_{\neq}$ is acyclic. The question arises of whether $\mathbf{CSP}_{\neq}^{\text{acyc}}$ is easier than **CSP**. Interestingly, as shown by Papadimitriou and Yannakakis [87], the answer is *no* for the non-parameterized setting, but *yes* for the parameterized setting:

THEOREM 2.7 ([87]). *The problem* $\mathbf{CSP}_{\neq}^{\text{acyc}}$ *is* NP*-complete.*

THEOREM 2.8 ([87]). *The problem* $\mathbf{CSP}_{\neq}^{\text{acyc}}(\mathbf{vars})$ *is fixed-parameter tractable.*

In a series of papers, Gottlob, Leone, and Scarcello have developed a concept of hypergraph decompositions which allows a slice-wise generalization of acyclic constraint satisfaction instances [56]. Let $H = (V, E)$ be a hypergraph. A *generalized hypertree decomposition* of $H$ is a triple $(T, \chi, \lambda)$ where $(T, \chi)$ is a tree decomposition of the primal graph of $H$ and $\lambda$ is a mapping that assigns to every node $t$ of $T$ a set $\lambda(t)$ of hyperedges of $H$ such that each vertex $x \in \chi(t)$ is contained in some hyperedge $e \in \lambda(t)$. Thus $\lambda(t)$ is a *set cover* of $\chi(t)$. The *width* of a generalized hypertree decomposition $(T, \chi, \lambda)$ is the size of a largest set $\lambda(t)$ over all nodes $t$ of $T$. The *generalized hypertree width* $\text{ghw}(H)$ of $H$ is the minimum width over all generalized hypertree decompositions of $H$. The *hypertree width* $\text{hw}(H)$ is defined similarly, considering hypertree decompositions that satisfy an additional technical condition. Recently Adler, Gottlob, and Grohe [1] have shown that hypertree width and generalized hypertree width are related by a small multiplicative constant, namely $\text{ghw}(H) \leq \text{hw}(H) \leq 3 \cdot \text{ghw}(H) + 1$. For fixed $k$ one can decide in polynomial time whether $\text{hw}(G) \leq k$ (see below); for generalized hypertree width, however, deciding $\text{ghw}(H) \leq 3$ is NP-hard [57].

We define the constraint satisfaction parameters **ghw** and **hw** via the (generalized) hypertree width of constraint hypergraphs. Gottlob, Leone, and Scarcello [55] have shown that for any constant $k$ one can decide constraint satisfaction instances $I$ with $\mathbf{hw}(I) \leq k$ in polynomial time; the problem is even in the complexity class LOGCFL and so highly parallelizable.

THEOREM 2.9 ([55]). *The problems* $\mathbf{CSP}(\mathbf{hw})$ *and* $\mathbf{CSP}(\mathbf{ghw})$ *are in* XP.

The polynomial time algorithm of Gottlob et al. [55] can be outlined as follows. Let $I$ be an instance of size $s$ with $\mathbf{ghw}(I) \leq k$ and let $H_I = (V, E)$ be its constraint hypergraph; by the above result of Adler et al., $\mathbf{hw}(I) \leq 3k + 1 =: k'$. First we compute a hypertree decomposition of $H$ of width at most $k'$. This can be accomplished in time $O(|E|^{2k'}|V|^2)$. Second, using the decomposition, we transform $I$ into a solution-equivalent acyclic instance $I'$. This can be accomplished in time $O(s^{k'})$. Third, we solve the acyclic instance $I'$ by Yannakakis' polynomial-time algorithm as outlined above. In total we have the overall time complexity $O(s^{2(k'+1)} \log s)$.

This algorithm does not render the problem $\mathbf{CSP}(\mathbf{ghw})$ fixed-parameter tractable. A recent result of Gottlob, Grohe, Musliu, Samer, and Scarcello [54] shows that the first step of the above procedure involves a W[2]-hard problem:

THEOREM 2.10 ([54]). *The problem of deciding whether* $\text{ghw}(H) \leq k$ *(or,* $\text{hw}(H) \leq k$*) holds for a given hypergraph* $H$ *is* W[2]*-hard with respect to the parameter* $k$*.*

Chen and Dalmau [17] have shown that it is possible to avoid the decomposition step. They present a game-theoretic algorithm that decides consistency in polynomial time for instances whose generalized hypertree width is bounded by a constant. Their algorithm, however, is not a fixed-parameter algorithm either. Furthermore, a similar argument as used for the proof of Theorem 2.3 shows that the problems $\mathbf{CSP}(\mathbf{hw}, \mathbf{dom})$ and $\mathbf{CSP}(\mathbf{ghw}, \mathbf{dom})$ are W[1]-hard.

Grohe and Marx [62] have proposed two new hypergraph parameters that give raise to new classes of polynomial-time solvable constraint satisfaction instances. A *fractional edge cover* of a hypergraph $H = (V, E)$ is a mapping $\psi : E \to [0, \infty]$ such that $\sum_{e \in E, v \in e} \psi(e) \geq 1$ holds for all vertices $v \in V$. The *weight* of $\psi$ is given by $\sum_{e \in E} \psi(e)$. The *fractional edge cover number* $\rho^*(H)$ is the minimum of the weights of all edge covers of $H$. For a constraint satisfaction instance $I$ let $\rho^*(I) = \rho(H_I)$. Using Shearer's Lemma [20], Grohe and Marx have shown that, given a constraint satisfaction instance $I$ of size $s$ and $k = \rho^*(I)$, all solutions of $I$ can be listed in time $s^{k+O(1)}$. This yields the following result.

THEOREM 2.11 ([62]). *The problem* $\mathbf{CSP}(\rho^*)$ *is in* XP.

This result is interesting since there are classes of constraint satisfaction instances of constant $\rho^*$ with arbitrary high **ghw**, and classes of instances with arbitrarily high $\rho^*$ and constant **ghw** [62]. Furthermore, Grohe and Marx have defined *fractional hypertree decompositions* of hypergraphs, combining the concepts of generalized hypertree decompositions and fractional edge covers. This new decomposition concept gives rise to the hypergraph parameter *fractional hypertree width* $\mathbf{fhw}(H)$. For every hypergraph $H$, $\mathbf{fhw}(H)$ is bounded by both $\rho^*(H)$ and $\mathbf{ghw}(H)$; hence fractional hypertree width generalizes both other parameters. It is not known whether for a given hypergraph $H$ with $\mathbf{fhw}(H) \leq k$ for a fixed constant $k$, one can find a fractional hypertree decomposition of $H$ of width at most $k$ in polynomial time. However, if such a decomposition is provided, then the instance can be solved in polynomial time. For a class $\mathcal{H}$ of hypergraphs let $\mathbf{CSP}[\mathcal{H}]$ denote the constraint satisfaction problem restricted to instances whose constraint hypergraphs belong to $\mathcal{H}$.

THEOREM 2.12 ([62]). *The problem* $\mathbf{CSP}(\mathbf{vars})[\mathcal{H}]$ *is fixed-parameter tractable for every class* $\mathcal{H}$ *of hypergraphs with bounded fractional hypertree width.*

Since **fhw** is more general than $\mathbf{tw}^*$, Theorem 2.3 implies that the problem $\mathbf{CSP}(\mathbf{fhw}, \mathbf{dom})$ is W[1]-hard.

## 3. PROPOSITIONAL SATISFIABILITY

### 3.1. Preliminaries

Propositional satisfiability is the classical problem of determining whether a propositional formula in conjunctive normal form has a satisfying truth assignment. Cook's famous proof that this problem is NP-complete placed satisfiability as the cornerstone of complexity theory. Despite its seemingly specialized nature, satisfiability has proved to be extremely useful in a wide range of different disciplines such as hardware and software verification [10, 76, 105] and planning [71].

We consider propositional formulas in conjunctive normal form (CNF) presented as a finite sets of *clauses* where a clause is a finite set of *literals*, and a literal is a negated or unnegated propositional *variable*. For example,

$$F = \{\{\neg x, y, z\}, \{\neg y, \neg z\}, \{x, \neg y\}\}$$

represents the propositional formula $(\neg x \vee y \vee z) \wedge (\neg y \vee \neg z) \wedge (x \vee \neg y)$. For a literal $\ell$ we denote by $\overline{\ell}$ the literal of opposite polarity, i.e., $\overline{x} = \neg x$ and $\overline{\neg x} = x$. Similarly, for a set $L$ of literals, we put $\overline{L} = \{\, \overline{\ell} : \ell \in L \,\}$. We say that two clauses $C, D$ *overlap* if $C \cap D \neq \emptyset$, and we say that $C$ and $D$ *clash* if $C$ and $\overline{D}$ overlap.

For a clause $C$ we denote by $\mathrm{var}(C)$ the set of variables that occur (negated or unnegated) in $C$; for a CNF formula $F$ we put $\mathrm{var}(F) = \bigcup_{C \in F} \mathrm{var}(C)$. We measure the size of a CNF formula $F$ by its *length* $\sum_{C \in F} |C|$.

A *truth assignment* is a mapping $\tau : X \to \{0, 1\}$ defined on some set $X$ of variables. We extend $\tau$ to literals by setting $\tau(\neg x) = 1 - \tau(x)$ for $x \in X$. $F[\tau]$ denotes the formula obtained from $F$ by removing all clauses that contain a literal $\ell$ with $\tau(\ell) = 1$ and by removing from the remaining clauses all literals $\ell'$ with $\tau(\ell') = 0$; $F[\tau]$ is the *restriction* of $F$ to $\tau$. Note that $\mathrm{var}(F[\tau]) \cap X = \emptyset$ holds for every assignment $\tau : X \to \{0, 1\}$ and every CNF formula $F$. A truth assignment $\tau : X \to \{0, 1\}$ *satisfies* a CNF formula $F$ if $F[\tau] = \emptyset$. A formula $F$ is *satisfiable* if there exists a truth assignment that satisfies $F$; otherwise $F$ is *unsatisfiable*. **SAT** is the problem of deciding whether a given CNF formula is satisfiable.

EXAMPLE 3.1. A classical application of **SAT** is the tautology testing problem. Consider the propositional formula $\Psi = (x \to (y \to x))$. We transform $\Psi$ into a CNF formula using a technique which is due to Tseitin [104]. First we replace the subformula $(y \to x)$ by a new "extension variable" $u$ and produce the set of clauses $F_1 = \{\{y, u\}, \{\neg x, u\}, \{\neg y, x, \neg u\}\}$ which asserts the equivalence $u \equiv (y \to x)$. Next, we replace $(x \to u)$ by a new extension variable $v$ and produce the set of clauses $F_2 = \{\{x, v\}, \{\neg u, v\}, \{\neg x, u, \neg v\}\}$ which asserts the equivalence $v \equiv (x \to u)$. It is easy to see that $\Psi$ is a tautology if and only if the CNF formula $F = F_1 \cup F_2 \cup \{\{\neg v\}\}$ is unsatisfiable.

In general, for every propositional formula $\Psi$ we can obtain in linear time a CNF formula $F$ such that $\Psi$ is a tautology if and only if $F$ is unsatisfiable. □

Obviously, we can decide the satisfiability of a CNF formula $F$ with $n$ variables by checking all $2^n$ truth assignments $\tau : \mathrm{var}(F) \to \{0, 1\}$; surprisingly, no algorithm is known that performs, in the worst case, significantly better than the brute force search. More precisely, no algorithm is known that decides satisfiability of CNF formulas with $n$ variables in time $2^{o(n)}$, and it is believed that no such algorithm exists [69]. State-of-the-art satisfiability solvers, however, *are* capable of solving large CNF formulas originating from real-world applications with thousands of variables. This successful performance of solvers is usually explained by the presence of a "hidden structure" in problem instances that arise from practical applications. The parameterized approach to satisfiability entails the identification of parameters that allow fixed-parameter algorithms and exploit the hidden structure.

Similarly as for constraint satisfaction above, we consider as a *satisfiability parameter* any computable function $p$ that assigns to a CNF formula $F$ a non-negative integer $p(F)$. For satisfiability parameters $p_1, \ldots, p_r$ we consider the following generic parameterized decision problem:

**SAT**$(p_1, \ldots, p_r)$

*Instance:* A CNF formula $F$, non-negative integers $k_1, \ldots, k_r$ with $p_1(F) \leq k_1, \ldots, p_r(F) \leq k_r$.

*Parameters:* $k_1, \ldots, k_r$.

*Question:* Is $F$ satisfiable?

As for constraint satisfaction, we can consider the satisfiability parameter $\mathbf{kern}_{\mathcal{A}}(F)$, the size of the CNF formula $F'$ obtained from $F$ by a polynomial-time algorithm $\mathcal{A}$. The algorithm simplifies a given instance by applying simplification and data-reduction rules and outputs a satisfiability-equivalent CNF formula $F'$. Clearly **SAT**$(\mathbf{kern}_{\mathcal{A}})$ is fixed-parameter tractable by definition.

*Unit propagation* and *pure literal elimination* are the main simplification and data reduction methods used in SAT solvers. If a CNF formula $F$ contains a unit clause $\{\ell\}$, we can simplify it by unit propagation to the satisfiability-equivalent CNF formula $F[\ell = 1]$. Note that $F[\ell = 1]$ may contain new unit clauses that where not present in $F$. A literal $\ell$ is a *pure literal* of a CNF formula $F$ if some clauses of $F$ contain $\ell$ but no clause contains $\overline{\ell}$. If $\ell$ is a pure literal of $F$, then we can simplify $F$ by obtaining the satisfiability-equivalent CNF formula $F[\ell = 1]$. Also pure literal elimination propagates.

### 3.2. Treewidth and related parameters

We associate certain graphs and hypergraphs with CNF formulas similarly as for constraint satisfaction. The *primal graph* $G_F$ of a CNF formula is the graph with vertex set $\mathrm{var}(F)$ where two variables $x, y$ are joined by an edge if and only if $F$ contains a clause $C$ with $x, y \in \mathrm{var}(C)$. The *incidence graph* $G^*_F$ 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 $v \in \mathrm{var}(C)$. The *directed incidence graph* $D^*_F$ is obtained from $G^*_F$ by orienting an edge $xC$ from $x$ to $C$ if $\neg x \in C$ and from $C$ to $x$ if $x \in C$. Finally, the *formula hypergraph* $H_F$ is the hypergraph with vertex set $\mathrm{var}(F)$ and hyperedges $\mathrm{var}(C)$ for all $C \in F$. We define the satisfiability parameters $\mathbf{tw}(F) = \mathrm{tw}(G_F)$ and $\mathbf{tw}^*(F) = \mathrm{tw}(G^*_F)$.

If a CNF formula $F$ has clauses of bounded size, then we can transform $F$ into a constraint satisfaction instance $I$ such that $F$ is satisfiable if and only if $I$ is consistent. Namely, we replace each clause $C$ with a constraint whose constraint relation contains $2^{|C|} - 1$ tuples in the obvious way. However, if the size of clauses is unbounded, then this transformation yields an exponential blow-up of the instance size. It is known that the treewidth of a graph $G$ plus 1 is an upper bound for the size of a clique in $G$. Note also that a clause of size $s$ yields a clique of size $s$ in the primal graph. Hence, if the treewidth of the primal graph of a CNF formula $F$ is at most $k$, then the size of clauses of $F$ is bounded by $k + 1$. In other words, bounding the treewidth of primal formula graphs automatically bounds the size of clauses. Consequently, the above transformation does not cause an exponential blow-up for CNF formulas whose primal graphs have bounded treewidth. Thus Theorem 2.2 immediately gives us the following result.

THEOREM 3.1. *The problem* **SAT**(**tw**) *is fixed-parameter tractable.*

Alekhnovich and Razborov [4] have pointed out the significance of the hypergraph parameter *branchwidth* for satisfiability. We briefly recall the definition of branchwidth. Let $H$ be a hypergraph, $T = (V, E)$ a ternary tree (i.e., all vertices of $T$ have either degree 1 or 3), and $\tau$ a one-to-one mapping from the set of leaves of $T$ to the set of hyperedges of $H$; $(T, \tau)$ is called a *branch decomposition* of $H$. The *order* of an edge $e$ of $T$ is the number of vertices of $H$ which belong to hyperedges $\tau(t_1), \tau(t_2)$ for nodes $t_1$ and $t_2$ that belong to different components of $T - e$. The *width* of a branch decomposition $(T, \tau)$ is the maximum order of all edges of $T$; the *branchwidth* $\mathrm{bw}(H)$ of $H$ is the smallest width over all its branch decompositions. The branchwidth of a CNF formula is the branchwidth of its formula hypergraph, that is, $\mathbf{bw}(F) = \mathrm{bw}(H_F)$. The following is a consequence of the results of Alekhnovich and Razborov [4].

THEOREM 3.2 ([4]). *The problem* **SAT**(**bw**) *is fixed-parameter tractable.*

This result has been obtained via a modification of Robertson and Seymour's algorithm for computing branch-decompositions [92]. Bacchus, Dalmao, and Pitassi [7] have proposed an efficient fixed-parameter algorithm for CNF formulas of bounded branchwidth. This algorithm, which actually computes the number of satisfying assignments, is based on the DPLL procedure and uses caching techniques for an efficient reuse of solutions for subproblems.

Using a result of Robertson and Seymour [91], one can show the following lemma [100].

LEMMA 3.1. *For every CNF formula $F$ without pure literals we have*

$$\mathbf{bw}(F) \leq \mathbf{tw}(F) + 1 \leq 2 \cdot \mathbf{bw}(F).$$

Thus Theorems 3.1 and 3.2 are essentially equivalent since pure literals can be eliminated in polynomial time without increasing the treewidth.

### 3.3. Treewidth and clique-width

Similarly as for constraint satisfaction (see the previous section) bounding the treewidth of incidence graphs gives larger classes of CNF formulas than bounding the treewidth of primal graphs. Indeed satisfiability

is fixed-parameter tractable for the more general parameter.

THEOREM 3.3. *The problem* $\mathbf{SAT}(\mathbf{tw}^*)$ *is fixed-parameter tractable.*

This theorem can be shown by means of linear programming on tree decompositions of incidence graphs [42, 94]. However, using general results of descriptive complexity theory, one can easily derive fixed-parameter tractability of $\mathbf{SAT}(\mathbf{tw}^*)$. Next we will outline this approach (a variant was proposed by Szeider [100]).

Courcelle [23] has shown that if a graph property can be expressed in a certain fragment of monadic second order logic, $\mathrm{MSO}_2$, then this property can be checked in linear time for graphs of bounded treewidth; see also Downey and Fellows' book [30] for an elegant proof of this result. $\mathrm{MSO}_2$ admits quantification over sets of vertices and over sets of edges. $\mathrm{MSO}_1$ denotes the weaker fragment of monadic second order logic where only quantification over sets of vertices is allowed; for our purposes it suffices to consider $\mathrm{MSO}_1$.

EXAMPLE 3.2. Whether a graph $G = (V, E)$ is 3-colorable can be expressed by the $\mathrm{MSO}_1$ formula

$$(\exists R \subseteq V)(\exists G \subseteq V)(\exists B \subseteq V)[\mathrm{partition}(R, G, B) \\ \wedge \mathrm{proper}(R) \wedge \mathrm{proper}(G) \wedge \mathrm{proper}(B)]$$

using the abbreviations

$$\mathrm{partition}(R, G, B) \equiv \\ R \cap G = \emptyset \wedge R \cap B = \emptyset \\ \wedge B \cap G = \emptyset \wedge R \cup G \cup B = V,$$

$$\mathrm{proper}(X) \equiv \\ \forall u, v \in V(uv \in E \rightarrow \neg(u \in X \wedge v \in X)).$$

Hence, it follows from Courcelle's Theorem that the problem of deciding whether a graph of treewidth $k$ is 3-colorable is fixed-parameter tractable with respect to the parameter $k$. □

We establish Theorem 3.3 using the following graph theoretic formulation of satisfiability. For a CNF formula $F$ we obtain the graph $G_\triangle(F)$ with vertex set $\{ x, \neg x : x \in \mathrm{var}(F) \} \cup F \cup \{ a_C, b_C : C \in F \}$; edges are defined such that (i) each literal is adjacent with its complement and with all clauses that contain the literal, and (ii) each clause $C$ forms a triangle with the vertices $a_C$ and $b_C$. The purpose of the "auxiliary" vertices $a_C, b_C$ is to distinguish between vertices representing clauses and vertices representing literals. It is straightforward to state an $\mathrm{MSO}_1$ formula $\Psi$ that expresses the following property of $G_\triangle(F)$:

> "There exists a set $V_1$ of literals such that for every literal $\ell \in V_1$ its complement $\overline{\ell}$ is not in $V_1$, and every clause is adjacent with at least one literal in $V_1$."

It is not difficult to see that $F$ is satisfiable if and only if $\Psi$ is true for $G_\triangle(F)$, and that $\mathrm{tw}(G_\triangle(F))$ is bounded in terms of $\mathrm{tw}(G^*(F))$. Hence, Theorem 3.3 follows by Courcelle's Theorem.

Courcelle's Theorem provides a very general and powerful means for showing fixed-parameter tractability. However it is important to note that the running times of algorithms obtained by this result involve huge hidden constants which make an implementation difficult and limits the practical applicability. However, once the fixed-parameter tractability of a problem is established, one can try to develop made-to-measure combinatorial algorithms for the problem under consideration.

Clique-width is a graph parameter that is more general than treewidth in the sense that every graph class of bounded clique-width has also bounded treewidth, but there are graph classes of bounded clique-width and unbounded treewidth. Since a result similar to Courcelle's Theorem holds for clique-width, clique-width gives raise to a more general parameterization of the satisfiability problem. Before we state the relevant results in detail we give the definition of clique-width.

Let $k$ be a positive integer. A *$k$-graph* is a graph whose vertices are labeled by integers from $\{1, \ldots, k\}$. We consider an arbitrary graph as a $k$-graph with all vertices labeled by 1. We call the $k$-graph consisting of exactly one vertex $v$, labeled by $i \in \{1, \ldots, k\}$, an *initial $k$-graph* and denote it by $i(v)$. The *clique-width* $\mathrm{cwd}(G)$ of a graph $G$ is the smallest integer $k$ such that $G$ can be constructed from initial $k$-graphs by means of repeated application of the following three operations.

(i) *Disjoint union* (denoted by $\oplus$);
(ii) *Relabeling*: changing all labels $i$ to $j$ (denoted by $\rho_{i \rightarrow j}$);
(iii) *Edge insertion*: connecting all vertices labeled by $i$ with all vertices labeled by $j$, $i \neq j$ (denoted by $\eta_{i,j}$).

A construction of a $k$-graph using the above operations can be represented by an algebraic term composed of $i(v)$, $\oplus$, $\rho_{i \rightarrow j}$, and $\eta_{i,j}$, $(i, j \in \{1, \ldots, k\}$, and $i \neq j)$. Such a term is called a *$k$-expression defining $G$*. Thus, the clique-width of a graph $G$ is the smallest integer $k$ such that $G$ can be defined by a $k$-expression.

EXAMPLE 3.3. The clique $K_4$ on the vertices $u, v, w, x$ is defined by the 2-expression

$$\rho_{2 \rightarrow 1}(\eta_{1,2}(\rho_{2 \rightarrow 1}(\eta_{1,2}(\rho_{2 \rightarrow 1}(\eta_{1,2}( \\ 2(u) \oplus 1(v))) \oplus 2(w))) \oplus 2(x))).$$

Hence $\mathrm{cwd}(K_4) \leq 2$. In general, every clique on two or more vertices has clique-width 2. □

Similarly one can define the *clique-width of a directed graph $D$* as the smallest integer $k$ such that $D$ can be defined by a $k$-expression where the operation $\eta_{i,j}$ connects all vertices labeled by $i$ with all vertices labeled by $j$ by edges *directed* from $i$-labeled vertices to $j$-labeled vertices.

Courcelle, Makowsky, and Rotics [24] have shown that any property for graphs (or directed graphs) definable in $\mathrm{MSO}_1$ can be checked in linear time if a $k$-expression of the given graph is provided as an input; here $k$ is considered as a constant. This result immediately applies to satisfiability via directed incidence graphs. Namely, a CNF formula $F$ is satisfiable if and only if the following property holds for the directed incidence graph $D^*(F)$:

"The set of variables can be partitioned into two sets $V_0, V_1$ such that for every clause $C$ there exists either (i) a variable $x \in V_0$ and a directed edge $(x, C)$ or (ii) a variable $x \in V_1$ and a directed edge $(C, x)$."

Note that the partition $V_0, V_1$ corresponds to a satisfying assignment $\tau$ with $\tau(x) = i$ for $x \in V_i$, $i = 0, 1$. Considering the satisfiability parameter $\mathbf{dcwd}^*(F) = \mathrm{cwd}(D^*(F))$ we do not have yet fixed-parameter tractability of $\mathbf{SAT}(\mathbf{dcwd}^*)$ since the result of Courcelle et al. [24] requires a $k$-expression as an input. Determining the clique-width of a given graph is NP-hard, as shown by Fellows, Rosamond, Rotics, and Szeider [40]. However, Oum and Seymour provide a fixed-parameter algorithm for approximating clique-width.

THEOREM 3.4 ([84]). *Given a graph $G$ of order $n$ and clique-width $k$ one can find a $(2^{3k+2}-1)$-expression for $G$ in time $O(n^9 \log n)$.*

As observed by Fischer, Makowsky, and Ravve [42], the proof of Theorem 3.4 carries over to directed graphs. Hence, combining the $\mathrm{MSO}_1$ result of [24] with Theorem 3.4, one obtains fixed-parameter tractability of $\mathbf{SAT}(\mathbf{dcwd}^*)$.

THEOREM 3.5 ([42]). *The problem $\mathbf{SAT}(\mathbf{dcwd}^*)$ is fixed-parameter tractable.*

From the theoretical point of view, Theorem 3.5 subsumes Theorems 3.1 and 3.3 since there exists a computable function $f$ such that $\mathbf{tw}^*(F) \leq k$ implies $\mathbf{dcwd}^*(F) \leq f(k)$. The latter follows from a result of Courcelle and Olariu [25].

We note that Theorem 3.5 can also be obtained using Theorem 3.4 for undirected graphs since it is easy to show that $\mathrm{cwd}(G_\triangle(F))$ is bounded in terms of $\mathrm{cwd}(D^*(F))$.

### 3.4. Maximum deficiency and falsum number

Matchings in incidence graphs give raise to the satisfiability parameter *maximum deficiency*, first considered by Franco and Van Gelder [47]. A set $M$ of edges of a graph $G$ is a *matching* if each vertex of $G$ is incident with at most one edge in $M$. A matching of largest size of a graph $G$ is a *maximum matching* of $G$. It is well known that a maximum matching can be found in polynomial time. The following observation is due to Aharoni and Linial [2].

LEMMA 3.2. *If the incidence graph of a CNF formula $F$ has a matching $M$ such that every clause of $F$ is incident with an edge of $M$, then $F$ is satisfiable.*

Namely, given such a matching, we can satisfy each clause independently by choosing the right truth value for the variable to which the clause is matched. The *maximum deficiency* $\mathbf{md}(F)$ of a CNF formula $F$ is the number of clauses of $F$ that are not incident with an edge of an arbitrary but fixed maximum matching of the incidence graph of $F$. Lemma 3.2 covers the case $\mathbf{md}(F) = 0$. Defining the *deficiency* of a CNF formula $F$ as the difference $\mathbf{d}(F) = |F| - |\mathrm{var}(F)|$, the following equation, which can be established using Hall's Theorem, explains the origin of the term "maximum deficiency"

$$\mathbf{md}(F) = \max_{F' \subseteq F} \mathbf{d}(F').$$

Fleischner, Kullmann, and Szeider [43] have shown that $\mathbf{SAT}(\mathbf{md})$ is in XP. Szeider [99] has improved this result to fixed-parameter tractability.

THEOREM 3.6 ([99]). *The problem $\mathbf{SAT}(\mathbf{md})$ is fixed-parameter tractable.*

This result can be applied to the recognition of *minimal unsatisfiable* CNF formulas (i.e., CNF formulas that are unsatisfiable but become satisfiable by removing any clause). In general, recognition of minimal unsatisfiable CNF formulas is DP-complete [86]. A problem is in DP if and only if it can be formulated as the conjunction of an NP-problem and a co-NP-problem [85]. A canonical DP-problem is the following: given two CNF formulas $F_1, F_2$, determine whether $F_1$ is satisfiable and $F_2$ is unsatisfiable. Deficiency and maximum deficiency agree for minimal unsatisfiable CNF formulas [2, 43, 99]. Hence, using the algorithm of Theorem 3.6 as a subroutine, one can obtain the following result.

THEOREM 3.7 ([99]). *The recognition of minimal unsatisfiable CNF formulas is fixed-parameter tractable with respect to the parameter deficiency.*

Note that the satisfiability parameters $\mathbf{md}$ and $\mathbf{dcwd}^*$ are *incomparable* in the following sense. There are CNF formulas with constant $\mathbf{md}$ and arbitrarily large $\mathbf{dcwd}^*$, and conversely, there are CNF formulas with arbitrarily large $\mathbf{md}$ and constant $\mathbf{dcwd}^*$ [99]. Of related interest is a recent paper of Szeider [102] where the relationship between maximum deficiency and the size of backdoor sets is considered.

Next we consider propositional formulas that are not in conjunctive normal form. A propositional formula $\psi$ is called $\mathbf{f}$-*implicational* if $\rightarrow$ (implication) is the only connective of $\psi$; however, $\psi$ may contain the constant $\mathbf{f}$ (falsum) that is always mapped to 0 by any truth assignment. Franco, Goldsmith, Schlipf, Speckenmeyer, and Swaminathan [46] have shown the following.

THEOREM 3.8 ([46]). *Satisfiability of* $\mathbf{f}$*-implicational formulas with at most two occurrences of each variable and* $k$ *occurrences of* $\mathbf{f}$ *is fixed-parameter tractable with respect to the parameter* $k$*.*

Heusch, Porschen, and Speckenmeyer [68] have improved the algorithm of Franco et al. using dynamic programming techniques.

Szeider [100] has considered a canonical translation of CNF formulas $F$ into $\mathbf{f}$-implicational formulas $\psi_F$ with at most two occurrences of each variable. The *falsum number* $\mathbf{fn}(F)$ of a CNF formula $F$ is the number of occurrences of $\mathbf{f}$ in $\psi_F$. Theorem 3.8 yields directly the fixed-parameter tractability of $\mathbf{SAT}(\mathbf{fn})$.

THEOREM 3.9 ([100]). *The problem* $\mathbf{SAT}(\mathbf{fn})$ *is fixed-parameter tractable.*

Since $\mathbf{md}(F) \leq \mathbf{fn}(F)$ holds for CNF formulas $F$ without pure literals [100], Theorem 3.6 subsumes Theorem 3.9.

### 3.5. Backdoor sets

We have already considered the concept of backdoor sets in the section on constraint satisfaction. This concept also applies to satisfiability [107]. Consider a class $\Gamma$ of CNF formulas. A set $B$ of variables of a CNF formula $F$ is a *strong* $\Gamma$*-backdoor set* of $F$ if for all truth assignments $\tau : B \to \{0,1\}$, the restriction $F[\tau]$ belongs to $\Gamma$. Similarly as above, let $\mathbf{b}_\Gamma(F)$ denote the size of a smallest strong $\Gamma$-backdoor set of $F$. For a base class $\Gamma$ we consider the following parameterized problem.

> $\mathbf{SBDS}(\Gamma)$
>
> *Instance:* A CNF formula $F$, a non-negative integer $k$.
>
> *Parameter:* $k$.
>
> *Question:* Does $F$ have a strong $\Gamma$-backdoor set of size at most $k$? I.e., is $\mathbf{b}_\Gamma(F) \leq k$?

Let $\Gamma$ be a class of CNF formulas for which satisfiability can be decided in polynomial time. Assume that $\mathbf{SBDS}(\Gamma)$ is fixed-parameter tractable in the sense that a fixed-parameter algorithm finds a strong $\Gamma$-backdoor set of size at most $k$ if such set exists. Then, clearly, $\mathbf{SAT}(\mathbf{b}_\Gamma(F))$ is fixed-parameter tractable.

For the base classes HORN (CNF formulas where each clause contains at most one positive literal) and 2CNF (CNF formulas with clauses of size at most 2), Nishimura, Ragde, and Szeider [82] have shown the following.

THEOREM 3.10 ([82]). *The problems* $\mathbf{SBDS}(\Gamma)$ *and* $\mathbf{SAT}(\mathbf{b}_\Gamma(F))$ *are fixed-parameter tractable for* $\Gamma \in \{\mathrm{HORN}, \mathrm{2CNF}\}$.

The algorithms of Nishimura et al. make use of the concept of deletion backdoor sets. For a CNF formula $F$ and a set of variables $B \subseteq \mathrm{var}(F)$, let $F - B$ denote the CNF formula $\{ C \setminus (B \cup \overline{B}) : C \in F \}$. We say that $B$ is a *deletion* $\Gamma$*-backdoor set* of $F$ if $F - B \in \Gamma$. We denote the size of a smallest deletion $\Gamma$-backdoor set of $F$ by $\mathbf{db}_\Gamma(F)$.

Consider a base class $\Gamma$ with the property that all subsets of a CNF formula of $\Gamma$ also belong to $\Gamma$; in that case we say that $\Gamma$ is *clause-induced*. Since $F[\tau] \subseteq F - B$ holds for every truth assignment $B \to \{0,1\}$, it follows that for a clause-induced base class $\Gamma$, each deletion $\Gamma$-backdoor set is also a strong $\Gamma$-backdoor set. Often it is easier to search for deletion backdoor sets than for strong backdoor sets; however, smallest deletion backdoor sets can be larger than smallest strong backdoor sets.

For the base classes HORN and 2CNF we are lucky: it can be shown that strong and deletion backdoor sets for these two classes coincide [82]. Consequently, the search for strong HORN or 2CNF-backdoor sets can be accomplished by a simple search-tree algorithm. Let $F$ be a CNF formula that contains a non-Horn clause $C$; i.e., $C$ contains at least two positive literals $x, y$. Every deletion HORN-backdoor set of $F$ must contain either $x$ or $y$. Thus we can branch into cases $F - \{x\}$ and $F - \{y\}$ and repeat recursively with the parameter $k-1$. The branching step for the detection of deletion 2CNF-backdoor sets is similar: we take a clause $C$ of size at least 3 and pick three distinct variables $x, y, z \in \mathrm{var}(C)$. We branch into cases $F - \{x\}$, $F - \{y\}$, and $F - \{z\}$, and repeat recursively with the parameter $k - 1$.

In a recent paper, Samer and Szeider [95] have generalized the notion of backdoor sets to Quantified Boolean Formulas (QBF) and have show that Theorem 3.10 also holds in this more general setting.

Consider the class UP of CNF formulas that can be decided by unit propagation; that is, by repeated application of unit propagation to $F \in \mathrm{UP}$ we can either reduce $F$ to the empty CNF formula ($F$ is satisfiable) or we can reduce $F$ to a CNF formula that contains the empty clause ($F$ is unsatisfiable). It is well known that every unsatisfiable Horn formula belongs to UP [72]. Thus, a possible generalization of Theorem 3.10 is to consider strong UP-backdoor sets. However, Szeider [101] has shown that the detection of strong UP-backdoor sets is a W[P]-complete problem. The same holds true for the class PL of CNF formulas decidable by pure literal elimination, and the class UP + PL of CNF formulas decidable by a combination of unit propagation and pure literal elimination.

THEOREM 3.11 ([101]). *The problems* $\mathbf{SBDS}(\mathrm{UP})$, $\mathbf{SBDS}(\mathrm{PL})$, *and* $\mathbf{SBDS}(\mathrm{UP} + \mathrm{PL})$ *are* W[P]*-complete.*

Note that for the classes UP, PL, and UP + PL, deletion backdoor sets are not always strong backdoor sets (the classes are not clause-induced). Hence deletion backdoor sets do not provide a way for overcoming the limits of Theorem 3.11.

Let RHORN denote the class of *renamable Horn* CNF formulas, i.e., of CNF formulas $F$ for which there exists a set $X \subseteq \mathrm{var}(F)$ such that, replacing in the clauses of $F$ the literal $x$ by $\neg x$ and the literal $\neg x$ by $x$ whenever $x \in X$, yields a Horn formula. Obviously, RHORN properly contains HORN. In turn, from an unsatisfiable formula $F \in$ RHORN we can derive the empty clause by unit propagation, i.e, $F \in$ UP [72]. The parameterized complexity of **SBDS**(RHORN) is an interesting open question, as it lies in a certain sense between the fixed-parameter tractable **SBDS**(HORN) and the W[P]-complete **SBDS**(UP). Since RHORN is clause-induced, one can also study the relaxed problem of deciding whether a given CNF formula has a deletion RHORN-backdoor set of size at most $k$. The latter problem is equivalent (under parameterized reductions) to the problem of deciding whether a graph $G$ with a perfect matching $M$ has a vertex cover of size at most $|M|/2 + k$. Note that for such a graph $G$, $|M|/2$ is a *guaranteed value* for the size of a vertex cover; parameterizations above guaranteed value have recently received a lot of attention [77, 66, 65].

A further interesting base class is the class CLU of *cluster formulas*. A CNF formula is a cluster formula if it is the variable-disjoint union of *hitting formulas* which are CNF formulas where any two clauses of clash [73].

The next lemma is due to an observation of Iwama [70].

LEMMA 3.3. *A hitting formula $F$ with $n$ variables has exactly $2^n - \sum_{C \in F} 2^{n-|C|}$ satisfying assignments $\tau : \mathrm{var}(F) \to \{0, 1\}$.*

Consequently, we can decide the satisfiability of hitting formulas, and in turn, the satisfiability of cluster formulas, in polynomial time.

Nishimura, Ragde, and Szeider [83] have considered the parameterized complexity of backdoor set detection for the base class CLU. Note that CLU is clause-induced, hence both strong and deletion backdoor sets are relevant.

THEOREM 3.12 ([83]). *The problem* **SBDS**(CLU) *is* W[2]*-hard.*

This negative result follows by a reduction from the parameterized hitting set problem. The relaxed parameter $\mathbf{db}_{\mathrm{CLU}}$ admits a fixed-parameter algorithm.

THEOREM 3.13 ([83]). *The problem* $\mathbf{SAT}(\mathbf{db}_{\mathrm{CLU}})$ *is fixed-parameter tractable.*

This result is obtained by means of an algorithm that systematically destroys certain obstructions that consist of pairs or triples of clauses. To this end, the *obstruction graph* of a CNF formula $F$ is considered. The vertex set of this graph is the set of variables of $F$; two variables $x, y$ are joined by an edge if and only if at least one of the following conditions hold.

(i) $F$ contains two clauses $C_1, C_2$ that do not clash, $x \in \mathrm{var}(C_1 \cap C_2)$, and $y \in \mathrm{var}(C_1 \setminus C_2)$;
(ii) $F$ contains three clauses $C_1, C_2, C_3$ such that $C_1$ and $C_3$ do not clash, $x \in \mathrm{var}((C_1 \setminus C_3) \cap \overline{C_2})$, and $y \in \mathrm{var}((C_3 \setminus C_1) \cap \overline{C_2})$.

Vertex covers of obstruction graphs are closely related to backdoor sets: Every deletion CLU-backdoor set of a CNF formula $F$ is a vertex cover of the obstruction graph of $F$. Conversely, every vertex cover of the obstruction graph of a CNF formula $F$ is a strong CLU-backdoor set of $F$.

The satisfiability parameter $\mathbf{clu}(F)$, the *clustering-width*, is defined as the size of a smallest vertex cover of the obstruction graph of $F$. Hence the following inequalities hold for every CNF formula $F$.

$$\mathbf{b}_{\mathrm{CLU}}(F) \leq \mathbf{clu}(F) \leq \mathbf{db}_{\mathrm{CLU}}(F).$$

Finding a size $k$ vertex cover of a given graph (or deciding that such a vertex cover does not exist) is fixed-parameter tractable, see Example 1.1. Since every vertex cover of the obstruction graph of a CNF formula $F$ is a strong CLU-backdoor set of $F$, we have the following.

THEOREM 3.14 ([83]). *The problem* $\mathbf{SAT}(\mathbf{clu})$ *is fixed-parameter tractable.*

Moreover, Theorem 3.13 follows, since we always have $\mathbf{clu}(F) \leq \mathbf{db}_{\mathrm{CLU}}(F)$. There are classes of CNF formulas with constant **clu** and unbounded $\mathbf{dcwd}^*$, and classes of CNF formulas with unbounded **clu** and constant $\mathbf{dcwd}^*$ [83]. Hence the clustering number is incomparable with any of the satisfiability parameters **tw**, $\mathbf{tw}^*$, **bw**, and $\mathbf{dcwd}^*$.

## 4. LOGIC PROGRAMMING AND NON-MONOTONIC REASONING

### 4.1. Computing stable models

A well-studied form of non-monotonic reasoning is logic programming under the *stable model semantics*. Before we survey parameterized problems that arise in this context we briefly review some concepts and definitions. Here we restrict our scope to propositional logic programming.

A *literal* is an atom $A$ or a negated atom $\neg A$. A *disjunctive logic program* (DLP) $P$ is a finite set of *rules* of the form

$$A'_1 \vee \cdots \vee A'_r \longleftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg A_{m+1} \wedge \cdots \wedge \neg A_n$$

where $A, A_1, \ldots, A_n, A'_1, \ldots, A'_r$ are atoms [88]. Here the head $H(r)$ of a rule $r$ of $P$ is the set of atoms appearing at the left-hand-side of $r$, and the body $B(r)$ is the set of atoms at the right-hand-side of $r$. We also write $H(P)$ (respectively $B^-(P)$) for the union of $H(r)$ (respectively $B^-(r)$) for all rules $r$ of $P$. If $|H(r)| = 1$ for all rules $r$ of a disjunctive logic program $P$, we call $P$

a *normal logic program*. A DLP is *positive* if $B^-(r) = \emptyset$ holds for all rules $r$. A positive normal logic program is a *Horn program*.

The body $B(r)$ of a rule is partitioned into the set of unnegated atoms $B^+(r)$ and the set of negated atoms $B^-(r)$. The *universe* $U(P)$ of such a program $P$ is the set of all atoms occurring in $P$, and an *interpretation* $I$ of $P$ is a subset of $U(P)$ which fixes a set of atoms that are *true* according to $I$, while the atoms in $U(P) \setminus I$ are *false* according to $I$. An interpretation $I$ of a disjunctive logic program $P$ *satisfies a rule* $r$ if $H(r) \cap I \neq \emptyset$ or $B^+(r) \setminus I \neq \emptyset$ or $B^-(r) \cap I \neq \emptyset$. If $I$ satisfies all the rules of $P$ we call $I$ a *model* of $P$.

The *reduct* $P^I$ of a disjunctive logic program $P$ under an interpretation $I$ is obtained by performing the following two steps: first we remove every rule $r$ such that $B^-(r) \cap I \neq \emptyset$; second we remove all negated atoms from the bodies of the remaining rules. The reduct of a disjunctive logic program is a positive logic program, but not necessarily a Horn program, given the possible disjunctions in the rule heads. Thus the reduct $P^I$ may have *several* minimal models with respect to set-inclusion. However, the reduct of a normal logic program is a Horn program. Since the intersection of two models of a Horn program $P$ is again a model of $P$, it follows that a satisfiable Horn program $P$ has a unique smallest model with respect to set-inclusion, the *least model* of $P$.

Now, an interpretation $I$ is called a *stable model* of the disjunctive logic program $P$ if $I$ happens to coincide with a minimal model of $P^I$ (or *the* minimal model of $P^I$, if $P$ is normal). We say that a formula $\varphi$ is a *brave consequence* of a disjunctive logic program $P$ if there is at least one stable model of $P$ such that $M$ satisfies $\varphi$, in symbols, $M \models \varphi$. We say that a formula $\varphi$ is a *cautious consequence* of $P$ if $M \models \varphi$ for every stable model $M$ of $P$.

EXAMPLE 4.1 ([60]). Consider the normal logic program $P$ consisting of the rules

$$\begin{array}{lcl} p & \longleftarrow & q \wedge \neg r \\ r & \longleftarrow & q \wedge s \\ s & \longleftarrow & \neg q \wedge r \\ q & \longleftarrow & \neg u \\ u & \longleftarrow & \neg q. \end{array}$$

The interpretation $I = \{u\}$ is a stable model for $P$, since the reduct $P^I$ consisting of the rules

$$\begin{array}{lcl} p & \longleftarrow & q \\ r & \longleftarrow & q \wedge s \\ s & \longleftarrow & r \\ u & \longleftarrow & \end{array}$$

is the least model of $P^I$. The only further stable model of $P$ is the interpretation $\{p, q\}$. □

### 4.2. Small and large stable models

The following parameterized problems were considered by Truszczyński [103].

**SmallStableModel**

*Instance:* A normal logic program $P$ and a non-negative integer $k$.

*Parameter:* $k$.

*Question:* Does $P$ admit a stable model of size at most $k$?

**LargeStableModel**

*Instance:* A normal logic program $P$ and a non-negative integer $k$.

*Parameter:* $k$.

*Question:* Does $P$ admit a stable model of size at least $|P| - k$?

For solving **SmallStableModel** it suffices to check all sets of atoms of $P$ of size at most $k$. The number of such subsets is bounded by $O(n^k)$ for $n = |U(P)|$, thus the problem is in XP. A more refined analysis [75] puts it into W[2]; a reduction from weighted CNF satisfiability determines the problem as W[2]-complete.

THEOREM 4.1 ([75, 103]). *The problem* **SmallStableModel** *is* W[2]*-complete.*

Next we consider the problem **LargeStableModel**. Consider a normal logic program $P$ with $m = |P|$ rules. Let $k$ be a non-negative integer. We apply two preprocessing steps: We obtain a logic program $P_1$ from $P$ by removing from the bodies of rules of $P$ all literals $\neg A$ for which $A \notin H(P)$ holds. This removal does not effect the set of stable models. Furthermore, we obtain a logic program $P_2$ from $P_1$ by removing all rules $r$ from $P_1$ for which $|B^-(r)| > k$ holds. The following can be shown:

(i) An interpretation $I$ containing at least $m-k$ atoms is a stable model of $P_1$ if and only if it is a stable model of $P_2$.
(ii) If a stable model $I$ of $P_2$ contains at least $m - k$ atoms, then $|B^-(P_2)| \leq k + k^2$.

Thus, if $|B^-(P_2)| > k + k^2$ we can reject the instance. However, each stable model of $P_2$ is determined by a subset of $B^-(P_2)$. Consequently, if $|B^-(P_2)| \leq k + k^2$, then the number of such subsets is bounded by $O(2^{k+k^2})$. Hence we have the following result.

THEOREM 4.2 ([103]). *The problem* **LargeStableModel** *is fixed-parameter tractable.*

Lonc and Truszczyński [75] have determined the parameterized complexity of several model search problems for general logic programs, definite Horn programs, and purely negative programs, distinguishing

the search for models, supported models, and stable models (see Lonc and Truszczyński's paper for the respective definitions).

### 4.3. Structural parameters for logic programs

Similarly as for satisfiability and constraint satisfaction above, we define a generic parameterized problem for stable models; here $p_1, \ldots, p_r$ denote computable mappings that assign to a logic program $P$ a non-negative integer $p_i(P)$.

> **SM**$(p_1, \ldots, p_r)$
> *Instance:* A normal logic program $P$ and non-negative integers $k_1, \ldots, k_r$ with $p_1(P) \leq k_1, \ldots, p_r(P) \leq k_r$.
> *Parameters:* $k_1, \ldots, k_r$.
> *Question:* Does $P$ admit a stable model?

Gottlob, Scarcello, and Sideri [60] have considered (directed) graphs to model the structure of logic programs. The *dependency graph* $D(P)$ is the directed graph with vertex set $U(P)$ and two types of arcs defined as follows. There is a *positive arc* from $A$ to $B$ if there exists a rule $r \in P$ with $\{A\} = H(r)$ and $B \in B^+(r)$; there is a *negative arc* from $A$ to $B$ if there exists a rule $r \in P$ with $\{A\} = H(r)$ and $B \in B^-(r)$. From $D(P)$ one obtains the *undirected dependency graph* $G(P)$ by replacing each positive arc running from $A$ to $B$ by an edge between $A$ and $B$, and by replacing each negative arc from $A$ to $B$ by an induced path of length two from $A$ to $B$, introducing a new vertex $v_{AB}$. The parameter **fvs**$(P)$ is defined as the size of a smallest feedback vertex set of $G(P)$.

THEOREM 4.3 ([60]). *The problem* **SM**(**fvs**) *is fixed-parameter tractable.*

The proof of this result follows from the following considerations. First, it is easy to see that if **fvs**$(P) \leq k$, then the undirected dependency graph $G(P)$ has a feedback vertex set $S$ of size at most $k$ containing only atoms, since each vertex of the form $v_{AB}$ can be replaced by one of its two neighbors. Given such a feedback vertex set $S \subseteq U(P)$ one considers, one after the other, all assignments $\tau : S \to \{0, 1\}$ and the corresponding restriction $P[\tau]$ of $P$ to $\tau$. The *restriction* $P[\tau]$ is defined similarly as for CNF formulas, see Section 3. It can be shown that if $I$ is a stable model of $P$, then it is also a stable model of $P[\tau]$. Moreover, since the undirected dependency graph of $P[\tau]$ is acyclic, $P[\tau]$ is a "stratified" program and has a unique stable model $M_\tau$ which can be computed in linear time. We check whether $M_\tau$ is a stable model of $P$, and, if so, we add it to a cumulating set $\Sigma$. When we have considered all assignments $\tau : S \to \{0, 1\}$ we are left with the set $\Sigma$ of all stable models of $P$. Hence **SM**(**fvs**) is fixed-parameter tractable. As we have actually computed all stable models, we can now determine whether a given atom belongs to some stable model (*brave reasoning*) and whether a given atom belongs to all stable models (*cautious reasoning*).

The fixed-parameter tractability of the **SM** problem has also been explored under the treewidth parameter. This was actually done in the more general context of disjunctive logic programs.

EXAMPLE 4.2 (adapted from [26]). Consider the following disjunctive logic program $P$, describing the behavior of a reviewer while reviewing a paper:

$$\begin{aligned} \mathit{good} \vee \mathit{bad} &\leftarrow \mathit{paper} \\ \mathit{tired} &\leftarrow \neg \mathit{angry} \wedge \neg \mathit{happy} \\ \mathit{happy} &\leftarrow \mathit{good} \\ \mathit{angry} &\leftarrow \mathit{bad} \\ \mathit{smoke} &\leftarrow \mathit{happy} \\ \mathit{smoke} &\leftarrow \mathit{angry} \\ \mathit{paper} &\leftarrow \end{aligned}$$

Intuitively, here *paper* means that the reviewer currently reviews a paper, *good* means that the paper is good, *bad*, that it is bad, and so on. As one can easily verify, there are only two stable models of $P$, which are:

$$M_1 = \{\mathit{paper}, \mathit{good}, \mathit{happy}, \mathit{smoke}\} \text{ and}$$
$$M_2 = \{\mathit{paper}, \mathit{bad}, \mathit{angry}, \mathit{smoke}\}.$$

Thus, for example, *happy* and *angry* are both brave consequences of $P$, but none of the two is a cautious consequence of $P$. However $\mathit{happy} \vee \mathit{angry}$ is a cautious consequence of $P$, and so is *smoke*. □

Note that each propositional logic program is also a disjunctive logic program, and thus disjunctive logic programming is a true generalization of propositional logic programming as introduced before. It immediately follows that all fixed-parameter tractability results for disjunctive logic programs also apply to normal logic programs.

Let us denote the problem of checking whether a disjunctive logic program has a stable model by **DSM**, and the problem of checking whether for a given disjunctive logic program $P$ and propositional formula $\varphi$ in CNF, whether $\varphi$ is a brave (cautious) consequence of $P$ by **BRAV** (**CAUT**). Eiter and Gottlob [34] have determined the complexity of **DSM**, **BRAV**, and **CAUT**: Both **DSM** and **BRAV** are $\Sigma_2^p$-complete, and **CAUT** is $\Pi_2^p$-complete. Moreover, **BRAV** remains $\Sigma_2^p$-complete even if $\varphi$ consists of a single atom and **CAUT** remains $\Pi_2^p$-complete even if $\varphi$ consists of a single negated atom.

For a computable mapping $p$ that assigns a non-negative integer to a disjunctive logic program, or, in the case of **BRAV** and **CAUT**, to a pair consisting of a disjunctive logic program and a formula $\varphi$, we can define in the obvious way the parameterized versions **DSM**$(p)$, **BRAV**$(p)$, and **CAUT**$(p)$ of **DSM**, **BRAV**, and **CAUT**, respectively. For example, **BRAV**$(p)$ has

as instance a disjunctive logic program $P$, a formula $\varphi$ in CNF, and a non-negative integer $k$, such that $p(P, \varphi) \leq k$. The question is whether $P$ admits a stable model satisfying $\varphi$.

Since each disjunctive logic program is syntactically in CNF, the parameters **tw** and **tw**$^*$ are well defined for such programs, and thus for the problem **DSM**. For each instance $(P, \varphi)$ of the problems **BRAV** and **CAUT**, we define **tw**$^*(I)$ as **tw**$^*(P \wedge \varphi)$, and **tw**$(I)$ as **tw**$(P \wedge \varphi)$, where $P \wedge \varphi$ is simply the CNF formula resulting from conjoining the two CNF formulas $P$ and $\varphi$. The following positive result was recently shown:

THEOREM 4.4 ([59]). *The problems* **DSM**(**tw**$^*$), **BRAV**(**tw**$^*$), *and* **CAUT**(**tw**$^*$) *are all fixed-parameter tractable and actually solvable in linear time for fixed parameter* **tw**$^*$.

The proof uses Courcelle's Theorem by formulating the problems **DSM**, **BRAV**, and **CAUT** as $\mathrm{MSO}_1$ sentences which are evaluated over structures of bounded incidence treewidth encoding the instances $P$, and $(P, \varphi)$, respectively. The interesting aspect of these results (and proofs) is that the non-parameterized versions of these problems are complete for classes at the second level of the Polynomial Hierarchy. Thus, when keeping the **tw**$^*$ parameter fixed, we jump down two levels the Polynomial Hierarchy. In this sense, the FPT results are somewhat more drastic than FPT results for NP-complete problems. In fact, we also have a slightly more complex $\mathrm{MSO}_1$ formula which contains an alternation of second-order quantifiers. More FPT results of this sort will be discussed in the next section.

Since, as already noted, bounded **tw** implies bounded **tw**$^*$, the fixed-parameter tractability of **DSM**(**tw**), **BRAV**(**tw**), and **CAUT**(**tw**) trivially follows from Theorem 4.4.

It would be interesting to explore the problems **DSM**, **SM**, **BRAV**, and **CAUT** for yet other structural parameters and to develop a backdoor set approach as described in Sections 2 and 3 above for various base classes.

### 4.4. Closed World Reasoning

Gottlob, Pichler, and Wei [59] also study parameterized versions of a number of other forms of non-monotonic reasoning problems, and show that they are fixed-parameter tractable w.r.t. the parameter treewidth. In this section we give a short overview of these results.

*Closed world reasoning* is an important technique used in AI and database theory. Its simplest form, the *Closed World Assumption (CWA)* [89] is based on the observation that in many situations it is common use to express a data or knowledge base $T$ just by listing "positive knowledge", and assuming that what does not logically follow from $T$ is false.

EXAMPLE 4.3. Consider the data relation STUDENT that stores the students enrolled in a painting class, could be STUDENT= {*emma*, *john*, *leila*, *mary*, *zoe*}. This relation could be logically formalized as a conjunction $T$ of logical atoms: $T \equiv$ *student(emma)* $\wedge$ *student(john)* $\wedge$ *student(leila)* $\wedge$ *student(mary)* $\wedge$ *student(zoe)*. In this example, we have one atom for each data tuple. This formalization is, however, somewhat problematic given that we usually assume that *all* students enrolled in the course are listed in the STUDENT relation. Thus, in the relational database view, STUDENT is a model whose true atoms are *precisely* those listed. For example, we would normally infer from the knowledge present in the STUDENT relation that Irene and Tom are not students of the painting class. However, first-order logic does not allow us to do so. In fact, neither $\neg$*student(irene)* nor $\neg$*student(tom)* is a logical consequence of $T$. □

The idea that led to the CWA was thus to add the negation of all facts (atoms) that are not logical consequences of $T$ to $T$, getting a new theory CWA$(T)$, and then take CWA$(T)$ as the intended logical semantics of the given data relations. Formally we put

$$\mathrm{CWA}(T) = T \cup \{\, \neg K : K \text{ is a positive literal such that } T \not\models K \,\}.$$

For Example 4.3 it now clearly holds that CWA$(T) \models \neg$*student(irene)* and CWA$(T) \models \neg$*student(tom)* (assuming that *irene* and *tom* belong to some suitably defined universe, e.g., of all students of the university). Obviously, reasoning under the CWA is non-monotonic. For example if we add the atom *student(irene)* to $T$, then we can no longer infer $\neg$*student(irene)*.

The CWA interpretation CWA$(T)$ of a logical theory $T$ works well as long as $T$ does not contain disjunctive information. It fails dramatically for theories as simple as $D \equiv a \vee b$, where $a$ and $b$ are propositional atoms. In fact, in this case, we have CWA$(D) \models \neg a$ and CWA$(D) \models \neg b$ and thus CWA$(D)$ is inconsistent. In general, CWA$(T)$ is consistent if and only if $T$ has a unique minimal model $M$, and, in this case, CWA$(T)$ has precisely $M$ as its unique model.

On input $T$, the complexity of reasoning under the CWA, i.e., of checking whether CWA$(T)$ is consistent, is currently an open problem. It is known that this problem is co-NP-hard and can be solved in polynomial time with a logarithmic number of calls to an oracle in NP [32]. Several more sophisticated closed world reasoning mechanisms that redress the consistency problem of CWA at least partially have been designed. Examples for such mechanisms are *Generalized CWA (GCWA)* [80], *Extended GCWA (EGCWA)* [108], *Careful CWA (CCWA)* [50]. and *Extended CWA (ECWA)*, also known as *propositional circumscription* [51, 79]. For definitions of these mechanisms we refer to [32]. Note that, just like the

CWA, all these reasoning methods are non-monotonic. For $C \in \{$ CWA, GCWA, EGCWA, CCWA, ECWA $\}$, deciding whether $C(T) \models \varphi$ holds is $\Pi_2^p$-complete or harder [32].

For $C \in \{$ CWA, GCWA, EGCWA, CCWA, ECWA $\}$ we consider the following parameterized problems.

> $C$-**Deduction**($\mathbf{tw}^*$)
> *Instance:* Propositional CNF formulas $T$ and $\varphi$, and a non-negative integer $k$ with $\mathrm{tw}^*(T \wedge \varphi) \leq k$.
> *Parameter:* $k$.
> *Question:* Does $C(T) \models \varphi$ hold?

By means of Tseitin's transformation (see Example 3.1) this problem generalizes in a natural way to instances where $T$ and $\varphi$ are not in CNF.

Gottlob et al. [59] note that the deduction problem for all these forms of non-monotonic reasoning can be encoded in terms of MSO sentences over a combined structure encoding both the theory $T$ and the formula $\varphi$. We thus have the following result.

THEOREM 4.5. *For each of the CWA reasoning mechanisms $C \in \{$ CWA, GCWA, EGCWA, CCWA, ECWA $\}$, the problem $C$-***Deduction**($\mathbf{tw}^*$) *is fixed-parameter tractable and actually linear time solvable for constant* $\mathbf{tw}^*$.

### 4.5. Propositional Abduction

Another form of non-monotonic reasoning is *abduction*. Abduction is a kind of "reverse reasoning" where one seeks possible implicants of a manifestation or, in other terms, for causes of a symptom.

In the following we will consider a simple form of plain propositional abduction. For the abduction problems below, an instance is given by a tuple $\mathcal{P} = (V, H, M, T)$ where $V$ is a set of propositional variables, $H$ is a subset of $V$ (the "hypotheses"), $M$ is a subset of $V$ (the "manifestations"), and $T$ is a consistent propositional formula (the "theory"). A *solution* of $\mathcal{P}$ is a subset $S \subseteq H$, such that $T \cup S$ is consistent and $T \cup S \models M$. Given an instance $\mathcal{P}$, the basic problems of propositional abduction are the following:

- **Solvability**: Does there exist a solution of $\mathcal{P}$?
- **Relevance**: Given $h \in H$, is $h$ contained in at least one solution of $\mathcal{P}$?
- **Necessity**: Given $h \in H$, is $h$ contained in every solution of $\mathcal{P}$?

Often, a refined version of abduction is used, where one is not interested in all solutions, but only in solutions that are minimal w.r.t. to some preorder $\preceq$ on the powerset $2^H$. In this case, the problems **Relevance** and **Necessity** are adapted accordingly, and we speak of $\preceq$-**Relevance** (respectively, $\preceq$-**Necessity**). Plain abduction as above corresponds to the special case where $\preceq$ is equality. The following preorders have also been considered.

- Subset-minimality "$\subseteq$."
- Prioritization "$\subseteq_P$" for a fixed number $p$ of priorities: $H$ is partitioned into "priorities" $H_1, \ldots, H_p$. Then $A \subseteq_P B$, if and only if $A = B$ or there exists a $k$ s.t. $A \cap H_i = B \cap H_i$ for all $i < k$ and $A \cap H_k \subset B \cap H_k$.
- Minimum cardinality "$\leq$": $A \leq B$ if and only if $|A| \leq |B|$.
- Penalization "$\sqsubseteq_p$" (also referred to as "weighted abduction"): To each element $h \in H$, a weight $w(h)$ is attached. Then $A \sqsubseteq_p B$ if and only if $\sum_{h \in A} w(h) \leq \sum_{h \in B} w(h)$.

EXAMPLE 4.4 ([33], inspired by [22]). Consider the following theory $T$, set of hypotheses $H$, and set of manifestations $M$.

$$\begin{aligned} T &= \{\, \neg(\mathit{rich\_mixture} \wedge \mathit{lean\_mixture}), \\ &\quad\ \mathit{rich\_mixture} \rightarrow \mathit{high\_fuel\_consumption}, \\ &\quad\ \mathit{lean\_mixture} \rightarrow \mathit{overheating}, \\ &\quad\ \mathit{low\_oil} \rightarrow \mathit{overheating}, \\ &\quad\ \mathit{low\_water} \rightarrow \mathit{overheating} \}, \\ H &= \{\, \mathit{rich\_mixture}, \mathit{lean\_mixture}, \\ &\quad\ \mathit{low\_oil}, \mathit{low\_water} \}, \\ M &= \{\, \mathit{high\_fuel\_consumption}, \mathit{overheating} \}. \end{aligned}$$

Then, {*rich_mixture*, *low_oil*} and {*rich_mixture*, *low_water*} are solutions w.r.t. the preorders $=$, $\subseteq$, and $\leq$, but the set of hypotheses {*rich_mixture*, *low_oil*, *low_water*} is a solution only for the preorder $=$. Note that {*rich_mixture*, *lean_mixture*} is ruled out as a solution, since it is inconsistent with $T$. □

It is easy to see that logic-based abduction is a form of non-monotonic reasoning. For instance, adding the formula $\neg \mathit{low\_oil}$ to the theory $T$ of the above example has the effect that {*rich_mixture*, *low_oil*} is no longer a solution. Thus, from stronger theories we may sometimes abduce less.

In different contexts, different preorders may be the most appropriate. A detailed discussion can be found in the paper of Eiter and Gottlob [33]. The complexity of $\preceq$-**Relevance** and $\preceq$-**Necessity** for all these relations (including $=$) was studied by Eiter and Gottlob [33] who show that these problems all lie on the second and third level of the Polynomial Hierarchy. Gottlob et al. [59] consider parameterized versions of the above propositional abduction problems. For instances $(V, H, M, T)$ where $T$ is a propositional formula in CNF, the parameter is the bound on the treewidth of the incidence graph of $T$.

THEOREM 4.6 ([59]). *Each of the following problems is fixed-parameter tractable:* **Solvability**($\mathbf{tw}^*$), **Relevance**($\mathbf{tw}^*$), *and* **Necessity**($\mathbf{tw}^*$), *as well as* $\preceq$-**Relevance**($\mathbf{tw}^*$) *and* $\preceq$-**Necessity**($\mathbf{tw}^*$) *with* $\preceq \in \{=, \subseteq, \subseteq_P, \leq, \sqsubseteq_p\}$. *Moreover, if* $\preceq \in \{=, \subseteq, \subseteq_P\}$, *then the problems can be solved in linear time for constant* $\mathbf{tw}^*$.

The proofs for preorders $\preceq \in \{=, \subseteq, \subseteq_P\}$ use Courcelle's Theorem. The theory $T$ is encoded in a finite structure of the same incidence treewidth as $T$, and the $\preceq$-**Relevance** and $\preceq$-**Necessity** problems are formulated as an MSO sentence with second-order quantifier alternations. However, for the preorders $\leq$ and $\sqsubseteq_p$, such a MSO-formalization is not possible. In fact, set cardinality comparisons are known not to be expressible in MSO. Therefore the proof of Theorem 4.6 for the preorders $\leq$ and $\sqsubseteq_p$ used a result by Arnborg, Lagergreen, and Seese [6] stating that the evaluation of formulas that extend MSO with weight functions, sums and minimum/maximum-functions is fixed-parameter tractable (but not necessarily linear) with respect to the parameter treewidth.

## 5. CORES AND DATA EXCHANGE

In this section we deal with the problem of computing cores of graphs, and, more generally, of relational structures. Roughly, a core of a relational structure $A$ is the smallest relational structure contained in $A$ that is homomorphically equivalent to $A$. This core is unique up to isomorphism. Cores were considered in graph theory [67], and more recently in the context of *data exchange* [37, 38, 52, 58], where methods are studied for transferring data from one database to another database having a different structure.

If $G = (V, E)$ is a graph, then an endomorphism of $G$ is a mapping $h : V \to V$ such that for each edge $xy \in E$, also $h(x)h(y) \in E$. A subgraph $G'$ of a graph $G$ is called a *core* of $G$ if $G'$ is a minimal endomorphic image of $G$ (see [67]) . For example, if a graph $G$ is 3-colorable and, in addition, contains a triangle, then this triangle is a core of $G$. Since all cores of a graph are isomorphic, we usually speak about *the core* of a graph. It is well known that computing cores of graphs is NP-hard [67].

The concept of core can be extended as follows to finite relational *instances* with constants and variables, i.e., finite structures whose universe elements are of two sorts: Constants and variables. Let $I$ be an instances over the universe $U = C \cup V$, where $C$ are the constants and $V$ are the variables. Then an endomorphism is a mapping $h : U \to U$ such that the following hold.

(i) $h$ preserves constants, i.e., $h(c) = c$ holds for all $c \in C$, and
(ii) for each relation $R$ of the structure, and for each tuple $(x_1, \ldots, x_k) \in R$, we have $(h(x_1), \ldots, h(x_k)) \in R$.

Relational instances with variables arise when dealing with unknown data in form of *marked null values*. Such marked null values are to be considered existentially quantified. If we identify the tuples of a relational database with logical atoms, then the logical semantics of a database $D$ with variables is given by a prenex formula $\varphi_D$ whose quantifier prefix existentially quantifies over all variables of $D$ and whose matrix consists of the conjunction of all atoms of $D$.

EXAMPLE 5.1. Suppose we know that the employees Julia and Sue have the same phone number, but we do not know which one. We can use a specific variable (or marked null), say $x_1$, to represent this phone number in a relational instance, and write $x_1$ both in the field of Sue's phone and in the field of Julia's phone. Consider the database $D = \{phone(sue,x_1),\ phone(bill,x_2),\ phone(julia,x_3),\ phone(bill,73859)\}$ and the formula $\varphi_D = (\exists x_1\, \exists x_2)(phone(sue, x_1) \wedge phone(bill, x_2) \wedge phone(julia, x_1) \wedge phone(bill, 73859))$. The database $D$ is obviously redundant, as the tuple $phone(bill,x_2)$ can be eliminated from $D$ without changing the semantics. By eliminating this tuple we actually obtain the core of $D$. □

More generally, it is very easy to see and well known (cf. [38]) that the core of a finite relational instance $I$ is (up to isomorphism) the smallest sub-instance $J$ of $I$ which is logically equivalent to $I$, i.e., such that $\varphi_J \equiv \varphi_I$.

Let us consider the following two problems:

(i) **GraphCoreIdentification** (**GCI**): Given a graph $G = (V, E)$, and a subgraph $G'$ of $G$, decide whether $G'$ is the core of $G$.
(ii) **InstanceCoreIdentification** (**ICI**): Given an instance $I$ and a sub-instance $J$ of $I$ (i.e. an instance whose relations consist of subsets of those of $I$), decide whether $J$ is the core of $I$.

Note that **ICI** is at least as hard as **GCI**, since for every instance $G$ of **GCI** we can trivially obtain an equivalent instance of **ICI** by simply identifying the set $V$ of vertices of $G$ with a set of variables; we then get an instance of **ICI** whose set of constraints is empty.

Gottlob and Fermüller [53] have shown that the Clause Condensation Problem, which is immediately seen to be equivalent to **ICI**, is DP-complete (see Section 3.4 for more information on DP). More recently, Fagin, Kolaitis, and Popa [38] showed by a somewhat more involved proof, that also **GCI** is DP-complete.

The Gaifman graph of an instance $I$ of **ICI** has as vertices the variables of $I$ and an edge between two variables $x$ and $y$ if $x$ and $y$ appear in a same tuple of $I$. The *blockwidth* $\mathbf{blw}(I)$ of $I$ is the size of the largest connected component of the Gaifman graph of $I$. The parameterized problem **ICI**(**blw**) is defined in the obvious way.

THEOREM 5.1 ([52]). *The problem* **ICI**(**blw**) *is* W[1]*-hard.*

This problem is most likely not in any class W[t] of the W-hierarchy because **ICI** is DP-hard and thus also co-NP-hard.

*Data exchange* aims at materializing in a target database data stemming from some source database. While data exchange has been recognized as an

important problem for several decades, systematic research on foundational and algorithmic issues of this problem has started only a few years ago with the fundamental work of Fagin, Kolaitis, Miller, and Popa [37]. The basic and most fundamental data exchange problem for relational databases, as defined by Fagin et al. [37], is as follows. Given a source database schema $\sigma$, a target database schema $\tau$, a source database instance $S$, and a set of constraints $\Sigma$, find a target database instance $T$ such that $S$ and $T$ satisfy all constraints in $\Sigma$, denoted by $(S,T) \models \Sigma$. Fagin et al. [37, 38] restrict their attention to the following types of constraints

(i) *Tuple generating dependencies (TGDs)* which are first-order implications of the form $\forall \bar{u}\,(\varphi(\bar{u}) \implies \exists \bar{v}\,\psi(\bar{u},\bar{v}))$, where $\varphi$ and $\psi$ are conjunctions of atoms and $\bar{u}$ and $\bar{v}$ are lists of variables. If $\bar{v}$ is empty, then we speak about a *full TGD.*

(ii) *Equation generating dependencies (EGDs)* which are first-order implications of the form $\forall \bar{u}\,(\varphi(\bar{u}) \implies v = w)$, where $\varphi$ is a conjunction of atoms, $\bar{u}$ is a list of variables, and $v$ and $w$ are single variables from the list $\bar{u}$.

*Source-to-target constraints* are those where (i) the premise $\varphi(\bar{u})$ contains atoms whose predicate symbols are relation names of the source signature $\sigma$ only, and where (ii) the conclusion $\psi(\bar{u},\bar{v})$ is made of atoms whose predicate symbols are relation names of the target signature $\tau$ only. All atoms occurring in target constraints refer to the target signature only.

In particular, Fagin et al. [37, 38] consider the case where the set $\Sigma = \Sigma_{st} \cup \Sigma_t$ of constraints consists of source-to-target constraints $\Sigma_{st}$ encoding conditions on the mapping between source and target data, and the target constraints $\Sigma_t$ which express data dependencies on the target database.

This setting allows us to formulate a large class of constraints.

EXAMPLE 5.2. Let a source database contain a relation

*student*(STUDNAME,BIRTHDATE,SSN,ZIPCODE)

and let the target database contain two relations

*person*(NAME,BORN,SSN,ZIPCODE,PHONE),

*zc*(ZIPCODE,STATE),

then a typical source-to-target constraint would be

$$st1:\ (\forall u_1,u_2,u_3,u_4)\,(\mathit{student}(u_1,u_2,u_3,u_4) \implies (\exists v)\,\mathit{person}(u_1,u_2,u_3,u_4,v)),$$

while the target constraints could be

$$t1:\ (\forall u_1,u_2,u_3,u_4,u_5)\,(\mathit{person}(u_1,u_2,u_3,u_4,u_5) \implies (\exists v)\,\mathit{zc}(u_5,v))$$

$$t2:\ (\forall u,v)\,(\mathit{zc}(u,v) \wedge \mathit{zc}(u,w) \implies v = w).$$

Here $st_1$ and $t_1$ are inclusion dependencies, and $t_2$ is a functional dependency. Note that $t1$ and $t2$ together express a "foreign key" constraint. □

It is easy to see that, in addition to functional, inclusion dependencies, and foreign key constraints, also multivalued dependencies and even join dependencies can be expressed by formulas in this setting. Thus, this setting is very general and encompasses all major dependencies used in database design and for database maintenance.

The problem is to check whether a target instance $T$ exists, and if so, to compute a good one. Variables (or, equivalently, labeled null values) are allowed to appear in target instances. In Example 5.2, a tuple $(\mathit{doe}, 19880203, 1234567, 94305)$ of the *student* source relation could be translated into a tuple $(\mathit{doe}, 19880203, 1234567, 94305, x_1)$ of the *person* relation, where $x_1$ is a variable representing a null value. For further, more detailed examples, we refer to the paper of Fagin et al. [37].

A *universal solution* of a data exchange problem is a target instance $T$ which is more general than all other solutions, i.e., such that for each other solution $T'$ there exists a homomorphism $T \to T'$. Fagin, Kolaitis, Miller, and Popa [37] have shown that universal solutions of data exchange problems can be obtained via the well-known *chase* procedure [9, 3, 78]. One first chases the set $\Sigma_{st}$ of source-to-target TGDs over the source instance $S$ and obtains in polynomial time an initial target instance $T = S^{\Sigma_{st}}$. Then one chases the target constraints $\Sigma_t$ over $T$. If this chase terminates, one obtains a finite universal solution. In order to guarantee termination, Fagin et al. [37, 38] restrict themselves to target TGDs which are *weakly acyclic*. Weak acyclicity is a syntactic condition on TGDs ensuring that there are no cyclic dependencies among argument positions involving existential constraints (see [38] for a precise definition). Weak acyclicity has been so far the most general known sufficient condition for termination of the chase. This concept was developed by Deutsch [28] and Popa [37]. Fagin and his colleagues [37] have shown that universal solutions are very useful for query answering. In particular, any universal solution can be used to obtain the *certain answers tuples* to a conjunctive query over the target schema, i.e., those answer tuples that are contained in *all* solutions of the data exchange problem.

If a data exchange problem $P$ is solvable, then it can have several universal solutions, and these solutions can noticeably differ in size. However, as observed in [38], the cores of all universal solutions to a data exchange problem are all mutually isomorphic, and thus there is, up to isomorphism, one single smallest solution, which is, up to isomorphism, equal to the core of each universal solution. To compute this smallest solution, one can first compute an arbitrary universal solution $J$ and then compute the core of $J$. Fagin, Kolaitis, and Popa [38] strongly argue that this should be the solution of choice.

Fagin et al. [38] formulated the following problem: Given a (solvable) data exchange problem whose source-to-target constraints are TGDs and whose target constraints consist of weakly-acyclic TGDs and

arbitrary EGDs, can the core of a universal solution be computed in polynomial time? Here the source schema $\sigma$, the target schema $\tau$ and the set of constraints $\Sigma$ are all considered to be fixed, and the problem instance thus consists of a source instance $S$.

Unfortunately, even though (in case of solvable data exchange problems) a universal solution can be computed in polynomial time via the chase procedure, the solutions can be of unbounded block width. In fact, even very simple TGDs such as $(\forall x, y)\,((P(x) \wedge P(y)) \rightarrow P(x, y))$ can "lump" variables from different components (blocks) together and thus merge several small components into very large components of connected variables.

Thus, at a first glance, Theorem 5.1 may suggest a negative answer to the above problem raised by Fagin, Kolaitis, and Popa. However, it turned out that it has a positive answer.

THEOREM 5.2 ([58]). *The core of a universal solution of a solvable data exchange problem whose source-to-target constraints are TGDs and whose target constraints consist of weakly-acyclic TGDs and arbitrary EGDs can be computed in polynomial time.*

The proposed solution to the problem is technically rather involved. The reader is referred to the conference paper [58] for further details.


### Acknowledgment

We thank Mike Fellows for valuable comments on a preliminary version of the paper and Marko Samer for careful proof reading. Georg Gottlob's work was supported by a Royal Society Wolfson Research Merit Award. Stefan Szeider's work was supported by the EPSRC, Project EP/E001394/1 Fixed-parameter algorithms and satisfiability.


## REFERENCES


[1] I. Adler, G. Gottlob, and M. Grohe. Hypertree-width and related hypergraph invariants. In *Proceedings of the 3rd European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB'05)*, DMTCS Proceedings Series, volume AE, pages 5–10, 2005. Full version to appear in *European Journal of Combinatorics*.

[2] R. Aharoni and N. Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Combin. Theory Ser. A*, 43:196–204, 1986.

[3] A. Aho, C. Beeri, and J. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, 1979.

[4] M. Alekhnovich and A. A. Razborov. Satisfiability, branch-width and Tseitin tautologies. In *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02), Vancouver, BC, Canada, 16-19 November 2002*, pages 593–603, 2002.

[5] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a $k$-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[6] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.

[7] F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03), Cambridge, MA, USA, 11-14 October 2003*, pages 340–351, 2003.

[8] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.

[9] C. Beeri and M. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[10] P. Bjesse, T. Leonard, and A. Mokkedem. Finding bugs in an alpha microprocessor using satisfiability solvers. In G. Berry, H. Comon, and A. Finkel, editors, *Computer Aided Verification: 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*, pages 454–464, 2001.

[11] H. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica* 11(1-2):1–22, 1993.

[12] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Automata, languages and programming (Tampere, 1988)*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118. Springer Verlag, 1988.

[13] H. L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993.

[14] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.

[15] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[16] H. L. Bodlaender. Discovering treewidth. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, volume 3381 of *Lecture Notes in Computer Science*, pages 1–16. Springer Verlag, 2005.

[17] H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In P. Beek, editor, *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP 2005), Sitges, Spain, October 1-5, 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 167–181. Springer Verlag, 2005.

[18] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[19] J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for vertex cover. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS 2006), Stará Lesná, Slovakia, August 28-September 1, 2006*, pages 238–249, 2006.

[20] F. R. K. Chung, R. L. Graham, P. Frankl, and J. B. Shearer. Some intersection theorems for ordered sets and graphs. *J. Combin. Theory Ser. A*, 43(1):23–37, 1986.

[21] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, volume I, chapter 8. Elsevier, 2006.

[22] L. Console, D. T. Dupré, and P. Torasso. On the relationship between abduction and deduction. *J. Log. Comput.*, 1(5):661–690, 1991.

[23] B. Courcelle. Recognizability and second-order definability for sets of finite graphs. Technical Report I-8634, Université de Bordeaux, 1987.

[24] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discr. Appl. Math.*, 108(1-2):23–52, 2001.

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

[26] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[27] R. Dechter. Constraint networks. In *Encyclopedia of Artificial Intelligence*, volume 1, pages 276–285. John Wiley & Sons, 2nd edition, 1992.

[28] A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *Proc. of the 9th International Conference on Database Theory (ICDT 2003), Siena, Italy, January 8-10, 2003*, Lecture Notes in Computer Science, volume 2572, pages 225–241. Springer Verlag, 2003.

[29] R. G. Downey. Parameterized complexity for the skeptic. In *Proc. 18th IEEE Conf. on Computational Complexity 2003, Aarhus, Denmark, 7-10 July 2003*, pages 147–168, 2003. invited paper.

[30] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer Verlag, 1999.

[31] R. G. Downey, M. R. Fellows, and K. W. Regan. Descriptive complexity and the $W$ hierarchy. In *Proof complexity and feasible arithmetics (Rutgers, NJ, 1996)*, volume 39 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 119–134. Amer. Math. Soc., Providence, RI, 1998.

[32] T. Eiter and G. Gottlob. Propositional circumscription and extended closed-world reasoning are $\Pi_2^P$-complete. *Theoret. Comput. Sci.*, 114(2):231–245, 1993.

[33] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.

[34] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.*, 15(3-4):289–323, 1995.

[35] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-Time extremal structure I. In H. Broersma, M. Johnson, and S. Szeider, editors, *Algorithms and Complexity in Durham 2005, Proceedings of the first ACiD Workshop*, volume 4 of *Texts in Algorithmics*, pages 1–41. King's College Publications, 2005.

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

[37] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In *Proc. of the 9th International Conference on Database Theory (ICDT 2003), Siena, Italy, January 8-10, 2003*, Lecture Notes in Computer Science, volume 2572, pages 207–224. Springer Verlag. Full version in: *Theor. Comput. Sci.* 336(1): 89-124 (2005).

[38] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. In *Proc. of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2003), June 9-12, 2003, San Diego, CA, USA*, pages 90–101, 2003. Full version in ACM TODS, 30(1):147-210(2005).

[39] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999.

[40] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width minimization is NP-hard. In *Proceedings of STOC 2006; the 38th ACM Symposium on Theory of Computing, Seattle, Washington, USA*, pages 354–362. ACM, 2006.

[41] H. Fernau. Parameterized algorithmics: A graph-theoretic approach. Technical report, Universtät Tübingen, 2005. Habilitation thesis.

[42] E. Fischer, J. A. Makowsky, and E. R. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.*, in press.

[43] H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoret. Comput. Sci.*, 289(1):503–516, 2002.

[44] J. Flum and M. Grohe. Model-checking problems as a basis for parameterized intractability. *Logical Methods in Computer Science*, 1(1:2):1–36, 2005.

[45] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer Verlag, 2006.

[46] J. Franco, J. Goldsmith, J. Schlipf, E. Speckenmeyer, and R. P. Swaminathan. An algorithm for the class of pure implicational formulas. *Discr. Appl. Math.*, 96/97:89–106, 1999.

[47] J. Franco and A. Van Gelder. A perspective on certain polynomial time solvable classes of satisfiability. *Discr. Appl. Math.*, 125:177–214, 2003.

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

[49] M. R. Garey and D. R. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, 1979.

[50] M. Gelfond and H. Przymusinska. Negation as failure: Careful closure procedure. *Artif. Intell.*, 30(3):273–287, 1986.

[51] M. Gelfond, H. Przymusinska, and T. C. Przymusinski. The extended closed world assumption and its relationship to parallel circumscription. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 24-26, 1986, Cambridge, Massachusetts, PODS'86*, pages 133–139, 1986.

[52] G. Gottlob. Computing cores for data exchange: New algoritms and practical solutions. In *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2005), June 13-15, 2005, Baltimore, Maryland, USA*, pages 148–159, 2005.

[53] G. Gottlob and C. G. Fermüller. Removing redundancy from a clause. *Artif. Intell.*, 61(2):263–289, 1993.

[54] G. Gottlob, M. Grohe, N. Musliu, M. Samer, and F. Scarcello. Hypertree decompositions: Structure, algorithms, and applications. In D. Kratsch, editor, *Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG'05)*, volume 3787 of *Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, 2005.

[55] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.

[56] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions: a survey. In *Mathematical foundations of computer science, 2001 (Mariánské Lázně)*, volume 2136 of *Lecture Notes in Computer Science*, pages 37–57. Springer Verlag, 2001.

[57] G. Gottlob, Z. Miklos, and T. Schwentick. Generalized hypertree decompositions: NP-hardness and tractable variants. In *Proc. ACM Symp on Theory of Database Systems, (PODS 2007), 11-14 June 2007, Beijing, China*, pages 13-22, ACM, 2007.

[58] G. Gottlob and A. Nash. Data exchange: Computing cores in polynomial time. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2006), June 26-28, 2006, Chicago, Illinois, Maryland, USA.*, pages 40–49, 2006.

[59] G. Gottlob, R. Pichler, and F. Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. In *21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, Boston, Massachusetts, USA, July 16-20, 2006*. AAAI Press, pages 250–256, 2006.

[60] G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002.

[61] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), pages 1–24, 2007.

[62] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *Proceedings of the of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), Miami, Florida, USA, January 22-26, 2006*, pages 289–298. ACM Press, 2006.

[63] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 657–666, New York, 2001. ACM.

[64] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Improved fixed-parameter algorithms for two feedback set problems. In F. K. H. A. Dehne, A. López-Ortiz, and J.-R. Sack, editors, *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 158–168. Springer Verlag, 2005.

[65] G. Gutin, A. Rafiey, S. Szeider, and A. Yeo. The linear arrangement problem parameterized above guaranteed value. *Theory Comput. Syst.*, in press.

[66] G. Gutin, A. Yeo, and S. Szeider. Fixed-parameter complexity of minimum profile problems. In *Proceedings of IWPEC 2006, 2nd International Workshop on Parameterized and Exact Computation*, volume 4169 of *Lecture Notes in Computer Science*, pages 60–71. Springer Verlag, 2006.

[67] P. Hell and J. Nešetřil. The core of a graph. *Discrete Math.*, 109(1-3):117–126, 1992.

[68] P. Heusch, S. Porschen, and E. Speckenmeyer. Improving a fixed parameter tractability time bound for the shadow problem. *J. of Computer and System Sciences*, 67(4):772–788, 2003.

[69] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity. *J. of Computer and System Sciences*, 63(4):512–530, 2001.

[70] K. Iwama. CNF-satisfiability test by counting and polynomial average time. *SIAM J. Comput.*, 18(2):385–391, 1989.

[71] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proceedings ECAI 1992, Vienna, Austria, August 3-7, 1992*, pages 359–363, 1992.

[72] H. Kleine Büning and T. Lettman. *Propositional logic: deduction and algorithms*. Cambridge University Press, Cambridge, 1999.

[73] H. Kleine Büning and X. Zhao. Satisfiable formulas closed under replacement. In H. Kautz and B. Selman, editors, *Proceedings for the Workshop on Theory and Applications of Satisfiability*, volume 9 of *Electronic Notes in Discrete Mathematics*. Elsevier Science Publishers, North-Holland, 2001.

[74] P. G. Kolaitis and M. Y. Vardi. 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).

[75] Z. Lonc and M. Truszczyński. Fixed-parameter complexity of semantics for logic programs. *ACM Trans. Comput. Log.*, 4(1):91–119, 2003.

[76] A. G. M. Prasad, A. Biere. A survey of recent advances in sat-based formal verification. *Software Tools for Technology Transfer*, 7(2):156–173, 2005.

[77] M. Mahajan, V. Raman, and S. Sikdar. Parameterizing MAXSNP problems above guaranteed values. In *Proceedings of 2nd International Workshop on Parameterized and Exact Computation IWPEC 2006*, volume 4169 of *Lecture Notes in Computer Science*, pages 38–49. Springer Verlag, 2006.

[78] D. Maier, A. Mendelzon, and J. Sagiv. Testing implication of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.

[79] J. L. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artif. Intell.*, 13(1-2):27–39, 1980.

[80] J. Minker. On indefinite databases and the closed world assumption. In *6th Conference on Automated Deduction, New York, USA, June 7-9, 1982, Proceedings*, pages 292–308, 1982.

[81] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[82] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada), informal proceedings*, pages 96–103, 2004.

[83] N. Nishimura, P. Ragde, and S. Szeider. Solving #SAT using vertex covers. In *Proceedings of SAT 2006, Ninth International Conference on Theory and Applications of Satisfiability Testing, August 12-15, 2006, Seattle, Washington, USA*, volume 4121 of *Lecture Notes in Computer Science*, pages 396–409. Springer Verlag, 2006. Full version to appear in *Acta Informatica*.

[84] S. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

[85] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[86] C. H. Papadimitriou and D. Wolfe. The complexity of facets resolved. *J. of Computer and System Sciences*, 37(1):2–13, 1988.

[87] C. H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *J. of Computer and System Sciences*, 58(3):407–427, 1999.

[88] T. C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Comput.*, 9(3/4):401–424, 1991.

[89] R. Reiter. On closed world data bases. In: H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.

[90] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B*, 41(1):92–114, 1986.

[91] N. Robertson and P. D. Seymour. Graph minors X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.

[92] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995.

[93] M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. In *Proceedings of CP 2006 , Twelfth International Conference on Principles and Practice of Constraint Programming, September 24-29, 2006, Nantes, France*, volume 4204 of *Lecture Notes in Computer Science*, pages 499–513. Springer Verlag, 2006.

[94] M. Samer and S. Szeider. Algorithms for propositional model counting. In *Proceedings of LPAR 2007, 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Yerevan, Armenia, October 15-19, 2007*, Lecture Notes in Computer Science, Springer-Verlag, 2007. To appear.

[95] M. Samer and S. Szeider. Backdoor sets of quantified boolean formulas. In J. Marques-Silva and K. A. Sakallah, editors, *Proceedings of SAT 2007, Tenth International Conference on Theory and Applications of Satisfiability Testing, May 28-31, 2007, Lisbon, Portugal,*, volume 4501 of *Lecture Notes in Computer Science*, pages 230–243. Springer Verlag, 2007.

[96] T. J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, 1978.

[97] J. Siekmann and G. Wrightson, editors. *Automation of reasoning. Classical Papers on Computer Science 1967–1970*, volume 2. Springer Verlag, 1983.

[98] H. Simonis. Sudoku as a constraint problem. In B. Hnich, P. Prosser, and B. Smith, editors, *Modelling and Reformulating Constraint Satisfaction Problems, Fourth International Workshop, Sitges (Barcelona), Spain, 1 October 2005, Proceedings*, pages 13–27, 2005.

[99] S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *J. of Computer and System Sciences*, 69(4):656–674, 2004.

[100] S. Szeider. On fixed-parameter tractable parameterizations of SAT. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability, 6th International Conference, SAT 2003, Selected and Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 188–202. Springer Verlag, 2004.

[101] S. Szeider. Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning*, 35(1-3):73–88, 2005. Reprinted as Chapter 4 of the book *SAT 2005 - Satisfiability Research in the Year 2005*, edited by E. Giunchiglia and T. Walsh, Springer Verlag, 2006.

[102] S. Szeider. Matched formulas and backdoor sets. In J. Marques-Silva and K. A. Sakallah, editors, *Proceedings of SAT 2007, Tenth International Conference on Theory and Applications of Satisfiability Testing, May 28-31, 2007, Lisbon, Portugal,*, volume 4501 of *Lecture Notes in Computer Science*, pages 94–99. Springer Verlag, 2007.

[103] M. Truszczyński. Computing large and small stable models. *Theory Pract. Log. Program.*, 2(1):1–23, 2002.

[104] G. S. Tseitin. Complexity of a derivation in the propositional calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR*, 8:23–41, 1968. English translation reprinted in [97].

[105] M. N. Velev and R. E. Bryant. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. *J. Symbolic Comput.*, 35(2):73–106, 2003.

[106] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, pages 1173–1178. Morgan Kaufmann, 2003.

[107] R. Williams, C. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Sixth International Conference on Theory and Applications of Satisfiability Testing, S. Margherita Ligure - Portofino, Italy, May 5-8, 2003 (SAT 2003), informal proceedings*, pages 222–230, 2003.

[108] A. H. Yahya and L. J. Henschen. Deduction in non-Horn databases. *Journal of Automated Reasoning*, 1(2):141–160, 1985.

[109] M. Yannakakis. Algorithms for acyclic database schemes. In C. Zaniolo and C. Delobel, editors, *Very Large Data Bases, 7th International Conference, Sep. 9–11, 1981, Cannes, France, Proceedings*, pages 81–94. IEEE Computer Society, 1981.