The structure of tractable constraint satisfaction problems

Martin Grohe

Institut für Informatik, Humboldt Universität
Unter den Linden 6, 10099 Berlin, Germany

Abstract We give a survey of recent results on the complexity of constraint satisfaction problems. Our main emphasis is on tractable structural restrictions.

1 Introduction

The objective of a *constraint satisfaction problem (CSP)* is to assign values to variables subject to constraints on the values. Obviously, this is a very general type of problem, and it is not surprising that many algorithmic problems in various areas of computer science can be described as CSPs. It is neither surprising that, in general, CSPs are computationally hard. Considerable efforts have been made to precisely understand the complexity of CSPs, with the goal of identifying tractable restrictions (often referred to as "islands of tractability" in this context) and, ultimately, determining the boundary between tractable and intractable CSPs. There are two main types of restrictions that have been studied: *Constraint language restrictions*, which are concerned with the types of constraints that occur, and *structural restrictions*, which are concerned with the structure induced by the constraints on the variables, for example, with the way the constraints overlap.

1.1 CSP-Instances

An *instance* of a CSP is a triple (V, D, C) consisting of a set V of *variables*, a *domain* D, and a set C of *constraints*. The objective is to find an assignment of values from D to the variables such that all constraints in C are satisfied. The constraints are expressions of the form $Rx_1 \dots x_k$, where R is a k-ary relation on D and $x_1, \dots, x_k$ are variables. A constraint $Rx_1 \dots x_k$ is satisfied if the k-tuple of values assigned to the variables $x_1, \dots, x_k$ belongs to the relation R. The *constraint language* of a CSP-instance (V, D, C) is the set of all relations that occur in the constraints in C.

Example 1. We describe SAT, the satisfiability problem for Boolean formulas in conjunctive normal form (CNF), as a CSP: A CNF-formula ϕ corresponds to a CSP-instance whose variables are the variables of ϕ, whose domain is $\{0, 1\}$, and whose constraints are given by the clauses. For example, the clause $(x \vee \neg y \vee \neg z)$ corresponds to a constraint $Rxyz$, where R is the ternary relation $\{(0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$ on $\{0, 1\}$.

Example 2. As a second example, we consider the 3-Colourability problem for graphs. Its objective is to colour the vertices of a graph with 3 colours in such a way that adjacent vertices get different colours. Each graph $\mathcal{G}$, viewed as an instance of 3-Colourability, corresponds to the following CSP-instance: Variables are the vertices of $\mathcal{G}$, the domain is the set of colours, say, $\{\text{red, blue, green}\}$, and for each edge $\{v, w\}$ of $\mathcal{G}$ there is a constraint Ivw, where I is the disequality relation on the domain.

Example 3. As a third example, consider the Clique problem, whose objective it is to decide whether a graph $\mathcal{G}$ has a clique of size k. An instance $(\mathcal{G}, k)$ of Clique corresponds to the following CSP-instance: The variables are $x_1, \ldots, x_k$, the domain is the vertex set of $\mathcal{G}$, and there are constraints Ex_ix_j for $1 \leq i \leq j$, where E is the edge relation of $\mathcal{G}$.

1.2 Relational structures and homomorphisms

It is a well known observation (going back to Feder and Vardi [16]) that CSPs can be described as homomorphism problems for relational structures (and vice versa). A *(relational) vocabulary* is a finite set τ of relation symbols, each with a prescribed arity. A *structure* $\mathcal{A}$ of vocabulary τ (for short: τ-structure) consists of a finite set A, called the *universe* of $\mathcal{A}$, and a k-ary relation $r^{\mathcal{A}}$ for each k-ary relation symbol $r \in \tau$. A *homomorphism* from a τ-structure $\mathcal{A}$ to a τ-structure $\mathcal{B}$ is a mapping $h : A \to B$ from the universe of $\mathcal{A}$ to the universe of $\mathcal{B}$ that preserves all relations, that is, for all $r \in \tau$, say, of arity k, and all tuples $(a_1, \ldots, a_k) \in r^{\mathcal{A}}$ it holds that $(h(a_1), \ldots, h(a_k)) \in r^{\mathcal{B}}$.

With every CSP-instance $\mathcal{I} = (V, D, \mathcal{C})$ we associate two structures $\mathcal{A}(\mathcal{I})$ and $\mathcal{B}(\mathcal{I})$ as follows: The vocabulary $\tau(\mathcal{I})$ of both structures contains a k-ary relation symbol r for every k-ary relation R in the constraint language of $\mathcal{I}$. The universe of $\mathcal{B}(\mathcal{I})$ is D, and the relations of $\mathcal{B}$ are those appearing in the constraint language. More precisely, for every $r \in \tau(\mathcal{I})$ we let $r^{\mathcal{B}(\mathcal{I})} = R$. The universe of $\mathcal{A}(\mathcal{I})$ is V, and for each k-ary relation symbol $r \in \tau(\mathcal{I})$ we let $r^{\mathcal{A}(\mathcal{I})} = \{(x_1, \ldots, x_k) \mid Rx_1 \ldots x_k \in \mathcal{C}\}$. Then a mapping $h : V \to D$ is a satisfying assignment for the CSP-instance $\mathcal{I}$ if and only if it is a homomorphism from $\mathcal{A}(\mathcal{I})$ to $\mathcal{B}(\mathcal{I})$. Thus $\mathcal{I}$ is satisfiable if and only if there is a homomorphism from $\mathcal{A}(\mathcal{I})$ to $\mathcal{B}(\mathcal{I})$. Conversely, for all pairs of structures $\mathcal{A}$, $\mathcal{B}$ of the same vocabulary, we can easily construct a CSP-instance $\mathcal{I}$ such that $\mathcal{A}(\mathcal{I}) = \mathcal{A}$ and $\mathcal{B}(\mathcal{I}) = \mathcal{B}$.

1.3 Restricted CSPs

The structures $\mathcal{A}(\mathcal{I})$ and $\mathcal{B}(\mathcal{I})$ associated with a CSP-instance $\mathcal{I}$ precisely reflect the two kinds of restrictions on CSPs studied in the literature on the complexity of CSPs: $\mathcal{B}(\mathcal{I})$ is just a fancy representation of the constraint language. Hence restrictions on $\mathcal{B}(\mathcal{I})$ are *constraint language restrictions.* $\mathcal{A}(\mathcal{I})$ is the "structure induced by the constraints on the variables", made precise. Hence restrictions on $\mathcal{A}(\mathcal{I})$ are *structural restrictions.*

In general, for all classes C and D of structures, we have the following restricted CSP:

$\textsc{Csp}(\mathsf{C}, \mathsf{D})$
Instance: CSP-instance $\mathcal{I}$ with $\mathcal{A}(\mathcal{I}) \in \mathsf{C}$ and $\mathcal{B}(\mathcal{I}) \in \mathsf{D}$.
Problem: Decide if $\mathcal{I}$ is satisfiable.

We write $\textsc{Csp}(-, \mathsf{D})$ instead of $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ if C is the class of all structures and $\textsc{Csp}(\mathsf{C}, -)$ if D is the class of all structures. For structures $\mathcal{B}$, we write $\textsc{Csp}(\mathsf{C}, \mathcal{B})$ and $\textsc{Csp}(-, \mathcal{B})$ instead of $\textsc{Csp}(\mathsf{C}, \{\mathcal{B}\})$ and $\textsc{Csp}(-, \{\mathcal{B}\})$. Restrictions of the form $\textsc{Csp}(\mathsf{C}, \mathcal{B})$ are called *nonuniform* [27].

Example 2 revisited: The 3-Colourability problem corresponds to $\textsc{Csp}(\mathsf{G}, \mathcal{D})$, where G denotes the class of all (undirected, loop-free) graphs and $\mathcal{D}$ is a triangle. To see this, note that the disequality relation on a three element domain corresponds to a triangle if viewed as a graph. Since only graphs can be mapped homomorphically to a triangle, $\textsc{Csp}(\mathsf{G}, \mathcal{D})$ is essentially the same problem as $\textsc{Csp}(-, \mathcal{D})$. Thus 3-Colourability is a nonuniform CSP with a restricted constraint language.

Example 3 revisited: The Clique problem corresponds to $\textsc{Csp}(\mathsf{K}, \mathsf{G})$, where K denotes the class of all complete graphs. This problem is essentially the same problem as $\textsc{Csp}(\mathsf{K}, -)$. Thus Clique is a CSP with a restricted structure.

Example 1 revisited: The description of Sat as a problem $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ is slightly more complicated: The *sign pattern* of a clause $\gamma = (\lambda_1 \vee \lambda_2 \vee \ldots \vee \lambda_n)$ is the tuple $\sigma(\gamma) = (s_1, \ldots, s_n)$ of signs, where $s_i = +$ if λ_i is a positive literal (that is, a variable) and $s_i = -$ if λ_i is a negative literal (that is, a negated variable). For example, the sign pattern of the clause $(x \vee \neg y \vee \neg z)$ is $(+, -, -)$. With each n-ary sign pattern σ we associate an n-ary relation R_σ over the Boolean domain that consists of all satisfying assignments for clauses with sign pattern σ. For example, $R_{(+,-,-)} = \{0,1\}^3 \setminus \{(0,1,1)\}$.

Now let $\mathcal{I}$ be the CSP-instance associated with a CNF-formula ϕ. The vocabulary $\tau(\mathcal{I})$ consists of an n-ary relation symbol r_σ for every sign pattern σ that occurs in ϕ. Furthermore, $\mathcal{B}(\mathcal{I})$ is the $\tau(\mathcal{I})$-structure with universe $\{0,1\}$ and relations $r_\sigma^{\mathcal{B}(\mathcal{I})} = R_\sigma$. Let S be the class of all structures $\mathcal{B}$ whose universe is $\{0,1\}$ and whose vocabulary consists of finitely many relation symbols r_σ for sign patterns σ, such that $r_\sigma^{\mathcal{B}} = R_\sigma$. Then Sat corresponds to $\textsc{Csp}(-, \mathsf{S})$.[1]

The examples show that problems $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ are NP-hard in general. Furthermore, both structural restrictions $\textsc{Csp}(\mathsf{C}, -)$, such as Clique, and constraint language restrictions $\textsc{Csp}(-, \mathsf{D})$, such as Sat, can be NP-hard. Even nonuniform constraint language restrictions $\textsc{Csp}(-, \mathcal{B})$, such as 3-Colourability, can be NP-hard. However, observe that "nonuniform structural restrictions" of the form $\textsc{Csp}(\{\mathcal{A}\}, -)$ are always in PTIME, because the set of variables is fixed.

The reader may wonder why we write "NP-hard" instead of "NP-complete" — it seems obvious that all problems $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ are in NP. However, this is not entirely true, because the membership problem for the classes C and D may not be in NP (it may even be undecidable), and in this case it is not even decidable

[1] The astute reader may notice that the translation from Sat to $\textsc{Csp}(-, \mathsf{S})$ involves an exponential blow-up in size. We shall discuss this issue in Sec. 3.1.

in NP if a given CSP-instance $\mathcal{I}$ is an instance of $\textsc{Csp}(\mathsf{C}, \mathsf{D})$. We could avoid this problem by requiring the classes C and D to be polynomial time decidable, but there are interesting examples where they are not. Instead, we view $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ as a promise problem: We say that $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ is *solvable in polynomial time* if there is a polynomial time algorithm that, given an instance $\mathcal{I}$ with $\mathcal{A}(\mathcal{I}) \in \mathsf{C}$ and $\mathcal{B}(\mathcal{I}) \in \mathsf{D}$, correctly decides if $\mathcal{I}$ is solvable. We do not care what the algorithm does if the input is not of this form.

Definition 4. Let C, D be classes of structures. Then $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ is *tractable* if it is solvable in polynomial time (viewed as a promise problem) and *intractable* otherwise. $\textsc{Csp}(\mathsf{C}, \mathsf{D})$ is *hard* if it is NP-hard.

2 Constraint language restrictions

Most and the mathematically deepest work on the complexity of CSPs is concerned with constraint language restrictions (e.g., [7,8,6,4,13,16,22,26,29]). As a matter of fact, most of this work is only concerned with nonuniform constraint language restrictions of the form $\textsc{Csp}(-, \mathcal{B})$. The driving force behind this work is a conjecture that has first been stated by Feder and Vardi in 1993 (in the conference version of [16]):

Conjecture 1 (Dichotomy Conjecture). *For every structure $\mathcal{B}$, the problem $\textsc{Csp}(-, \mathcal{B})$ is either tractable or hard.*

In other words: Every problem of the form $\textsc{Csp}(-, \mathcal{B})$ is either in PTIME or NP-complete (as all such problems are contained in NP). A priori, there is no reason why this should be true, in particular in view of Ladner's theorem [28] stating that if PTIME $\neq$ NP, then there are problems in NP that are neither in PTIME nor NP-complete. Nevertheless, the Dichotomy Conjecture still stands unrefuted, and it has actually been proved in several significant special cases.

Two important special cases of the conjecture had already been proved when Feder and Vardi stated it. In 1978, Schaefer [29] studied what he called "generalised satisfiability problems". In our terminology, these are just CSPs over the Boolean domain $\{0, 1\}$. Let us call a structure $\mathcal{B}$ *Boolean* if its universe is $\{0, 1\}$.

Theorem 5 (Schaefer [29]). *The dichotomy conjecture holds for all Boolean structures. More precisely, for every Boolean structure $\mathcal{B}$, the problem $\textsc{Csp}(-, \mathcal{B})$ is tractable if $\mathcal{B}$ satisfies one of the following conditions:*

(1) Every relation of $\mathcal{B}$ contains a tuple in which all entries are 0.
(2) Every relation of $\mathcal{B}$ contains a tuple in which all entries are 1.
(3) Every relation of $\mathcal{B}$ is definable by a CNF-formula in which every clause contains at most one negative literal.
(4) Every relation of $\mathcal{B}$ is definable by a CNF-formula in which every clause contains at most one positive literal.
(5) Every relation of $\mathcal{B}$ is definable by a CNF-formula in which every clause contains at most two literals.

(6) Every relation of $\mathcal{B}$ is the set of solutions of a system of linear equations over the two element field $\mathrm{GF}(2)$.

Otherwise, $\textsc{Csp}(-, \mathcal{B})$ *is hard.*

Obviously, Schaefer's theorem implies the Dichotomy conjecture for all two-element structures $\mathcal{B}$. It is not hard to see that Schaefer's theorem also implies a dichotomy for all classes of Boolean structures: For every class B of Boolean structures, $\textsc{Csp}(-, \mathsf{B})$ is tractable if every structure $\mathcal{B} \in \mathsf{B}$ satisfies one of the conditions (1)–(6) and hard otherwise.

Note that $\textsc{Csp}(-, \mathcal{B})$ is tractable for all one-element structures $\mathcal{B}$. Hence, for the rest of this section we assume that all structures have at least two elements.

Hell and Nešetřil [22] studied the complexity of the graph homomorphism problem for a fixed target graph $\mathcal{H}$ (the so called *$\mathcal{H}$-colouring problem*) and proved a dichotomy result. In our terminology, their result reads as follows:

Theorem 6 (Hell and Nešetřil [22]). *The dichotomy conjecture holds for graphs. More precisely, for every graph $\mathcal{H}$, the problem* $\textsc{Csp}(-, \mathcal{H})$ *is tractable if $\mathcal{H}$ is bipartite; otherwise, it is hard.*

Remember that we assume graphs to be undirected an loop-free. The dichotomy can easily be extend to graphs with loops, because $\textsc{Csp}(-, \mathcal{H})$ is tractable for every $\mathcal{H}$ that has a loop. Hell and Nešetřil's theorem also implies a dichotomy for classes of graphs: For every class H of graphs, $\textsc{Csp}(-, \mathsf{H})$ is tractable if every $\mathcal{H} \in \mathsf{H}$ is bipartite and hard otherwise. For directed graphs, the situation is much more complicated: Feder and Vardi [16] proved that the dichotomy conjecture for directed acyclic graphs is equivalent to the general dichotomy conjecture. Even for oriented trees, the conjecture is still open (cf. [23,24]).

In recent years, Bulatov, Cohen, Dalmau, Jeavons, Krokhin and others very successfully pursued an algebraic approach to the complexity of CSPs. While still short of proving the dichotomy conjecture, they made some remarkable progress. We cannot hope to explain the approach in any depth in this short survey. For detailed presentations of the algebraic approach, we refer the reader to [8,11,26]. Our modest goal for the rest of this section is to state the main results obtained by this approach and the currently conjectured dividing line between tractable and hard constraint language restrictions.

The *ℓth power* of a τ-structure $\mathcal{B}$ is the τ-structure $\mathcal{B}^\ell$ with universe B^ℓ and relations

$$r^{\mathcal{B}^\ell} = \Big\{ \big((b_{11}, \ldots, b_{1\ell}), \ldots, (b_{k1}, \ldots, b_{k\ell})\big) \;\Big|\; (b_{1j}, \ldots, b_{kj}) \in r^{\mathcal{B}} \text{ for } 1 \leq j \leq \ell \Big\}$$

for every k-ary $r \in \tau$. An *(ℓ-ary) polymorphism* of $\mathcal{B}$ is a homomorphism from $\mathcal{B}^\ell$ to $\mathcal{B}$. Equivalently, an ℓ-ary polymorphism of $\mathcal{B}$ can be described as an ℓ-ary operation h on B (that is, a mapping $h : B^\ell \to B$) such that for every k, every k-ary $r \in \tau$, and every matrix $(b_{ij})_{\substack{1 \leq i \leq k \\ 1 \leq j \leq \ell}} \in B^{k \times \ell}$ the following holds: If all columns of the matrix are in $r^{\mathcal{A}}$, then the k-tuple obtained by applying h to each row of the matrix is in $r^{\mathcal{A}}$.

Example 7. Consider the Boolean structure $\mathcal{B}$ with one ternary relation

$$R = \big\{(0,1,0),(0,1,0),(1,0,0),(1,1,1)\big\}$$

Observe that R is the set of solutions to the linear equation $x_1 + x_2 + x_3 = 1$ over GF(2). Then the mapping $f : \{0,1\}^2 \to \{0,1\}$ defined by $f(x,y) = x+y+1$ (addition in GF(2)) is a binary polymorphism of $\mathcal{B}$. To see this, note that if the two columns of the matrix

$$\begin{pmatrix} b_1 & c_1 \\ b_2 & c_2 \\ b_3 & c_3 \end{pmatrix}$$

solve the equation $x_1 + x_2 + x_3 = 1$, then so does the vector

$$\begin{pmatrix} f(b_1,c_1) \\ f(b_2,c_2) \\ f(b_3,c_3) \end{pmatrix} = \begin{pmatrix} b_1 + c_1 + 1 \\ b_2 + c_2 + 1 \\ b_3 + c_3 + 1 \end{pmatrix}$$

The set of all polymorphisms of a structure $\mathcal{B}$, denoted by $\mathrm{Pol}(\mathcal{B})$, is called the *clone* of $\mathcal{B}$. The following lemma establishes a fundamental connection between the complexity of $\textsc{Csp}(-,\mathcal{B})$ and the clone of $\mathcal{B}$ that underlies the algebraic approach:

Lemma 8 (Jeavons [25]). *Let $\mathcal{B},\mathcal{B}'$ be structures over the same universe. If $\mathrm{Pol}(\mathcal{B}) \subseteq \mathrm{Pol}(\mathcal{B}')$ then $\textsc{Csp}(-,\mathcal{B}')$ is polynomial time reducible to $\textsc{Csp}(-,\mathcal{B})$.*

In particular, if $\mathrm{Pol}(\mathcal{B}) = \mathrm{Pol}(\mathcal{B}')$ then $\textsc{Csp}(-,\mathcal{B})$ and $\textsc{Csp}(-,\mathcal{B}')$ are polynomial time equivalent.

In other words: The larger the clone of a structure, the simpler the corresponding CSP. Intuitively, this is plausible, because more complex relations will have fewer polymorphisms. Note that the clone of every structure contains all projections $f(x_1,\ldots,x_\ell) = x_i$. Thus if the clone of a structure $\mathcal{B}$ (with at least two elements) contains only the projections, then $\textsc{Csp}(-,\mathcal{B})$ is certainly hard. The converse of this observation fails; it is easy to construct structures $\mathcal{B}$ such that $\textsc{Csp}(-,\mathcal{B})$ is hard and $\mathrm{Pol}(\mathcal{B})$ does not only contain projections.

An *algebra* is a pair (A,F) consisting of a set A and a set F of operations on A. With every structure $\mathcal{B}$ we associate the algebra $\mathbf{A}(\mathcal{B}) = (B,\mathrm{Pol}(\mathcal{B}))$. It follows immediately from Lemma 8 that for all structures $\mathcal{B},\mathcal{B}'$, if $\mathbf{A}(\mathcal{B}) = \mathbf{A}(\mathcal{B}')$ then $\textsc{Csp}(-,\mathcal{B})$ and $\textsc{Csp}(-,\mathcal{B}')$ are polynomial time equivalent.

The set of *term operations* of an algebra (A,F) is the closure of F and all projections under composition. Since a clone contains all projections and is closed under composition, the term operations of $\mathbf{A}(\mathcal{B})$ are precisely the operations in $\mathrm{Pol}(\mathcal{B})$. An algebra is *idempotent* if all its term operations are idempotent, that is, $f(a,\ldots,a) = a$ for all a.

Lemma 9 (Bulatov, Jeavons, and Krokhin [8]). *For every structure $\mathcal{B}$ there exists a structure $\mathcal{B}'$ such that $\mathbf{A}(\mathcal{B}')$ is idempotent, and $\textsc{Csp}(-,\mathcal{B})$ and $\textsc{Csp}(-,\mathcal{B}')$ are polynomial time equivalent.*

For the readers familiar with the terminology, let us remark that the structure $\mathcal{B}'$ of Lemma 9 is an expansion of the *core* of $\mathcal{B}$ by unary relations $\{b\}$ for all elements b. In particular, this implies that the cardinality of the universe of $\mathcal{B}'$ is less than or equal to the cardinality of the universe of $\mathcal{B}$.

A *subalgebra* of an algebra (A, F) is an algebra (A', F'), where A' is a subset of A that is closed under all operations in F and F' consists of the restrictions of the operations in F to A'. A *homomorphic image* of (A, F) is an algebra (A', F') such that there exist surjective mappings $h : A \to A'$ and $\iota : F \to F'$ such that for all ℓ-ary operations $f \in F$, $\iota(f)$ is an ℓ-ary operation on A' with

$$h(f(a_1, \ldots, a_\ell)) = \iota(f)(h(a_1), \ldots, h(a_\ell))$$

for all tuples $(a_1, \ldots, a_\ell) \in A^\ell$. A *factor* of an algebra (A, F) is a homomorphic image of a subalgebra of (A, F). A factor is *nontrivial* if it has at least two elements. Now we are ready to state the main conjecture by Bulatov, Jeavons, and Krokhin [8]:

Conjecture 2 (BJK-Conjecture). *For every structure $\mathcal{B}$, if $\mathbf{A}(\mathcal{B})$ is idempotent, then the problem* $\textsc{Csp}(-, \mathcal{B})$ *is hard if $\mathbf{A}(\mathcal{B})$ has a nontrivial factor all of whose operations are projections, and tractable otherwise.*

By Lemma 9, the BJK-Conjecture implies the Dichotomy Conjecture. But in addition, the BJK-Conjecture predicts where the dividing line between tractable and hard instances is located. Bulatov, Jeavons, and Krokhin [8] proved one direction of the BJK-Conjecture: If $\mathbf{A}(\mathcal{B})$ has a nontrivial factor all of whose operations are projections, then $\textsc{Csp}(-, \mathcal{B})$ is hard. It follows from Theorems 5 that the BJK-Conjecture holds for all two-element structures.

Theorem 10 (Bulatov [6]). *The BJK-Conjecture (and hence the Dichotomy Conjecture) holds for all three-element structures $\mathcal{B}$.*

Bulatov also proved the BJK-Conjecture for graphs [5] and so-called *conservative CSPs* [4], in which the set of values for each variable can be restricted arbitrarily. Conservative CSPs can be characterised as problems $\textsc{Csp}(-, \mathcal{B})$ for structures $\mathcal{B}$ such that every ℓ-ary polymorphism f satisfies $f(b_1, \ldots, b_\ell) \in \{b_1, \ldots, b_\ell\}$ for all $b_1, \ldots, b_\ell \in B$.

3 Structural restrictions

We start with an example that illustrates a simple, but important algorithmic idea:

Example 11. A *labelled tree* is a structure $\mathcal{T} = (T, E, P_1, \ldots, P_n)$, where E is binary, $n \geq 0$, and $P_1, \ldots, P_n$ are unary, such that the restriction (T, E) is a tree. Let T denote the class of all labelled trees. Then $\textsc{Csp}(\mathsf{T}, -)$ is tractable.

To see this, it will be most convenient to view $\textsc{Csp}(\mathsf{T}, -)$ as a homomorphism problem (as described in Section 1.2). The input consists of a labelled tree $\mathcal{T} =$

$(T, E, P_1, \dots, P_n)$ and a structure $\mathcal{B} = (B, F, Q_1, \dots, Q_n)$, where without loss of generality we assume that $\mathcal{T}$ and $\mathcal{B}$ have the same vocabulary. We fix an arbitrary root r for $\mathcal{T}$ and direct the edges away from the root. For $t \in T$, let $\mathcal{T}_t$ denote the induced subtree of $\mathcal{T}$ whose nodes are t and all its descendants in $\mathcal{T}$. For $t \in T$, $b \in B$, we write $t \sqsubseteq b$ if $(t \in P_i \implies b \in Q_i)$ for $1 \leq i \leq n$. Note that $t \sqsubseteq b$ is a necessary condition for the existence of a homomorphism from $\mathcal{T}$ to $\mathcal{B}$ that maps t to b.

For every node $t \in T$, let $H(t) \subseteq B$ be the set of all $b \in B$ such there is a homomorphism from $\mathcal{T}_t$ to $\mathcal{B}$ that maps t to b. The sets $H(t)$ can computed in the following way:

- If t is a leaf, then $H(t)$ consists of all $b \in B$ such that $t \sqsubseteq b$.
- If t has children $t_1, \dots, t_n$, we first compute $H(t_1), \dots, H(t_n)$ recursively. $H(t)$ consists of all $b \in B$ such that $t \sqsubseteq b$ and there exist $b_1 \in H(t_1), \dots, b_n \in H(t_n)$ such that $(b, b_1), \dots, (b, b_n) \in F$.

Then there is a homomorphism from $\mathcal{T}$ to $\mathcal{B}$ if and only if $H(r) \neq \emptyset$. Clearly, this yields a polynomial time algorithm.

The simple algorithmic idea underlying the example can be generalised from trees to structures that are, in some sense, "similar" to trees. It will be convenient to phrase the following definitions in terms of hypergraphs: A *hypergraph* is a pair $\mathcal{H} = (V, E)$ consisting of a finite set V of *vertices* and a set $E \subseteq 2^V$ of subsets of V called *(hyper)edges*. With each τ-structure $\mathcal{A}$ we associate a hypergraph $\mathcal{H}(\mathcal{A})$ as follows: The vertex set of $\mathcal{H}(\mathcal{A})$ is the universe of $\mathcal{A}$, and for all k, all k-ary $r \in \tau$, and all tuples $(a_1, \dots, a_k) \in r^{\mathcal{A}}$, the set $\{a_1, \dots, a_k\}$ is an edge of $\mathcal{H}(\mathcal{A})$. For a CSP-instance $\mathcal{I}$, we let $\mathcal{H}(\mathcal{I}) = \mathcal{H}(\mathcal{A}(\mathcal{I}))$. Note that the vertices of $\mathcal{H}(\mathcal{I})$ are the variables of $\mathcal{I}$ and the edges of $\mathcal{H}(\mathcal{I})$ are the scopes of the constraints of $\mathcal{I}$, where the *scope* of a constraint $Rx_1 \dots x_k$ is $\{x_1, \dots, x_k\}$.

A *tree decomposition* of a hypergraph $\mathcal{H} = (V, E)$ is a pair $(\mathcal{T}, B)$, where $\mathcal{T}$ is a tree and $B = (B_t)_{t \in T}$ a family of subsets of V such that for each $e \in E$ there is a node $t \in T$ with $e \subseteq B_t$, and for each $v \in V$ the set $\{t \in T \mid v \in B_t\}$ is connected in $\mathcal{T}$. The sets B_t are called the *bags* of the decomposition. The *width* of the decomposition (T, B) is $\max\{|B_t| \mid t \in T\} - 1$, and the *tree width* of $\mathcal{H}$, denoted by $\mathrm{tw}(\mathcal{H})$, is the minimum of the widths of all tree decompositions of $\mathcal{H}$. Figure 1 gives an example.

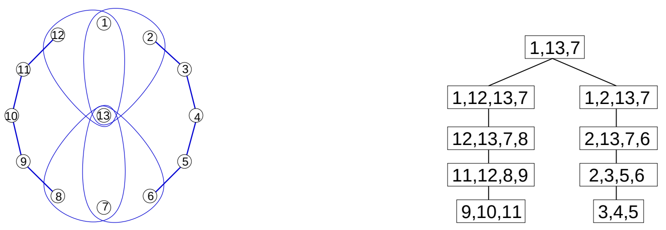

Figure 1. A hypergraph $\mathcal{H}$ and a tree decomposition of $\mathcal{H}$ of width 3

The *tree width* $\mathrm{tw}(\mathcal{A})$ of a structure $\mathcal{A}$ is defined to be the tree width of its hypergraph $\mathcal{H}(\mathcal{A})$. We say that a class C of structures has *bounded tree width* if there is a k such that $\mathrm{tw}(\mathcal{A}) \leq k$ for all $\mathcal{A} \in \mathsf{C}$. (We shall use a similar terminology for other invariants such as bounded hypertree width later without explicitly defining it.)

It is NP-complete to decide whether a given hypergraph or structure has tree width k if k is given as part of the input [2]. However, for every fixed k there is a linear time algorithm that computes a tree decomposition of width k for a given structure $\mathcal{A}$ if the tree width of $\mathcal{A}$ is k [3].

Tree width may be seen as a measure for the "tree-likeness" of hypergraphs and structures. Freuder [18] generalised Example 11 and proved that for every class C of structures of bounded tree width, the problem $\textsc{Csp}(\mathsf{C}, -)$ is tractable. This can be further generalised: Two structures $\mathcal{A}$ and $\mathcal{A}'$ are *homomorphically equivalent* if there is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$ and a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$. For example, all bipartite graphs with at least one edge are homomorphically equivalent. Observe that CSP-instances $\mathcal{I}$ and $\mathcal{I}'$ for which $\mathcal{A}(\mathcal{I})$ is homomorphically equivalent to $\mathcal{A}(\mathcal{I}')$ and $\mathcal{B}(\mathcal{I})$ is homomorphically equivalent to $\mathcal{B}(I')$ are either both satisfiable or both unsatisfiable.

Example 12. Let C be a class of structures such that each structure $\mathcal{A} \in \mathsf{C}$ is homomorphically equivalent to a labelled tree. Then $\textsc{Csp}(\mathsf{C}, -)$ is tractable.

To prove this, again we view $\textsc{Csp}(\mathsf{C}, -)$ as a homomorphism problem. The input consists of a structure $\mathcal{A} = (A, E, P_1, \ldots, P_n)$ that is homomorphically equivalent to a labelled tree and a structure $\mathcal{B} = (B, F, Q_1, \ldots, Q_n)$.

If we could efficiently compute a labelled tree $\mathcal{T}$ that is homomorphically equivalent to $\mathcal{A}$, then we could solve the problem by testing if there is a homomorphism from $\mathcal{T}$ to $\mathcal{B}$ as in Example 11. Unfortunately, there is no obvious way to find such a tree $\mathcal{T}$ (cf. [14]).

Let us define a game on $\mathcal{A}, \mathcal{B}$. The game is played by two players called *Spoiler* and *Duplicator*. Positions of the game are pairs $(a, b) \in A \times B$ such that $a \sqsubseteq b$ (i.e., $a \in P_i \Rightarrow b \in Q_i$ for $1 \leq i \leq n$). In the initial round of a play, Spoiler chooses an $a_0 \in A$ and Duplicator answers by choosing a $b_0 \in B$ with $b_0 \sqsupseteq a_0$. Then the initial position is (a_0, b_0). In each subsequent round, with current position (a, b), the next position (a', b') is determined as follows: Spoiler chooses a' such that $(a, a') \in E$. Then duplicator chooses b' such that $b' \sqsupseteq a'$ and $(b, b') \in F$. If in some round of the play Duplicator cannot answer, she loses. If she can continue to play forever, she wins.

We claim that there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$ if and only if Duplicator has a winning strategy for the game. The forward direction is easy: If $h : A \to B$ is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$ then Duplicator simply answers every choice a of Spoiler by choosing $h(a)$. For the backward direction, suppose that Duplicator has a winning strategy for the game. We exploit the fact that $\mathcal{A}$ is homomorphically equivalent to a labelled tree. Assume first that $\mathcal{A}$ *is* a labelled tree. We define a mapping $h : A \to B$ as follows: Let a_0 be arbitrary. Suppose Spoiler chooses a_0 in the initial round. For every $a \in A$ there is precisely one path $a_0 a_1 \ldots a_m = a$ from a_0 to a in the tree $\mathcal{A}$. We define $b_0, \ldots, b_m$ to be the Duplicator's answer if the Spoiler plays $a_0, \dots, a_m$ and let $h(a) = b_m$. It is easy to verify that h is indeed a homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

Now suppose that $\mathcal{A}$ is not a tree. Let $\mathcal{T} = (T, E', P'_1, \dots, P'_n)$ be a labelled tree that is homomorphically equivalent to $\mathcal{A}$, and let $g_1 : A \to T$, $g_2 : T \to A$ be homomorphisms from $\mathcal{A}$ to $\mathcal{T}$ and from $\mathcal{T}$ to $\mathcal{A}$, respectively. Then Duplicator has a winning strategy for the game on $\mathcal{T}$, $\mathcal{B}$: If Spoiler plays $t \in T$, she answers as she would have answered in the game on $\mathcal{A}, \mathcal{B}$ if Spoiler had played $g_2(t)$. Hence there is a homomorphism h from $\mathcal{T}$ to $\mathcal{B}$. Then $h \circ g_1$ is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

As the number of positions of the game is quadratic in the size of the input, it can be decided in polynomial time if Spoiler has a winning strategy for the game. Duplicator has a winning strategy if and only if Spoiler does not.

Again, this example can be generalised from trees to structures of bounded tree width. A class C of structures has *bounded tree width modulo homomorphic equivalence* if there is a k such that each structure $\mathcal{A} \in \mathsf{C}$ is homomorphically equivalent to a structure $\mathcal{A}'$ with $\mathrm{tw}(\mathcal{A}') \leq k$.

Theorem 13 (Dalmau, Kolaitis, and Vardi [14]). *Let C be a class of structures of bounded tree width modulo homomorphic equivalence. Then* $\textsc{Csp}(\mathsf{C}, -)$ *is tractable.*

Surprisingly, this theorem has a (partial) converse; for classes of structures of *bounded arity*, bounded tree width modulo homomorphic equivalence is also a necessary condition for the tractability of $\textsc{Csp}(\mathsf{C}, -)$. The *arity* of a structure is the maximum arity of its relations.

Theorem 14 (Grohe [20]). *Assume that* $\mathrm{FPT} \neq \mathrm{W}[1]$. *Let C be a recursively enumerable class of structures of bounded arity.*

Then $\textsc{Csp}(\mathsf{C}, -)$ *is tractable if and only if C has bounded tree width modulo homomorphic equivalence.*

Let us discuss the assumptions of this theorem: $\mathrm{FPT} \neq \mathrm{W}[1]$ is a complexity theoretic assumption from parameterized complexity theory (see [15,17]) that is widely believed to be true. The assumption that C be recursively enumerable is somewhat inessential. With a slightly stronger complexity theoretic assumption, the statement of the theorem can also be proved for classes C that are not recursively enumerable. The only serious restriction is that C be of bounded arity. While many natural CSPs (e.g., Clique, 3-Colourability) have bounded arity, some have not: Sat is one prominent example. The rest of the paper is devoted to $\textsc{Csp}(\mathsf{C}, -)$ for classes C of unbounded arity.

3.1 Unbounded arity

Let $\mathcal{I}$ be a CSP-instance. Observe that the arity of the structure $\mathcal{A}(\mathcal{I})$ is the maximum edge size of the hypergraph $\mathcal{H}(\mathcal{I})$.

The following example shows that there are classes C of unbounded tree width modulo homomorphic equivalence (and thus unbounded arity) such that $\textsc{Csp}(\mathsf{C}, -)$ is tractable:

Example 15. For $n \geq 1$, let r_n be an n-ary relation symbol, and let $\mathcal{A}_n$ be the $\{r_n\}$-structure with universe $\{x_1, \ldots, x_n\}$ and $r_n^{\mathcal{A}} = \big\{(x_1, \ldots, x_n)\big\}$. Let $\mathsf{C} = \{\mathcal{A}_n \mid n \geq 1\}$. It is easy to see that the structure $\mathcal{A}_n$ has tree width $n-1$ and is not homomorphically equivalent to a structure of smaller tree width. Thus C has unbounded tree width modulo homomorphic equivalence.

But $\textsc{Csp}(\mathsf{C}, -)$ is tractable. To see this, let $\mathcal{I}$ be an instance of $\textsc{Csp}(\mathsf{C}, -)$, say, with $\mathcal{A}(\mathcal{I}) = \mathcal{A}_n$. Then $\mathcal{I}$ has a single constraint $R_n x_1 \ldots x_n$. Thus $\mathcal{I}$ is satisfiable if and only if R_n is nonempty, and clearly this can be checked in polynomial time.

At this point, it will be necessary to think about how CSP-instances are actually specified. The crucial question is how to specify the relations in the constraint language. In absence of any specific information about the instances, it seems most reasonable to just list the tuples of the relations explictly. Then the size of the representation of an instance $\mathcal{I}$ is roughly

$$||\mathcal{I}|| = |V| + |D| + \sum_{R \in L} |R| \cdot \text{arity}(R) + \sum_{c \in C} |c|,$$

where L denotes the constraint language of $\mathcal{I}$ and the length $|c|$ of a constraint $c = Rx_1 \ldots x_k$ is defined to be $k+1$. In the following, we call $||\mathcal{I}||$ the *size* of $\mathcal{I}$. Complexity will always be measured in terms of $||\mathcal{I}||$. Note that $||\mathcal{I}|| \geq |V| + |D| + |C|$. In general, $||\mathcal{I}||$ can be much larger than $|V| + |D| + |C|$. The best upper bound we get is $||\mathcal{I}|| = O(|V| + \ell \cdot |D|^{\ell} + \ell \cdot |C|)$, where ℓ is the maximum of the arities of the relations in the constraint language. For the bounded arity case, this means that $||\mathcal{I}||$ is polynomial in $|V| + |D| + |C|$, and thus in this case the choice of representation of the relations is not so significant.

Of course the explicit representation of the relations in the constraint language is not the only representation one can think of. Indeed, if relations of large arity occur in practice they are usually represented implicitly, for example, by the clauses of a $\textsc{Sat}$ instance. Note that a clause with n literals specifies a relation with $2^n - 1$ elements. The complexity of CSPs where the relations in the constraint language are represented implicitly (e.g. by clauses) is studied in [10]. But for the rest of this paper, we only consider the explicit representation.

Let $\mathcal{H} = (V, E)$ be a hypergraph. An *edge cover* of $\mathcal{H}$ is a set $C \subseteq E$ of edges such that $V = \bigcup C$. Here $\bigcup C = \bigcup_{e \in C} e = \{v \in V \mid \exists e \in C : \ v \in e\}$. The *edge cover number* of $\mathcal{H}$, denoted by $\rho(\mathcal{H})$, is the minimum cardinality of an edge cover of $\mathcal{H}$. As usually, the edge cover number of a structure is defined to be the edge cover number of its hypergraph. Note that the structure $\mathcal{A}_n$ of Example 15 has edge cover number 1 and tree width $n-1$.

Example 16. Let C be a class of structures of bounded edge cover number. Then $\textsc{Csp}(\mathsf{C}, -)$ is tractable. We leave it to the reader to verify this straightforward claim.

The observation of the previous example can be combined with the ideas developed for trees and structures of bounded tree width. Let $\mathcal{H} = (V, E)$ be a hypergraph. A *hypertree decomposition of* $\mathcal{H}$ is a triple $(\mathcal{T}, B, C)$, where $(\mathcal{T}, B)$ is a tree decomposition of $\mathcal{H}$ and $C = (C_t)_{t \in T}$ is a family of subsets of E such that for every $t \in V$ we have $B_t \subseteq \bigcup C_t$. The sets C_t are called the *guards* of the decomposition. Note that the guard C_t is an edge cover of the subhypergraph induced by the bag B_t.

The *width* of the decomposition (T, B, C) is $\max\{|C_t| \mid t \in T\}$. The *hypertree width* of $\mathcal{H}$, denoted by $\mathrm{hw}(\mathcal{H})$, is the minimum of the widths of the hypertree decompositions of H.

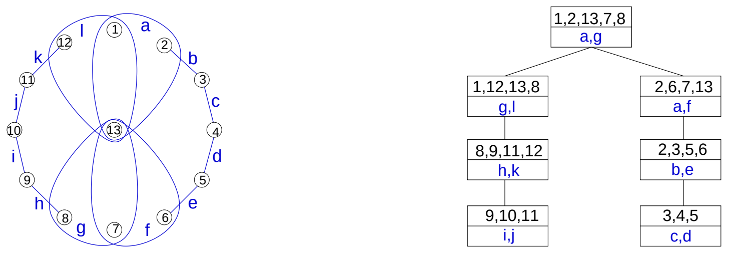

Figure 2. A hypergraph $\mathcal{H}$ and a hypertree decomposition of $\mathcal{H}$ of width 2

Actually, what we call hypertree decomposition here is usually called "generalised hypertree decomposition" in the literature. The "standard" hypertree decompositions incorporate an additional technical condition on how the guards must be arranged in the tree. However, it has been proved in [1] that the width measures derived from the two notions are the same up to a constant factor, and for our purposes, this makes them interchangeable. Let us also remark that hypertree width is called "cover width" in [9]. Another related decomposition called "spread cut decomposition" has been introduced in [12].

As usually, we define the hypertree width of a structure to be the hypertree width of its hypergraph. Gottlob, Leone, and Scarcello [19] proved that $\textsc{Csp}(\mathsf{C}, -)$ is tractable for all classes C of bounded hypertree width. Let us remark that Gottlob et al. stated their results for the problem of evaluating Boolean conjunctive database queries. Constraint satisfaction, homomorphism, Boolean conjunctive query evaluation, and also conjunctive query containment are all equivalent (see [27]). Chen and Dalmau generalised Gottlob et al.'s result to classes of structures of bounded hypertree width modulo homomorphic equivalence:

Theorem 17 (Chen and Dalmau [9]). *Let* C *be a class of structures of bounded hypertree width modulo homomorphic equivalence. Then* $\textsc{Csp}(\mathsf{C}, -)$ *is tractable.*

But this is still not the end of the story. The problem of finding a minimum edge cover of a hypergraph $\mathcal{H} = (V, E)$ has the following integer linear programming (ILP) formulation:

$$\text{minimise} \sum_{e\in E} x_e \text{ subject to } \sum_{\substack{e\in E\\ \text{with } v\in e}} x_e \geq 1 \qquad \text{for all } v \in V,$$

$$x_e \in \{0,1\} \qquad \text{for all } e \in E.$$

Let us consider the linear programming relaxation of this ILP, where the integrality constraints $x_e \in \{0,1\}$ are replaced by the inequalities $x_e \geq 0$. A feasible solution for this linear program is called a *fractional edge cover* of $\mathcal{H}$. The weight $\sum_{e\in E} x_e$ of an optimal solution is called the *fractional edge cover number* of $\mathcal{H}$; it is denoted by $\rho^*(\mathcal{H})$. An optimal fractional edge cover and hence the fractional edge cover number can be computed in polynomial time by solving the linear program. Computing the (integral) edge cover number is NP-complete.

It can be shown (see, e.g., [30], where edge covers are called "set covers") that for every hypergraph $\mathcal{H}$ with n vertices,

$$\rho^*(\mathcal{H}) \leq \rho(\mathcal{H}) \leq \rho^*(\mathcal{H}) \cdot \ln n. \tag{1}$$

The following example shows that the upper bound is fairly tight:

Example 18. Let $\ell \geq 1$ and $\mathcal{H}$ be the following hypergraph: $\mathcal{H}$ has a vertex v_S for every subset S of $\{1,\ldots,2\ell\}$ of cardinality ℓ. Furthermore, $\mathcal{H}$ has an edge $e_i = \{v_S \mid i \in S\}$ for every $i \in \{1,\ldots,2\ell\}$.

Observe that the fractional edge cover number $\rho^*(\mathcal{H})$ is at most 2, because $x_e = 1/\ell$ for every edge e of $\mathcal{H}$ yields a fractional edge cover of weight 2. Actually, it is easy to see that $\rho^*(\mathcal{H}) = 2$. It is also easy to see that $\rho(\mathcal{H}) = \ell + 1$.

Theorem 19 (Grohe and Marx [21]). *Let* C *be a class of structures of bounded fractional edge cover number. Then* $\text{Csp}(\mathsf{C}, -)$ *is tractable.*

Observe that $\text{hw}(\mathcal{H}) \leq \rho(\mathcal{H})$ for every hypergraph $\mathcal{H}$. To see this, consider the hypertree decomposition of $\mathcal{H}$ that consists of a one vertex tree, with a bag that contains all vertices and a guard that is an edge cover. Thus by (1) we have $\text{hw}(\mathcal{H}) \leq \rho^*(\mathcal{H}) \cdot \log n$ for every n-vertex hypergraph. It can be shown that the hypertree width of the graph $\mathcal{H}$ of Example 18 is $\ell + 1 = \Theta(\log n)$ [21]. Conversely, the hypergraph that is the disjoint union of n edges of cardinality 1 has hypertree width 1 and fractional edge cover number n. Thus fractional edge cover number and hypertree width are incomparable. We can combine the two invariants as follows:

A *fractional hypertree decomposition* of a hypergraph $\mathcal{H} = (V, E)$ is a triple $(\mathcal{T}, B, g)$, where $(\mathcal{T}, B)$ is a tree decomposition of $\mathcal{H}$ and $g = (g_t)_{t\in T}$ is a family of functions from E to the nonnegative rationals such that for every $t \in T$ it holds that

$$\sum_{e \text{ with } v\in e} g_t(e) \geq 1 \quad \text{for all } v \in B_t$$

Hence the *guard* g_t is a fractional edge cover of the subhypergraph induced by the bag B_t. The *width* of the decomposition (T, B, C) is $\max\big\{ \sum_{e\in E} g_t(e) \bigm| t \in T\big\}$.

The *fractional hypertree width* of $\mathcal{H}$ is the minimum of the widths of the fractional hypertree decompositions of H.

Conjecture 3. *For every class* C *of structures,* $\textsc{Csp}(\mathsf{C}, -)$ *is tractable if and only if* C *has bounded fractional hypertree width modulo homomorphic equivalence.*

Both directions of this conjecture are open. The forward direction is implied by a technical conjecture stated in [21], which is concerned with the number of maximal subsets of the vertex set of a hypergraph that have a certain fractional edge cover number.

This is a good place to stop. All that remains to do is prove the conjectures.

Acknowledgements

The author is very grateful to Isolde Adler, Albert Atserias, Andrei Bulatov, Hubie Chen, Stephan Kreutzer, Andrei Krokhin, Daniel Marx, Nicole Schweikardt, and Marc Thurley for valuable comments and corrections on an earlier draft of this paper.

References

1. I. Adler, G. Gottlob, and M. Grohe. Hypertree-width and related hypergraph invariants. In S. Felsner, editor, *Proceedings of the 3rd European Conference on Combinatorics, Graph Theory, and Applications*, volume AE of *DMTCS Proceedings Series*, pages 5–10, 2005.
2. S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8:277–284, 1987.
3. H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.
4. A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science*, pages 321–330, 2003.
5. A. Bulatov. H-coloring dichotomy revisited. *Theoretical Computer Science*, 349:31–39, 2005.
6. A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53:66–120, 2006.
7. A. Bulatov, A. Krokhin, and P. Jeavons. The complexity of maximal constraint languages. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 667–674, 2001.
8. A. Bulatov, A. Krokhin, and P. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.
9. H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In P. van Beek, editor, *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming*, volume 3709 of *Lecture Notes in Computer Science*, pages 167–181. Springer-Verlag, 2005.
10. H. Chen and M. Grohe. Constraint satisfaction with succinctly specified relations. In preparation.
11. D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 6. Elsevier, 2006.

12. D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction and spread cut decomposition. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005. To appear.
13. V. Dalmau. Generalized majority-minority operations are tractable. In *Proceedings of the 20th IEEE Symposium on Logic in Computer Science*, pages 438–447, 2005.
14. V. Dalmau, Ph. G. Kolaitis, and M. Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In P. Van Hentenryck, editor, *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer-Verlag, 2002.
15. R.G. Downey and M.R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.
16. T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.
17. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.
18. E.C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.
19. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64:579–627, 2002.
20. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 552–561, 2003.
21. 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*, pages 289–298, 2006.
22. P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.
23. P. Hell, J. Nešetřil, and X. Zhu. Complexity of tree homomorphisms. *Discrete Applied Mathematics*, 70:23–36, 1996.
24. P. Hell, J. Nešetřil, and X. Zhu. Duality and polynomial testing of tree homomorphisms. *Transactions of the American Mathematical Society*, 348(4):1281–1297, 1996.
25. P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.
26. P. Jeavons, D. A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.
27. Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, pages 205–213, 1998.
28. R.E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.
29. T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978.
30. V.V. Vazirani. *Approximation Algorithms.* Springer-Verlag, 2004.