Constraint Satisfaction, Databases, and Logic

Phokion G. Kolaitis*
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, U.S.A
kolaitis@cs.ucsc.edu

1 Introduction

Constraint satisfaction problems constitute a broad class of algorithmic problems that are ubiquitous in several different areas of artificial intelligence and computer science. In their full generality, constraint satisfaction problems are NP-complete and, thus, presumed to be algorithmically intractable. To cope with the intractability of these problems, researchers have devoted considerable research efforts to both the design of heuristic algorithms for constraint satisfaction and the pursuit of "islands of tractability", that is, special cases of constraint satisfaction problems for which polynomial-time algorithms exist.

During the past decade, the pursuit of "islands of tractability" of constraint satisfaction has been intensified and has led to a number of discoveries that have also unveiled tight connections between constraint satisfaction, database theory, logic, and universal algebra. Our goal in this paper is to present an overview of the current state of affairs in the study of the computational complexity of constraint satisfaction with emphasis on the connections of this area of research with database theory and logic. The paper is organized as follows: Section 2 contains the precise definition of the CONSTRAINT SATISFACTION PROBLEM and its reformulation as the HOMOMORPHISM PROBLEM; Section 3 contains some of the connections between constraint satisfaction problems and database theory; the remaining Sections 4, 5, and 6 contain a high-level account of some of the main results about the computational complexity of constraint satisfaction and the pursuit of tractable cases of this problem.

2 The Constraint Satisfaction Problem and the Homomorphism Problem

Constraint satisfaction problems were originally introduced by Montanari [Mon74] to model problems in computer vision. Since that time, however, it has been realized that numerous important problems in artificial intelligence and computer science can be modeled as constraint satisfaction problems (see [Dec92a; PJ97]). An instance of the CONSTRAINT SATISFACTION PROBLEM (CSP) consists of a set of variables, a set of possible values for the variables, and a set of constraints on tuples of variables that restrict the combinations of values that the variables may take; the question is to determine whether there is an assignment of values to the variables so that the constraints are satisfied. More precisely, a CSP-instance is a triple (V, D, C), where V of a finite set of *variables*, D is a finite *domain* of values for the variables, and C is a set *constraints* (t, R), where t is a tuple $t = (x_1, \dots, x_m)$ of variables for some m and R is a relation on D of arity m. *A solution* is a mapping $h : V \to D$ such that, for every constraint $(t,R) \in C$, we have that $h(t) = (h(x_1), \dots, h(x_m)) \in R$.

Consider the Boolean satisfiability problem 3-SAT: given a 3CNF-formula φ with variables $x_1, \dots, x_n$ and clauses $c_1, \dots, c_k$, is φ satisfiable? Such an instance of 3-SAT can be thought of as the CSP-instance in which the set of variables is $V = \{x_1, \dots, x_n\}$, the domain is $D = \{0,1\}$, and the constraints are determined by the clauses of φ. For example, a clause of the form $(x \vee \neg y \vee z)$ gives rise to the constraint $((x, y, z), \{0,1\}^3 - \{(0,1,0)\})$. In an analogous manner, 3-COLORABILITY can be modeled as a constraint satisfaction problem. Indeed, an instance $H = (V,E)$ of 3-COLORABILITY can be thought of as the CSP-instance in which the set of variables is the set V of the nodes of the graph H, the domain is the set $D = \{r, b, g\}$ of three colors, and the constraints are the pairs $((u, v), Q)$, where $(u,v) \in E$ and $Q = \{(r,b)(b,r), (r,g)(g,r), (b,g)(g,b)\}$ is the inequality relation on D.

A *vocabulary* is a finite set of relational symbols $R_1, \dots, R_m$ each of which has a fixed arity. A *relational structure* over some vocabulary is a tuple $\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_m^{\mathbf{A}})$ such that A is a non-empty set, called the *universe* of $\mathbf{A}$, and each $R_i^{\mathbf{A}}$ is a relation on A having the same arity as the symbol R_i. Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures over the same vocabulary. A *homomorphism* h *from* $\mathbf{A}$ *to* $\mathbf{B}$ is a mapping $h : A \to B$ from the universe A of $\mathbf{A}$ to the universe B of $\mathbf{B}$ such that, for every relation $R^{\mathbf{A}}$ of $\mathbf{A}$ and every tuple $(a_1, \dots, a_n) \in R^{\mathbf{A}}$, we have that $(h(a_1), \dots, h(a_n)) \in R^{\mathbf{B}}$. Feder and Vardi [FV98] were the first to point out that the CONSTRAINT SATISFACTION PROBLEM can be identified with the HOMOMORPHISM PROBLEM: given two relational structures $\mathbf{A}$ and $\mathbf{B}$, is there a homomorphism h from $\mathbf{A}$ to $\mathbf{B}$? The intuition behind this identification is that the structure $\mathbf{A}$ represents the variables and the tuples of variables that occur in the constraints, while the structure $\mathbf{B}$ represents the domain of values and the tuples of values that these constrained tuples of variables are

* Partially supported by NSF Grant No. IIS-9907419

allowed to take. Moreover, the homomorphisms from A to B are precisely the assignments of values to variables that satisfy the constraints. For instance, 3-COLORABILITY is equivalent to the problem of deciding whether there is a homomorphism *h* from a given graph H to the complete graph $\mathbf{K}_3 = (\{r,b,g\}, \{(r,b)(b,r),(r,g)(g,r),(b,g)(g,b)\}$ with 3 nodes. More generally, *k*-COLORABILITY, $k \geq 2$, amounts to the existence of a homomorphism from a given graph *H* to the complete graph K_k with A: nodes (also known as the k-clique). Numerous other important NP-complete problems can be viewed as special cases of the HOMOMORPHISM PROBLEM (and, hence, also of the CONSTRAINT SATISFACTION PROBLEM). For example, consider the CLIQUE problem: given a graph H and an integer *k,* does H contain a clique of size A;? A moment's reflection shows that this is equivalent to the question: given a complete graph K_k and a graph H, is there a homomorphism from K_k to H?

The conceptual insights gained from the identification of the CONSTRAINT SATISFACTION PROBLEM with the HOMOMORPHISM PROBLEM have facilitated the use of techniques from universal algebra in the study of constraint satisfaction [Jea98; FV98]). Moreover, they have clarified the tight connections between constraint satisfaction and database theory. We discuss some of these connections in the next section.

3 Constraint Satisfaction and Relational Databases

The most frequently asked queries in relational database systems involve the computation of the *join* of two or more relations in a database. Instead of spelling out the formal definition of the join operation, let us consider for concreteness a database relation *R(A, B, C)* with *A, B, C* as attributes and a database relation *S{B,C,D,E)* with B, C, D, *E* as attributes. Then the join $R \bowtie S$ consists of all quintuples (a,b,c,d,e) such that $(a,b,c) \in R$ and $(b,c,d,e) \in$ 5. Several different researchers, including [Bib88], have pointed out that computing the set of all solutions of a CSP-instance can be viewed as a join evaluation problem. Indeed, if *(V, D, C)* is a CSP-instance, then, for every constraint $(V_{i_1}, \ldots, V_{i_m}), R)$ in C, let $R_i(V_{i_1}, \ldots, V_{i_m})$ be the relation *R* itself viewed as a database relation with attributes $V_{i_1}, \ldots, V_{i_m}$. If $R_1, \ldots, R_t$ is the collection of all database relations obtained this way, then the join $R_1 \bowtie \cdots \bowtie R_t$ consists of all solutions to the CSP-instance (V, D, C).

Join evaluation is a special case of conjunctive query evaluation. Specifically, an n-ary *conjunctive query* Q over a relational vocabulary is a query definable by a positive existential first-order formula of the form $(\exists z_1) \cdots (\exists z_s) \psi(x_1, \ldots, x_n, z_1, \ldots, z_s)$, e $\psi(x_1, \ldots, x_n, z_1, \ldots, z_s)$ is a conjunction of atomic formulas. For example, the binary conjunctive query "there is a path of length 3 from *x1* to X2" is definable by the formula $(\exists z_1)(\exists z_2)(E(x_1, z_1) \wedge E(z_1, z_2) \wedge E(z_2, x_2))$. Note that joins are precisely those conjunctive queries in which none of the variables in ψ is quantified. At other extreme, a *Boolean conjunctive query* is a conjunctive query in which all variables of ψ have been quantified out.

Every finite relational structure A gives rise to a *canonical* Boolean conjunctive query $Q^{\mathbf{A}}$; the positive existential first-order sentence defining $Q^{\mathbf{A}}$ asserts that there exist as many elements as the cardinality of the universe of A and states all atomic facts satisfied by tuples from the universe of A. For example, if A = *(A, E)* is the 3-cycle with *A* = {1,2,3} and *E* = $\{(1,2),(2,3),$ } , then the canonical conjunctive query $Q^{\mathbf{A}}$ is definable by the sentence

$$(\exists x_1)(\exists x_2)(\exists x_3)(E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)).$$

The following basic result, due to Chandra and Merlin [CM77], establishes a strong connection between homomorphisms and conjunctive queries.

Theorem 3.1: [CM771 *The following are equivalent for finite relational structures A and* B.

1. *There is a homomorphism h from* $\mathbf{A}$ *to* $\mathbf{B}$.
2. $Q^{\mathbf{B}} \subseteq Q^{\mathbf{A}}$, *that is,* $Q^{\mathbf{B}}$ *logically implies* $Q^{\mathbf{A}}$.
3. $\mathbf{B} \models Q^{\mathbf{A}}$, *that is, the structure* $\mathbf{B}$ *satisfies the canonical query* $Q^{\mathbf{A}}$ *of* $\mathbf{A}$.

As an example, we saw earlier that a graph G = *(V, E)* is 3-colorable if and only if there is a homomorphism from G to K_3. Consequently, Theorem 3.1 implies that G is 3-colorable if and only if K_3 satisfies the canonical query Q^G of G.

In view of the identification of the CONSTRAINT SATISFACTION PROBLEM with the HOMOMORPHISM PROBLEM, the preceding Theorem 3.1 implies that the CONSTRAINT SATISFACTION PROBLEM can also be identified with two fundamental problems in database theory: *conjunctive query evaluation* and *conjunctive query implication (or containment).* This fundamental connection between constraint satisfaction and database theory was brought to front stage and further investigated in [KVOOal.

4 Computational Complexity of Constraint Satisfaction

The CONSTRAINT SATISFACTION PROBLEM is NP-complete, because it is clearly in NP and also contains NP-hard problems as special cases, including 3-SAT, 3-COLORABILITY, and CLIQUE. As explained in Garey and Johnson's classic monograph [GJ79], one of the main ways to cope with NP-completeness is to identify polynomial-time solvable cases of the problem at hand that are obtained by imposing restrictions on the possible inputs. For instance, HORN 3-SAT, the restriction of 3-SAT to Horn 3CNF-formulas, is solvable in polynomial-time using a unit-propagation algorithm. Similarly, it is known that 3-COLOR ABILITY restricted to graphs of bounded treewidth is solvable in polynomial time (see IDF99]). In the case of constraint satisfaction, the pursuit of tractable cases has evolved over the years from the discovery of isolated cases to the discovery of large "islands of tractability" of constraint satisfaction. In what follows, we will give an account of some of the progress made in this area. We begin by introducing some terminology and notation that will enable us to formalize the concept of an "island of tractability" of constraint satisfaction using the fact that the CONSTRAINT SATISFACTION PROBLEM can be identified with the HOMOMORPHISM PROBLEM.

In general, an instance of the HOMOMORPHISM PROBLEM consists of two arbitrary relational structures A and B. Thus, all restricted cases of this problem can be obtained by imposing restrictions on the input structures A and B.

Definition 4.1: Let *A, B* be two classes of relational structures. We write CSP (.4,B) to denote the restriction of the HOMOMORPHISM PROBLEM to input structures from *A* and *B.* In other words,

$$\text{CSP}(\mathcal{A},\mathcal{B}) \qquad \{(\mathbf{A},\mathbf{B}) : \mathbf{A} \in \mathcal{A}, \mathbf{B} \in \mathcal{B} \text{ and a homomorphism } h : \mathbf{A} \to \mathbf{B} \text{ exists}\}$$

An *island of tractability* of constraint satisfaction is a pair *(A, B)* of classes of relational structures such that CSP(A, *B)* is in the complexity class P of all decision problems solvable in polynomial time.

The ultimate goal in the pursuit of islands of tractability of constraint satisfaction is to identify or somehow characterize all classes *A* and *B* of relational structures such that CSP(A, *B)* is in P. The basic starting point in this investigation is to consider the cases in which one of the two classes *A, B* is as small as possible, while the other is as large as possible. This amounts to considering the cases in which one of *A, B* is the class *All* of all relational structures over some arbitrary, but fixed, relational vocabulary, while the other is a singleton {B} consisting of some fixed structure B over that vocabulary. Thus, the starting points of the investigation is to determine, for every relational structure B, the computational complexity of the decision problems CSP({B}, *All)* and CSP(A//,{B}).

Clearly, for each fixed B, the decision problem CSP({B},,4//) can be solved in polynomial time, because, given a structure A, the existence of a homomorphism from A to B can be checked by testing all functions *h* from the universe *B* of B to the universe *A* of A (the total number of such functions is $|A|^{|B|}$, which is a polynomial number in the size of the structure A). At the other extreme, however, the situation is quite different, since the computational complexity of CSP(.4//, {B}) may very well depend on the particular structure B. Indeed, CSP$(All, \{\mathbf{K}_3\})$ is NP-complete, because it is the 3-COLORABILITY problem; in contrast, CSP$(All, \{\mathbf{K}_2\})$ is in P, because it is the 2-COLORABILITY problem.

For simplicity, in what follows, for every fixed structure B, we put

CSP(B) = CSP(A//,{B})

and call this the *non-uniform* constraint satisfaction problem associated with B. Thus, the first major goal in the study of the computational complexity of constraint satisfaction is to identify those structures B for which CSP(B) is in P. Although this goal has yet to be realized, much progress has been made towards it. The next section contains a bird's-eye view of some of the main results obtained to date.

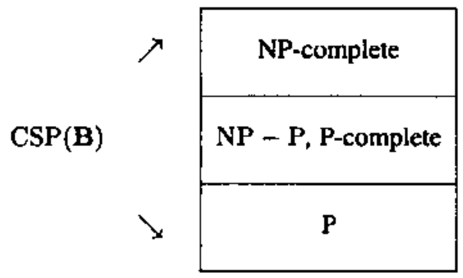
5 The Computational Complexity of Non-Uniform Constraint Satisfaction

The first major result in the study of non-uniform constraint satisfaction problems was obtained by Schaefer [Sch78], who, in effect, classified the computational complexity of all Boolean non-uniform constraint satisfaction problems. A *Boolean* structure is simply a relational structure with a 2-element universe, that is, a structure of the form B = $(\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$. A *Boolean non-uniform constraint satisfaction problem* is a problem of the form CSP(B) with B a Boolean structure. These problems are also known as GENERALIZED SATISFIABILITY PROBLEMS, because they can be viewed as variants of Boolean satisfiability problems in which the formulas are conjunctions of generalized connectives [GJ79]. In particular, they contain the well known problems k-SAT, *k* $\geq$ 2, 1-IN-3-SAT, POSITIVE 1-IN-3-SAT, NOT-ALL-EQUAL 3-SAT, and MONOTONE 3-SAT as special cases. For example, 3-SAT is CSP(B), where B = $(\{0,1\}, R_0, R_1, R_2, R_3)$ and R_i is the set of truth assignments that satisfy a 3-clause in which the first i-literals are negated, $i = 0,1,2,3$ (thus, $R_0 = \{0,1\}^3 - \{(0,0,0)\})$ Similarly, MONOTONE 3-SAT is CSP(B), where B '= $(\{0,1\}, R_0, R_3)$

Ladner [Lad75] showed that if P $\neq$ NP, then there are decision problems in NP that are neither NP-complete, nor they belong to P. Consequently, it is conceivable that a given family of NP-problems contains problems of such intermediate complexity. Schaefer [Sch78], however, showed that the family of all Boolean non-uniform constraint satisfaction problems contains no problems of intermediate complexity.

Theorem 5.1: (Schaefer's Dichotomy Theorem [Sch78])

- *If* $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ *is Boolean structure, then either* CSP(B) *is in* P *or* CSP(B) *is NP-complete. In a picture,*



- *Moreover, there is a polynomial-time algorithm to decide, given a Boolean structure* B, *whether* CSP(B) *is in* P *or it is NP-complete.*

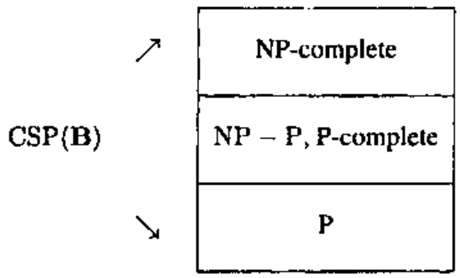
Schaefer [Sch78] actually showed that there are exactly six types of Boolean structures such that CSP(B) is in P, and provided explicit descriptions of them. Specifically, he showed that CSP(B) is in P precisly when at least one of the following six conditions is satisfied:

- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, ιof B is O-*valid,* that is, $R_i^{\mathbf{B}}$ contains the all-zeroes tuple $(0,\ldots,0)$.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, ιof B is \-valid, that is, $R_i^{\mathbf{B}}$ contains the all-ones tuple $(1,\ldots,1)$.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of B is *bijunctive,* that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some 2CNF formula.

- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq$ m, of B is *Horn,* that is, $R^B{}_i$ is the set of truth assignments satisfying some Horn formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq$ m, of B is *dual Horn,* that is, $R^B{}_i$ is the set of truth assignments satisfying some dual Horn formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of B is *affine,* that is, $R^B{}_t$ is the set of solutions to a system of linear equations over the two-element field.

Schaefef's Dichotomy Theorem yields a complete classification of the complexity of CSP(B) for Boolean structures B. At the same time, it raises the challenge of classifying the computational complexity of CSP(B) for arbitrary relational structures B. To this effect, Feder and Vardi [FV98] formulated the following important conjecture.

Conjecture 5.2: (Feder-Vardi Dichotomy Conjecture)
If B = $(\mathbf{B}, \mathbf{R}_1^{\mathbf{B}}, \ldots, R^B{}_m)$ is an arbitrary relational structure, then either CSP(B) is in P or CSP(B) is NP-complete. In a picture,



The Feder-Vardi Dichotomy Conjecture inspired intensive research efforts that resulted into significant advances towards resolving it. In particular, quite recently Bulatov confirmed two important cases of this conjecture that are described next.

Theorem 5.3: *Let* $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ *be a relational structure.*

- (Bulatov [Bul02a]) /f *B* = {0,1,2}, *then either* CSP(B) *is in* P *or* CSP(B) *is NP-complete.*
- (Bulatov IBul()3l) *If every non-empty subset of B is one of the relations $R^B{}_i$ of B, then either* CSP(B) *is in* P *or* CSP(B) *is HP-complete.*

Moreover, in both cases there is a polynomial-time algorithm to decide whether CSP(B) *is in P or it is NP-complete.*

The second of the two cases above is known as *conservative* constraint satisfaction. In effect, it is the case in which all possible unary constraints on the domain are present.

In spite of the progress made, the Feder-Vardi Dichotomy Conjecture remains unresolved for CSP(B) with $|B| \geq 4$. The research efforts towards this conjecture, however, have also resulted into the discovery of broad sufficient conditions for tractability of non-uniform constraint satisfaction that have provided unifying explanations for numerous seemingly disparate tractability results and have also led to the discovery of new islands of tractability of CSP(B). These broad sufficient conditions for tractability of CSP(B) are based on concepts and techniques from two different areas: universal algebra and logic. The approach via universal algebra yields sufficient conditions for tractabilty of CSP(B) in terms of *closure* properties of the relations in B under certain functions on *B.* The approach via logic yields sufficient conditions for tractability in terms of *expressibility* of CSP(B) in Datalog. In the remainder of this section, our primary focus is on the latter approach.

Datalog is the main database query language used in deductive database systems (see rU1189j). In a nutsfell, a Datalog program is a negation-free and function-free logic program. More precisely, a *Datalog program* is a finite set of rules of the form

$$T(\overline{x}) \ :- \ S_1(\overline{y}_1), \ldots, S_r(\overline{y}_r),$$

where T, $S_1, \ldots$, S_r are relation symbols. In effect, the right-hand side of each rule (called the *body* of the rule) is a conjunctive query in which all variables not occurring in the left-hand side (called the *head* of the rule) are existentially quantified. Datalog embodies recursion because relation symbols may occur both in the heads and the bodies of rules. Those that do are the *recursive* or or *intensional database predicates* (IDBs) of the program, while the remaining relation symbols are the *extensional database predicates* (EDBs). One of the IDB predicates is singled out as the *goal* of the program.

As a standard example, the *transitive closure* of the edge relation *E* of a graph H = (V, *E)* is defined by the following Datalog program.

$$\begin{array}{ll} T(x,y) & E(x,y) \\ T(x,y) & E(x,z) \wedge T(z,y) \end{array}$$

Every Datalog program can be evaluated "bottom-up" in a polynomial number of iterations on a given database. For example, the A:-th iteration of the above Datalog program yields the set of all pairs of nodes of H that are connected via a path of length at most K; moreover, at most *n* iterations suffice to compute the transitive closure of E, where *n* is the number of nodes in V. It follows that each fixed Datalog program can be evaluated in time polynomial in the size of a given relational structure. Consequently, if a query *Q* is definable by a Datalog program, then *Q* is in P. Thus, expressibility in Datalog is a sufficient condition for tractability.

Another important feature of Datalog is that queries definable by Datalog programs are *preserved under homomorphisms.* This means that if a structure A satisfies the goal of a Datalog program and there is a homomorphism from A to B, then also B satisfies the goal of the program.

Let B = $(B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ be a relational structure. It is easy to see that, except for trivial situations, CSP(B) is not preserved under homomorphisms, which implies that CSP(B) is *not* expressible in Datalog. Thus, at first sight, it appears that there is no link between expressibility in Datalog and tractability of non-uniform constraint satisfaction. A moment's reflection, however, reveals that the *complement* CSP(B) of CSP(B) *is* preserved under homomorphisms, where

CSP(B) = {A : no homomorphism *h* : A -► B exists}.

Consequently, it is conceivable that, for some structures B, the non-uniform constraint satisfaction problem CSP(B) is in P because its complement $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in Datalog. Feder and Vardi IFV98] pursued this link in depth and demonstrated that expressibility of $\overline{\mathrm{CSP}(\mathbf{B})}$ in Datalog is a unifying explanation for numerous tractability results about CSP(B).

As an important concrete example, consider 2-COLORABILITY, which is the same problem as CSP(K_2). The following Datalog program with *Q* as its goal expresses NON 2-COLORABILITY, since a graph is 2-colorable if and only if it contains no cycles of odd length.

$$\left|\begin{array}{lll} O(X,Y) & :- & E(X,Y) \\ O(X,Y) & :- & O(X,Z),E(Z,W),E(W,Y) \\ Q & :- & O(X,X) \end{array}\right.$$

As two additional important examples, recall that if B = $(\{0,1\}, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is a Boolean structure such that every relation R^B_i is Horn or every relation R^B_i is bijunctive, then CSP(B) is in P. It can be shown that in both these cases CSP(B) is expressible in Datalog.

To gauge the broad spectrum of tractable cases covered by Datalog, it is perhaps worth comparing tractability via Datalog to tractability via closure properties, which, as mentioned earlier, is an approach to tractability based on universal algebra. For this, we need to first introduce some basic concepts from universal algebra. If R is an n-ary relation on a set *B* and / : $B^k \rightarrow$ B is a function, then we say that *R* is *dosed under f* if for every k--tuples $(a_{11},\dots,a_{1n}),\dots,(a_{k1},\dots,a_{kn})$ in R, we have that the n,-tuple $(f(a_{11},\dots,a_{k1}),\dots,f(a_{1n},\dots,a_{kn}))$ is also in *R.* If $\mathbf{B} = (B, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is a relational structure, then a *polymorphism* of B is a function $f : B^k \rightarrow B$ for some $k \geq 1$ such that each relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, is closed under /. We write Pol(B) for the set of all polymorphisms of B. As it turns out, the complexity of CSP(B) is intimately connected to the kinds of functions that Pol(B) contains. In particular, it has been shown that if Pol(B) contains functions that satisfy certain algebraic identities, then CSP(B) is in P. This connection has been investigated in depth and with much success by Jeavons and his collaborators in a sequence of papers, including [JCG97; JCC98], and by Bulatov |Bul02b] (but also Feder and Vardi [FV98] had results along these lines). The following are the presently known most general sufficient conditions for tractability of CSP(B) based on closure properties.

Theorem 5.4: *Let* B = (B, R_i^B,... , R^B_m) *he a relational structure.*

- *If* Pol(B) *contains a near-unanimity function, then* CSP(B) is in P.
- //Pol(B) *contains a set function, then* CSP(B) *is in* P.
- If'Pol(B) *contains a Maltsev function, then* CSP(B) *is in?.*

A k-ary function / with $k \geq 3$ is a *near unanimity function* if *f(x_1,... , X_k) = y,* for every k-tuple *(x_1,... , x_k)* such that at least *k* - 1 of the x_is are equal to *y.* Note that the ternary majority function on {0,1} is a near unanimity function. A k-ary function / with $k \geq 2$ is a *set function* if for every k-tuple *(x_1,...* , x_k), we have that *f(x_1.....,* *x_k)* depends only on the set *{x_1,...* , *x_k}.* Note that the Boolean binary functions A and V arc set functions. Finally, a ternary function *f(x, y, z)* is a *Maltsev function* if, for every *x* and y, it satisfies the identities f(y, *y,x) — f(x,y,y) — x.* Note that the Boolean function $x \oplus y \oplus z$ is a Maltsev function. It should be pointed out that the preceding Theorem 5.4 contains as special cases the non-trivial tractable cases in Schaefer's Dichotomy Theorem 5.1. Indeed, it is known that a Boolean relation is bijunctive if and only if it is closed under the ternary majority operation; moreover, a Boolean relation is Horn (dual Horn) if and only if it is closed under $\wedge$ (respectively, V); finally, a Boolean relation is affine if and only if it is closed under $x \oplus y \oplus z$.

Concerning the comparison between Datalog and closure properties, it can be shown that if Pol(B) contains a near-unanimity function or a set function, then CSP(B) is expressible in Datalog. Thus, expressibility in Datalog subsumes two of the three sufficient conditions for tractability based on closure properties. It also known, however, that there are structures B (in fact, even Boolean structures) such that Pol(B) contains a Maltsev function, but CSP(B) is not expressible in Datalog. At the same time, there are structures B such that Pol(B) does not contain any near unanimity functions, set functions, or Maltsev functions, yet CSP(B) is tractable because CSP(B) is expressible in Datalog.

In what follows in this section, we will take a closer look at the connections between Datalog and non-uniform constraint satisfaction. In particular, we will address the following question: when is CSP(B) expressible in Datalog? As we will see, expressibility of CSP(B) in Datalog can be characterized in terms of pebble games and also in terms of consistency properties.

Combinatorial games are a versatile tool in analyzing the expressive power of logics. The most well known among these games are the Ehrenfeucht-Fraisse-games for first-order logic (see [EFr94]). A different family of games, known as k-pebble games, has been used to study fixed-point logics and infinitary logics with finitely many variables (see [KV901. We now describe a variant of k--pebble games that are suitable for analyzing the expressive power of Datalog [KV95].

Definition 5.5: Let $k \geq 2$ be a positive integer. The *existential k-pebble game* (or, in short, the *$(\exists, k)$-pebble game)* is played between two players, the Spoiler and the Duplicator, on two relational structures A and B according to the following rules: each player has k pebbles labeled 1, . . . , *k;* on the *i-th* move of a round of the game, $1 \leq i \leq k$, the Spoiler places a pebble on an element *a,* of .4, and the Duplicator responds by placing the pebble with the same label on an element b_i of *B.* The Spoiler wins the game at the end of that round, if the correspondence $a_i \mapsto b_i$, $1 \leq i \leq k$, is not a homomorphim between the substructures of A and B with universes *{a_1,...* , a_k} and *[b_1,...* ,b_k}, respectively. Otherwise, the Spoiler removes one or more pebbles, and a new round of the game begins. The Duplicator wins the $(\exists, k)$-pebble game if he has a *winning strategy,* that is to say, a systematic way that allows him to sustain playing "forever", so that the Spoiler can never win a round of the game.

To illustrate this game, let K_m be the complete undirected graph with *in* nodes. For every $k \geq 2$, the Duplicator wins the $(\exists, k)$-pebble game on K_k and K_{k+1}, but the Spoiler wins the $(\exists, k+1)$-pebble game on K_{k+1} and K_k. As another example, let L_s be the s-element linear order, $s \geq 2$. If m < n, then the Duplicator wins the $(\exists, 2)$-pebble game on L_m and L_n, but the Spoiler wins the $(\exists, 2)$-pebble game on L_n and L_m.

Note that the above description of a winning strategy for the Duplicator in the $(\exists, k)$-pebble game is rather informal. The concept of a winning strategy can be made precise, however, in terms of families of partial homomorphisms with appropriate closure and extension properties, where a *partial homomorphism from A to* B is a homomorphism from a substructure of A to a substructure of B.

Definition 5.6: Let *k* be a positive integer. A *winning strategy for the Duplicator in the existential k-pebble game on A and* B is a nonempty family *T* of partial homomorphisms from A to B such that:

1. For every $f \in \mathcal{F}$, the domain dom(/) of / has at most *k* elements.
2. *F* is *closed under subfunctions,* which means that if $g \in$ *T* and $f \subseteq g$, then $f \in \mathcal{F}$.
3. *T* has the *k-forth property,* which means that for every $f \in \mathcal{F}$ with $|\text{dom}(f)| < k$ and every $a \in A$ on which / is undefined, there is a $g \in \mathcal{F}$ that extends / and is defined on *a*.

Intuitively, the second condition provides the Duplicator with a "good" move when the Spoiler removes a pebble from an element A, while the third condition provides the Duplicator with a "good" move when the Spoiler places a pebble on an element of A.

For every positive integer k, let k-Datalog be the collection of all Datalog programs such that each rule has at most A: distinct variables. The next result describes the connection between $(\exists, k)$-pebble games and k-Datalog, and also gives some of the algorithmic properties of $(\exists, k)$-pebble games.

Theorem 5.7: [KV95; KVOOa] *Let k be a positive integer.*

- *Assume that Q is a query definable by a k-Datalog program. If A and* B *are two relational structures such that A satisfies Q and the Duplicator wins the* $(\exists, k)$*-pebble game on A and* B, *then also* B *satisfies Q.*
- *There is a polynomial-time algorithm to decide whether, given two finite structures A and* B, *the Spoiler or the Duplicator wins the* $(\exists, k)$*-pebble game on A and* B.
- *For every finite relational structure* B, *there is a k-Datalog program* B *that expresses the query: given a finite o -structure A, does the Spoiler win the* $(\exists, k)$*-pebble game on A andB?*

The preceding Theorem 5.7 can be used to characterize when CSP(B) is expressible in Datalog.

Theorem 5.8: [KVOOa] *Let k be a positive integer and* B *a finite relational structure. The following statements are equivalent.*

- CSP(B) *is expressible in k-Datalog*
- CSP(B) = {A : *Duplicator wins the* $(\exists, k)$*-pebble game on A and B}.*

Note that if a homomorphism from A to B exists, then the Duplicator wins the $(\exists, k)$-pebble game on A and B using the values of the homomorphism as his strategy. Thus, Theorem 5.8 asserts that CSP(B) is expressible in k-Datalog if and only if the following property holds: whenever the Duplicator wins the $(\exists, k)$-pebble on A and B, a homomorphism from A to B exists.

A striking consequence of Theorems 5.7 and 5.8 is that all non-uniform constraint satisfaction problems CSP(B) for which CSP(B) is expressible in fc-Datalog can be solved in polynomial time using the same algorithm, namely, the polynomial-time algorithm for determining the winner in the (3, fc)-pebble game. Moreover, since expressibility in Datalog subsumes the tractable cases in which Pol(B) contains a near unanimity function or a set function, it follows that the algorithm for determining the winner in the (, fc)-pebble game can also be used to solve CSP(B) in polynomial time in these two major tractable cases.

Many heuristic algorithms for constraint satisfaction involve "constraint propagation", which can be intuitively described as the derivation of new constraints from the original ones. This process has been formalized using various *consistency* concepts that make explicit additional constraints implied by the original constraints. The *strong k-consistency* property is the most important one among them; in informal terms, this property holds when every partial solution on fewer than *k* variables can be extended to a solution on *k* variables IDec92bl. Closely related to this is the process *of establishing strong k-consistency,* which is the question of whether additional constraints can be added to a given instance of the CONSTRAINT SATISFACTION PROBLEM in such a way that the resulting instance is strongly k-consistent and has the same space of solutions as the original one (see [Dec92b; KVOOb] for the formal definitions). It turns out that the following tight connection exists between strong fc-consistency properties and $(\exists, k)$-pebble games.

Theorem 5.9: [KVOOb] *The following are equivalent for a CSP-instance(V,D,C).*

- */t is possible to establish strong k-consistency for (V,D,C).*
- *The Duplicator wins the* $(\exists, k)$*-pebble game on A and* B, *where* A, B *form the instance of the* HOMOMORPHISM PROBLEM *associated with (V,* D, *C).*

Furthermore, when this happens, the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$ *of all winning configurations for the Duplicator in the* $(\exists, k)$*-pebble game gives rise to the largest (and, hence, least constrained) CSP-instance that establishes strong k-consistency for* (V, D, C).

The preceding theorem provides a different perspective on consistency properties and reinforces the usefulness of the

$(\exists, k)$-pebble games in the study of constraint satisfaction. In particular, Theorem 5.9 implies that computing winning strategies for the Duplicator in the $(\exists, k)$-pebble game is the most general form of constraint propagation. Moreover, as an immediate consequence of Theorems 5.8 and 5.9, we obtain another characterization of when CSP(B) is expressible in k-Datalog.

Corollary 5.10: *Let k be a positive integer and* B *be a fixed finite relational structure. Then the following are equivalent.*

- CSP(B) *is expressible in k-Datalog*
- *For every finite relational structure A, establishing strong k-consistency for A and* B *implies that there is a homomorphism from A to* B.

The results presented in this section make a strong case for the importance of the property "CSP(B) in expressible in Datalog" as a broad sufficient condition for tractability of CSP(B), since this property subsumes numerous tractable cases of non-uniform constraint satisfaction. In view of this, one would like to have an efficient algorith to test whether, given a relational structure B, we have that CSP(B) is expressible in Datalog. Such an algorithm could be used as a building block in heuristic algorithms for constraint satisfaction, where one first tests for expressibility in Datalog before resorting to some other exhaustive search procedure. At present, however, no such efficient algorithm is known. In fact, it is not even known whether expressibility of CSP(B) in Datalog is a decidable property. More precisely, the following problem, originally posed by Feder and Vardi [FV98], remains open.

Open Problem: Let $k \geq 2$ be a fixed positive integer. Is there an algorithm to decide whether, given B, we have that CSP(B) is expressible in k-Datalog?

6 Uniform Constraint Satisfaction and Bounded Treewidth

In Section 5, we focused on the pursuit of islands of tractability of non-uniform constraint satisfaction, that is, islands of the form

CSP(B) - CSP(All,{B}),

where B is a fixed relational structure. In particular, we saw that substantial progress has been made towards obtaining a complete classification of all relational structures B for which CSP(B) is solvable in polynomial time. The state of affairs, however, is different for constraint satisfaction problems of the form *CSP(A,B),* where neither *A* nor *B* is a singleton class. We call such problems *uniform constraint satisfaction problems,* because an instance of *CSP{A,B)* is a pair of structures $\mathbf{A} \in \mathcal{A}$ and $\mathbf{B} \in \mathcal{B}$ (unlike CSP(B), where an instance is just a structure A).

At present, we are far away from even coming close to a characterization of all classes *A* and *B* such that CSP(A, *B)* is solvable in polynomial time. Nonetheless, significant progress has been made in the case in which *B* is the class *All* of all relational structures over an arbitrary, but fixed, vocabulary. In what follows, we present the main results concerning islands of tractability of uniform constraint satisfaction of the *form CSP{A, All).*

As is well known, many algorithmic problems that are "hard" on arbitrary graphs become "easy" on trees. This motivated researchers to investigate whether the concept of a tree can be appropriately relaxed while maintaining good computational behavior. As part of their seminar work on graph minors, Robertson and Seymour introduced the concept of *treewidth* and showed that graphs of *bounded treewidth* are "tree-like" structures exhibiting such good behavior. The monograph by Downey and Fellows [DF991 contains complete definitions and characterizations of these concepts. Here, instead of giving the standard definition of treewidth in terms of *tree compositions,* we present an equivalent one in terms of *partial k-trees.*

Definition 6.1: Let *k* be a positive integer.

- A graph H is a *k-tree* if either H is a K_{k+1} -clique or there are a k-tree G, nodes $a_1, \ldots, a_k$ in G forming a $\mathbf{K}_k$-clique, and a node *b* in H - G such that H is obtained from G by connecting *b* to each of the nodes $a_1, \ldots, a_k$ (thus, forming a $\mathbf{K}_{k+1}$-clique).
- A graph is a *partial k-tree* if it is a subgraph (not necessarily induced) of a k-tree.
- The *treewidth* of a graph H, denoted by tw(H), is the smallest *k* such that H is a partial k-tree.
- We write $\mathcal{T}(k)$ to denote the class of all graphs H such that tw(H) < *k.*

Clearly, if T is a tree, then tw(T) = 1. Similarly, if $n \geq 3$ and C_n is the n-element cycle, then tw(C) = 2. At the other end of the scale, $\mathrm{tw}(\mathbf{K}_k) = k - 1$, for every $k \geq 2$. Computing the treewidth of a graph is an intractable problem. Specifically, the following problem is NP-complete: given a graph H and an integer $k \geq 1$, is $\mathrm{tw}(\mathbf{H}) \leq k$? Nonetheless, Bodlaender [Bod93] showed that for every fixed integer k > 1, there is a linear-time algorithm such that, given a graph H, it determines whether or not $\mathrm{tw}(\mathbf{H}) < k$.

The notion of treewidth can be defined for arbitrary finite relational structures A = $(A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ using the *Gaifi man graph* G^A of the structure A. Specifically, tw(A) = tw(G(A)), where G(A) = *(A,E)* and

$E(a, a') \iff a, a'$ occur in a tuple of $R_i^{\mathbf{A}}$, for some *i.*

In what follows, $\mathcal{T}(k)$ will denote the class of all relational structures of treewidth less than *k* over a fixed relational vocabulary.

Dechter and Pearl [DP891 and Freuder lFre90j showed that the classes of structures of bounded treewidth give rise to large islands of tractability of uniform constraint satisfaction.

Theorem 6.2: [DP89; Fre90l *If* $k \geq 2$ *is a positive integer, then CSP{T{k),All)isinP.*

The polynomial-time algorithm for CSP(T(k), *All)* in the above theorem is often described as a *bucket elimination algorithm* [Dec99]. It should be noted that it is not a constraint propagation algorithm. Instead, this algorithm uses the bound on the treewidth to test if a solution to the constraint satisfaction problem exists by solving a join evaluation problem in which all intermediate relations are of bounded arity.

Kolaitis and Vardi LKVOOb], and Dalmau, Kolaitis and Vardi [DKV02] investigated certain logical aspects of the treewidth of a relational structure and showed that this combinatorial concept is closely connected to the canonical conjunctive query of the structure being definable in a fragment of first-order logic with a fixed number of variables. This made it possible to show that the tractability of CSP(T(k), *All)* can be explained in terms of expressibility in k-Datalog. Moreover, it led to the discovery of larger islands of tractability of uniform constraint satisfaction.

Definition 6.3: Let $k \geq 2$ be a positive integer.

- FO^k is the collection of all first-order formulas with at most *k* distinct variables.
- L^k is the collection of all $\mathbf{FO}^k$-formulas built using atomic formulas, conjunction, and existential first-order quantification only.

As an example, it is easy to see that if C_n is the n-element cycle, $n \geq 3$, then the canonical conjunctive query Q^{Cn} is expressible in L^3. For instance, $Q^{\mathbf{C}_4}$ is logically equivalent to

$$(\exists x \exists y \exists z)(E(x,y) \wedge E(y,z) \wedge (\exists y)(E(z,y) \wedge E(y,x))).$$

As mentioned earlier, for every $n \geq 3$, we have that $tw(C_n) = 2$. The next result shows that this relationship between treewidth and number of variables needed to express the canonical conjunctive query is not an accident.

Theorem 6.4: [KVOOb; DKV02] *Let $k \geq 2$ be a positive integer.*

- *//$\mathbf{A} \in \mathcal{T}(k)$, then the canonical conjunctive query Q^A is expressible in L^k.*
- *If* B *is an arbitrary, but fixed, structure, then* $\mathrm{CSP}(\mathcal{T}(k), \{\mathbf{B}\})$ *is expressible in k-Datalog.*
- CSP(T(k), *All) can be solved in polynomial time by determining whether, given a structure $\mathbf{A} \in \mathcal{T}(k)$ and an arbitrary structure* B, *the Spoiler or the Duplicator wins the* (3, *k)-pebble on A and* B.

As a consequence of the above Theorem 6.4, we see that CSP(T(A:),^4//) can be solved in polynomial time using a constraint propagation algorithm that is quite different from the bucket elimination algorithm in Theorem 6.2.

Definition 6.5: Let A and B be two relational structures.

- We say that A and B are *homomorphically equivalent,* denoted $\mathbf{A} \sim_h \mathbf{B}$, if there are homomorphisms $h : \mathbf{A} \to$ B and $h' : \mathbf{B} \to \mathbf{A}$.
- We say that B is the *core* of A, and write core(A) = B, if
  1. B is a substructure of A.
  2. There is no homomorphism *h* : B -> B' from B to a proper substructure B' of B.

Clearly, $\mathrm{core}(\mathbf{K}_k) = \mathbf{K}_k$ and $\mathrm{core}(\mathbf{C}_n) = \mathbf{C}_n$. Moreover, if if H is a 2-colorable graph, then *core(H)* = $\mathbf{K}_2$. It should be note that cores play an important role in database query processing and optimization (see [CM77]). The next result shows that they can also be used to characterize when the canonical conjunctive query is definable in L^k.

Theorem 6.6: *Let $k \geq 2$ be a positive integer and A a relational structure. Then the following statements are equivalent.*

- *Q^A is definable in L^k.*
- *There is a structure $\mathbf{B} \in \mathcal{T}(k)$ such that $\mathbf{A} \sim_h \mathbf{B}$.*
  $\mathrm{core}(\mathbf{A}) \in \mathcal{T}(k)$.

Definition 6.7: If $k \geq 2$ is a positive integer, then $\mathcal{H}(\mathcal{T}(k))$ is the class of all relational structures over some fixed vocabulary that are homomorphically equivalent to a structure in *T(k).*

Equivalently, $\mathcal{H}(\mathcal{T}(k))$ is the class of all relational structures A such that core(A) has treewidth less than *k.*

It should noted that *T(k)* is properly contained in $\mathcal{H}(\mathcal{T}(k))$. Indeed, it is known that there are 2-colorable graphs of arbitrarily large treewidth. In particular, *grids* are known to have these properties (see IDF991). Yet, these graphs are members of $\mathcal{H}(\mathcal{T}(2))$, since their core is K_2.

Theorem 6.& [DKV02] *Let $k \geq 2$ be a positive integer.*

- *If* B *is an arbitrary, but fixed, structure, then CSP(H(T(k)),{B}) is expressible in k-Datalog.*
- *CSP{n(T(k)), All) is in* P. *Moreover,* CSP*Xri(T(k)),*A*ll) can be solved in polynomial time by determining whether, given a structure $\mathbf{A} \in \mathcal{H}(\mathcal{T}(k))$ and an arbitrary structure* B, *the Spoiler or the Duplicator wins the $(\exists, k)$-pebble on A and* B.

The preceding Theorem 6.8 yields new islands of tractability for uniform constraint satisfaction that properly subsume the islands of tractability constituted by the classes of structures of bounded treewidth. However, this expansion of the tractability landscape comes at a certain price. Specifically, as seen earlier, for every fixed *k* > 2, there is a polynomial-algorithm for determining membership in *T(k)* [Bod93]. In contrast, it has been shown that, for every fixed *k* > 2, determining membership in H*(T(k))* is an NP-complete problem [DKV021. Thus, these new islands of tractability are not as easily accessible as the earlier ones.

Since *H{T{k))* contains structures of arbitrarily large treewidth, the bucket elimination algorithm cannot be used to solve $\mathrm{CSP}(\mathcal{H}(\mathcal{T}(k)),$ *All)* in polynomial time. Thus, Theorem 6.8 also shows that determining the winner of the (3, *k)*-pebble is a polynomial time algorithm that applies to islands of tractability not covered by the bucket elimination algorithm.

It is now natural to ask whether there are classes *A* of relational structures over some fixed vocabulary such that they are larger than the classes $\mathcal{H}(\mathcal{T}(k))$ and CSP(\4, *All)* is solvable in polynomial time. A rather unexpected and remarkable new result by Grohe [Gro03] essentially shows that *no* such classes exist, provided a certain complexity-theoretic hypothesis is true.

Theorem 6.9: [Gro03] *Assume that* FPT $\neq$ *W[l]. If A is a recursively enumerable class of relational structures over some fixed vocabulary such that CSP(A,All) is in P, then there is a positive integer k such that $\mathcal{A} \subseteq \mathcal{H}(\mathcal{T}(k))$.*

The hypothesis FPT $\neq$ *W[l]* is a statement in *parametrized complexity* that is analogous to the hypothesis P $\neq$ NP, and it is widely accepted as being true (see [DF99]). In effect, Grohe's Theorem 6.9 is a converse to Theorem 6.8. Together, these two theorems yield a complete characterization of all islands of tractability of the form CSP(A, *All).* Moreover, they reveal that all tractable cases of the form CSP(A, *All)* can be solved by the same polynomial-time algorithm, namely, the algorithm for determining the winner in the $(\exists, k)$-pebble game. In other words, all tractable cases of constraint satisfaction of the form CSP(A, *All)* can be solved in polynomial time using constraint propagation.

References

[Bib88] W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence,* 35:401-413, 1988.

[Bod93] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. In *Proc. 25th ACM Symp. on Theory of Computing,* pages 226-234, 1993.

[Bul02a] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. 43rd IEEE Symposium on Foundations of Computer Science,* pages 649-658, 2002.

[Bul02b] A. Bulatov. Maltsev constraints are tractable. Technical Report PRG-RR-02-05, Oxford University, 2002.

[Bul03] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proc. 18th IEEE Symposium on Logic in Computer Science,* 2003.

[CM77] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th ACM Symp. on Theory of Computing,* pages 77-90, 1977.

[Dec92a] R. Dechter. Constraint networks. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence,* pages 276-185. Wiley, New York, 1992.

[Dec92b] R. Dechter. From local to global consistency. *Artificial Intelligence,* 55(1):87-107, May 1992.

[Dec99] R. Dechter. Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence,* 113(1—2):41—85, 1999.

[DF99] R.G. Downey and M.R. Fellows. *Parametrized Complexity.* Springer-Verlag, 1999.

[DKV02] V. Dalmau, Ph.G. Kolaitis, and M.Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proc. of the Eighth Int. Conference on Principles and Practice of Constraint Programming (CP 2002),* Lecture Notes in Computer Science, pages 311-326. Springer, 2002.

[DP89] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence,* pages 353-366, 1989.

[EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic.* Springer-Verlag, 2nd edition, 1994.

[Frc90] E.C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proc. of 8th National Conference on Artificial Intelligence,* pages 4-9, 1990.

[FV98] 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. on Computing,* 28:57-104, 1998. Preliminary version in *Proc. 25th ACM Symp. on Theory of Computing,* May 1993, pp. 612-622.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., 1979.

[Gro03] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. Submitted for publication, 2003.

[JCC98] P. Jeavons, D. Cohen, and M.C. Cooper. Constraints, consistency and closure. *Artificial Intelligence,* 101(1-2):251-65, May 1998.

[JCG97] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM,* 44(4):527-48, 1997.

[Jea98] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science,* 200(1-2): 185-204, 1998.

[KV90] Ph. G. Kolaitis and M. Y. Vardi. 0-1 laws for infinitary logics. In *Proc. 5th IEEE Symp. on Logic in Computer Science,* pages 156-167, 1990.

[KV95] Ph. G. Kolaitis and M. Y Vardi. On the expressive power of Datalog: tools and a case study. *Journal of Computer and System Sciences,* 51 (1): 11 0-1 34, August 1995.

[KVOOa] Ph.G. Kolaitis and M.Y Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences,* pages 302-332, 2000. Earlier version in: Proc. 17th ACM Symp. on Principles of Database Systems (PODS '98).

[KVOOb] Ph.G. Kolaitis and M.Y Vardi. A game-theoretic approach to constraint satisfaction. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI2000),* pages 175-181,2000.

[Lad75] R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the Association for Computing Machinery,* 22(1): 155-171, 1975.

[Mon74] U. Montanari. Networks of constraints: fundamental properties and application to picture processing. *Information Science,* 7:95-132, 1974.

[PJ97] J. Pearson and P. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway University of London, 1997.

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th ACM Symp. on Theory of Computing,* pages 216-226, 1978.

[U1189] J. D. Ullman. *Database and Knowledge-Base Systems, Volumes I and II.* Computer Science Press, 1989.