Conjunctive-Query Containment and Constraint Satisfaction

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

Moshe Y. Vardi†
Department of Computer Science
Rice University
Houston, TX 77005-1892, U.S.A.
vardi@cs.rice.edu

Abstract

Conjunctive-query containment is recognized as a fundamental problem in database query evaluation and optimization. At the same time, *constraint satisfaction* is recognized as a fundamental problem in artificial intelligence. What do conjunctive-query containment and constraint satisfaction have in common? Our main conceptual contribution in this paper is to point out that, despite their very different formulation, conjunctive-query containment and constraint satisfaction are essentially the same problem. The reason is that they can be recast as the following fundamental algebraic problem: given two finite relational structures A and B, is there a homomorphism $h : A \to B$? As formulated above, the homomorphism problem is *uniform* in the sense that both relational structures A and B are part of the input. By fixing the structure B, one obtains the following *non-uniform* problem: given a finite relational structure A, is there a homomorphism $h : A \to B$? In general, non-uniform tractability results do not uniformize. Thus, it is natural to ask: which tractable cases of non-uniform tractability results for constraint satisfaction and conjunctive-query containment do uniformize?

Our main technical contribution in this paper is to show that several cases of tractable non-uniform constraint satisfaction problems do indeed uniformize. We exhibit three non-uniform tractability results that uniformize and, thus, give rise to polynomial-time solvable cases of constraint satisfaction and conjunctive-query containment. We begin by examining the tractable cases of Boolean constraint satisfaction problems and show that they do uniformize. This can be applied to conjunctive-query containment via *binarization*; in particular, it yields one of the known tractable cases of conjunctive query containment. After this, we show that tractability results for constraint-satisfaction problems that can be expressed using Datalog programs with bounded number of distinct variables also uniformize. Finally, we establish that tractability results for queries with bounded *treewidth* uniformize as well.

*Partially supported by NSF grant CCR-9610257.
†Partially supported by NSF grants CCR-9628400 and CCR-9700061. URL: http://www.cs.rice.edu/~vardi/

1 Introduction

Conjunctive queries have had a conspicuous presence in both the theory and practice of database systems since the 1970s. Conjunctive queries constitute a broad class of frequently used queries, because their expressive power is equivalent to that of the Select-Join-Project queries in relational algebra [AHV95, Ull89]. For this reason, several algorithmic problems concerning conjunctive queries have been investigated in depth. In particular, *conjunctive-query containment* was recognized fairly early as a fundamental problem in database query evaluation and optimization. Indeed, conjunctive-query containment is essentially the same problem as conjunctive query evaluation; moreover, conjunctive-query containment can be used as a tool in query optimization, since query equivalence is reducible to query containment.

Chandra and Merlin [CM77] studied the computational complexity of conjunctive query containment and showed that it is an NP-complete problem. In recent years, there has been renewed interest in the study of conjunctive query containment, because of its close relationship to the problem of answering queries using materialized views [LMSS95, RSU95]. The latter has emerged as a central problem in integrating information from heterogeneous sources, an area that lately has been the focus of concentrated research efforts (see [Ull97] for survey). Since conjunctive-query containment is intractable in its full generality, researchers have embarked on a search for tractable cases. These are obtained by imposing syntactic or structural restrictions on the conjunctive queries Q_1 and Q_2 that serve as input to the problem: "is $Q_1 \subseteq Q_2$?". In particular, Saraiya [Sar91] showed that conjunctive-query containment can be solved in linear time, if every database predicate occurs at most twice in the body of Q_1. More recently, Chekuri and Rajaraman [CR97] showed that, for every $k \geq 1$, conjunctive-query containment can be solved in polynomial time, if Q_2 has *querywidth* at most k and a *query decomposition* of Q_2 of width k is available. The concept of querywidth is closely related to the well-studied concept of *treewidth* of a graph (see [vL90, Bod93]). It should be noted that queries of width 1 are precisely the *acyclic* queries; thus, Chekuri and Rajaraman's [CR97] results extend the earlier work of Yannakakis [Yan81] and Qian [Qia96] on query evaluation and containment for acyclic queries.

Since the early 1970s, researchers in artificial intelligence have investigated a class of combinatorial problems that became known as *constraint-satisfaction problems* (CSP). The input to such a problem consists of a set of variables, a set of possible values for the variables, and a set of constraints

between the variables; the question is to determine whether there is an assignment of values to the variables that satisfies the given constraints. The study of constraint satisfaction occupies a prominent place in artificial intelligence, because many problems that arise in different areas can be modeled as constraint-satisfaction problems in a natural way; these areas include Boolean satisfiability, temporal reasoning, belief maintenance, machine vision, and scheduling (see [Dec92, Kum92, Mes89, Tsa93]). In its full generality, constraint satisfaction is a NP-complete problem. For this reason, researchers in artificial intelligence have pursued both heuristics for constraint-satisfaction problems and tractable cases obtained by imposing restrictions on the constraints (see [MF93, Dec92, PJ97]).

What do conjunctive-query containment and constraint satisfaction have in common? Despite their very different formulation, it turns out that conjunctive-query containment and constraint satisfaction are essentially the same problem. The reason is that they can be recast as the following fundamental algebraic problem: given two finite relational structures A and B, is there a homomorphism $h : A \to B$? Indeed, on the side of conjunctive-query containment, it is well known that $Q_1 \subseteq Q_2$ if and only if there is a homomorphism $h : Q_2^D \to Q_1^D$, where Q_i^D is the *canonical* database associated with the query Q_i, $i = 1, 2$ [CM77]. On the side of constraint satisfaction, a perusal of the literature reveals that all constraint-satisfaction problems studied can be viewed as special cases of the above homomorphism problem [FV93] (see also [Jea97]). Although research in conjunctive-query containment and constraint satisfaction has focused on the search for tractable cases of these problems, it is fair to say that overall there has been little communication between the two communities (but see [Bib88, Dec90, GJC94, PJ97], which drew a connection between constraint satisfaction problems and satisfaction of join dependencies in relational databases). Furthermore, there does not seem to be much awareness that both communities are tackling essentially the same problem. One of our main goals in this paper is to bring this connection to front stage and, thus, enhance the interaction between database theory and artificial intelligence.

As formulated above, the homomorphism problem is *uniform* in the sense that both relational structures A and B are part of the input. By fixing the structure B, one obtains the following *non-uniform* problem CSP(B): given a finite relational structure A, is there a homomorphism $h : A \to B$? Over the past twenty years, researchers in computational complexity have studied such non-uniform problems in an attempt to determine for which structures B the associated CSP(B) problem is tractable and for which it is intractable. The first remarkable success on this front was obtained by Schaefer [Sch78], who pinpointed the computational complexity of *Boolean* CSP(B) problems, in which the structure B is Boolean (i.e., has the set $\{0, 1\}$ as its universe). Schaefer established a *dichotomy* theorem for Boolean CSP(B) problems. Specifically, he identified six classes of Boolean structures and showed that CSP(B) is solvable in polynomial time, if B is in one of these classes, but CSP(B) is NP-complete in all other cases. Note that each Boolean CSP(B) problem can be viewed as a *generalized satisfiability problem*. In particular, Schaefer's [Sch78] dichotomy theorem provides a coherent explanation for the computational complexity of Horn Satisfiability, 2-Satisfiability, One-in-Three Satisfiability, and other such Boolean satisfiability problems. After this, Hell and Nešetřil [HN90] established a dichotomy theorem for CSP(B) problems in which B is an *undirected* graph: if B is 2-colorable, then CSP(B) is solvable in polynomial time; otherwise, CSP(B) is NP-complete. Observe that if K_k is a clique with k nodes, then CSP(K_k) is the k-Colorability problem, $k \geq 2$. Thus, Hell and Nešetřil's dichotomy theorem generalizes the results concerning the computational complexity of the k-Colorability problem for each $k \geq 2$. Motivated by these dichotomy results, Feder and Vardi [FV93] raised the question: is every CSP(B) problem either solvable in polynomial time or NP-complete? Although they did not settle this question, Feder and Vardi [FV93] were able to isolate two conditions that imply polynomial-time solvability of CSP(B) problems; moreover, they argued that all known polynomially solvable CSP(B) problems satisfy one of these conditions. The first condition asserts that the complement of the CSP(B) problem at hand is expressible in Datalog (CSP(B) itself cannot be expressible in Datalog, because it is not a monotone problem); this condition covers such known tractable cases as Horn Satisfiability, 2-Satisfiability, and 2-Colorability. The second condition is group-theoretic and covers Schaefer's [Sch78] tractable class of *affine* satisfiability problems.

In general, non-uniform tractability results do not uniformize. Thus, tractability results for each problem in a collection of non-uniform CSP(B) problems do not necessarily yield a tractable case of the uniform constraint satisfaction problem (or of the conjunctive query containment problem). The reason is that both structures A and B are part of the input to the constraint satisfaction problem, and the running times of the polynomial-time algorithms for CSP(B) may very well be exponential in the size of B. Thus, it is natural to ask: which tractable cases of non-uniform CSP(B) problems uniformize and give rise to uniform tractable cases of constraint satisfaction and, equivalently, to conjunctive-query containment.

Our main technical contribution in this paper is to show that several cases of tractable non-uniform CSP(B) problems do indeed uniformize. We begin by examining the main tractable cases of Boolean CSP(B) problems considered by Schaefer [Sch78]. These are the cases where CSP(B) corresponds to a 2-satisfiability problem, a Horn satisfiability problem, a dual Horn satisfiability problem, or an affine satisfiability problem. For all these cases, uniform polynomial-time algorithms can be obtained by combining polynomial-time algorithms that detect membership in these cases, build a corresponding Boolean formula, and apply the polynomial-time algorithm for satisfiability of such formulas. It should be pointed out, however, that the formula-building algorithms for 2-Satisfiability, Horn satisfiability, and dual Horn satisfiability are in the worst case quadratic in the size of B. In turn, this yields cubic-time algorithms for the corresponding uniform constraint-satisfaction problems. We show here that a better bound can be achieved by designing algorithms that skip the formula-building phase. Although these results are about Boolean constraint-satisfaction problems, they turn out to have applications to conjunctive-query containment. For this, we show that conjunctive-query containment problems can be *binarized* and reduced to Boolean constraint-satisfaction problems. As a concrete application, we show that Sarayia's [Sar91] tractable case of conjunctive-query containment can be derived using this technique.

After this, we focus on the connections between Datalog and constraint satisfaction. As mentioned earlier, Feder and Vardi [FV93] realized that the tractability of many non-uniform CSP(B) problems can be globally explained by the fact that the complement of each of these problems is expressible in Datalog. Using pebble-game techniques in-

troduced in [KV95], we show here that such non-uniform tractability results uniformize, as long as Datalog programs with a bounded number of distinct variables are considered. Specifically, we establish that, for every $k \geq 1$, there is a polynomial-time algorithm for testing whether there is a homomorphism $h : A \to B$, where A and B are two given relational structures such that the complement of CSP(B) is expressible by a Datalog program with at most k distinct variables in each rule.

Up to this point, we have obtained tractable cases of the constraint satisfaction problem "is there a homomorphism $h : A \to B$?" by imposing restrictions on the structure B. At the level of conjunctive-query containment $Q_1 \subseteq Q_2$, this amounts to imposing restrictions on the query Q_1. Our last result yields a tractable case of conjunctive-query containment by imposing restrictions on the query Q_2; it uniformizes another non-uniform tractability result for CSP(B) problems in [FV93]. Specifically, we consider queries of bounded *treewidth* and establish that, for every $k \geq 1$, there is a polynomial-time algorithm for testing whether $Q_1 \subseteq Q_2$, where Q_2 is a conjunctive query of treewidth at most k. For this, we show that every conjunctive query of treewidth $k-1$ is expressible in FO^k, the fragment of first-order logic with at most k distinct variables, and then apply a polynomial-time algorithm for evaluating FO^k queries.

2 Preliminaries

Formally, a n-ary *conjunctive query* Q is a query definable by a positive existential first-order formula $\varphi(X_1, \ldots, X_n)$ having conjunction as its only Boolean connective, that is, by a formula of the form

$$(\exists Z_1) \ldots (\exists Z_m)\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_m),$$

where $\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_m)$ is a conjunction of extensional database predicates. The free variables $X_1, \ldots, X_n$ of the defininig formula are called the *distinguished variables* of Q. Such a conjunctive query is usually written as rule, whose head is $Q(X_1, \ldots, X_n)$ and whose body is $\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_m)$. For example, the formula

$$(\exists Z_1 \exists Z_2)(P(X_1, Z_1, Z_2) \wedge R(Z_2, Z_3) \wedge R(Z_3, X_2)$$

defines a conjunctive query Q, which as a rule becomes

$$Q(X_1, X_2) \text{ :- } P(X_1, Z_1, Z_2), R(Z_2, Z_3), R(Z_3, X_2).$$

If D is a database, then $Q(D)$ is the n-ary relation on D obtained by evaluating the query Q on D, that is, the collection of all n-tuples from D that satisfy the query. Let Q_1 and Q_2 be two n-ary queries having the same tuple of distinguished variables. If $Q_1(D) \subseteq Q_2(D)$ for every database D, we say that Q_1 is *contained in* Q_2, and write $Q_1 \subseteq Q_2$. The *conjunctive-query containment* problem asks: given two conjunctive queries Q_1 and Q_2, is $Q_1 \subseteq Q_2$?

It is well known that conjunctive-query containment can be reformulated as a *conjunctive-query evaluation* problem and also as a *homomorphism* problem. The link to these two other problems is via the *canonical* database D^Q associated with Q. This database is defined as follows. Each variable occurring in Q is considered a distinct element in D^Q. Every predicate in the body of Q is a predicate of D^Q as well; moreover, for every distinguished variable X_i of Q, there is a distinct unary predicate P_i (not occurring in Q). As regards the facts of D^Q, every subgoal in the body of Q gives rise to a tuple in the corresponding predicate of D^Q, and if X_i is a distinguished variable of Q then $P_i(X_i)$ is a fact of D^Q. Thus, in the example above, the canonical database consists of the facts $P(X_1, Z_1, Z_2)$, $R(Z_2, Z_3)$, $R(Z_3, X_2)$, $P_1(X_1)$, $P_2(X_2)$. Recall that a *homomorphism* between two relational structures A and B over the same vocabulary is a mapping $h : A \to B$ such that if $(c_1, \ldots, c_k) \in P^A$, then $(h(c_1), \ldots, h(c_k)) \in P^B$, where P is any predicate symbol in the vocabulary, and P^A and P^B are the interpretations of P on A and B. The relationship between conjunctive query containment, conjunctive query evaluation, and homomorphisms is provided by the following theorem.

Theorem 2.1: [CM77] *Let Q_1 and Q_2 be two n-ary conjunctive queries having the same tuple of distinguished variables. Then the following statements are equivalent.*

- $Q_1 \subseteq Q_2$.
- $(X_1, \ldots, X_n) \in Q_2(D^{Q_1})$, *where* $(X_1, \ldots, X_n)$ *is the tuple of the distinguished variables of* Q_1.
- *There is a homomorphism* $h : D^{Q_2} \to D^{Q_1}$.

Note that every database D gives rise to a Boolean conjunctive query Q^D whose body consists of the conjunction of all facts in D, where we view the elements of the databases as existentially quantified variables. In turn, this makes it possible to show that both conjunctive query evaluation and the existence of homomorphism between two finite relational structures are reducible to conjunctive query containment. In particular, there is a homomorphism $h : A \to B$ if and only if $Q^B \subseteq Q^A$.

Let us now focus on the *constraint-satisfaction* problem. As mentioned earlier, this problem is usually formulated as the question: does there exists an assignment of possible values to given variables, so that certain constraints are satisfied? Instead, we will consider an alternate elegant formulation in terms of homomorphisms. Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of finite relational structures. The *(uniform) constraint-satisfaction problem* CSP($\mathcal{A}, \mathcal{B}$) is the following decision problem: given a structure $A \in \mathcal{A}$ and a structure $B \in \mathcal{B}$, is there a homomorphism $h : A \to B$? Note that, by its very definition, each CSP($\mathcal{A}, \mathcal{B}$) problem is in NP. We write CSP($\mathcal{B}$) for the special uniform case CSP($\mathcal{A}, \mathcal{B}$) in which $\mathcal{A}$ is the class of all finite relational structures over the vocabulary of B. If $\mathcal{B}$ consists of a single structure B, then we write CSP($\mathcal{A}, B$) instead of CSP($\mathcal{A}, \{B\}$). We refer to such problems as *non-uniform* constraint satisfaction problems, because the inputs are just structures A in $\mathcal{A}$. We also write CSP(B) for the special non-uniform case CSP($\mathcal{A}, B$) in which $\mathcal{A}$ is the class of all finite relational structures over the vocabulary of B. Note that if B is a *Boolean structure*, i.e., it has $\{0, 1\}$ as its universe, then CSP(B) is a *generalized satisfiability* problem in the sense of Schaefer [Sch78] (see also [GJ79, LO6, page 260]). For example, if $B = (\{0, 1\}, \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$, then CSP($B$) is equivalent to Positive One-in-Three 3-SAT. Thus, CSP(B) may very well be an NP-complete problem.

We are interested in identifying classes $\mathcal{A}$ and $\mathcal{B}$ such that CSP($\mathcal{A}, \mathcal{B}$) is solvable in polynomial time. Such classes give rise to tractable cases of the constraint satisfaction problem and, hence, of the conjunctive query containment problem as well. For the past twenty years, researchers in computational complexity have investigated CSP(B) problems, and have discovered several polynomial-time cases. As a general rule, however, non-uniform tractable results do not uniformize. Indeed, it is not hard to construct classes $\mathcal{A}$ and

$\mathcal{B}$ of finite relational structures such that CSP($\mathcal{A}, \mathcal{B}$) is NP-complete, but for each $B \in \mathcal{B}$ the non-uniform CSP($\mathcal{A}, B$) problem is solvable in polynomial time. For example, let $\mathcal{K}$ be the class of all finite cliques, and let $\mathcal{G}$ be the class of all finite undirected graphs. It is clear that CSP($\mathcal{K}, \mathcal{G}$) is NP-complete, since it is equivalent to the CLIQUE problem. For every fixed finite undirected graph G, however, one can determine in a constant number of steps whether G has a clique of size k. This example is not isolated, since other NP-complete problems can be viewed this way. In particular, if $\mathcal{P}$ is the class of all finite paths, then CSP($\mathcal{P}, \mathcal{G}$) is equivalent to the Hamiltonian Path Problem, whereas for every finite graph G there is a linear-time algorithm for CSP($\mathcal{P}, G$). These negative results notwithstanding, in the sequel we will establish that several interesting non-uniform tractable cases do uniformize and give rise to tractable cases of constraint satisfaction and conjunctive query containment.

3 Boolean Constraint Satisfaction

Schaefer studied the computational complexity of Boolean CSP(B) problems, for which he established a *dichotomy* [Sch78]. More specifically, he identified six classes of Boolean structures and showed that CSP(B) is solvable in polynomial time, if B is in one of these classes, but CSP(B) is NP-complete in all other cases. This classification is in terms of defining formulas. A k-ary Boolean relation R can be viewed as a set of truth assignments on the propositional variables $p_1, \ldots, p_k$. Thus, for each k-ary Boolean relation R there is a propositional formula δ_R over the variables $p_1, \ldots, p_k$ such that $R = models(\delta_R)$. We call δ_R a *defining formula* of R, and we say that R is *definable by* δ_R. Schaefer showed that for a Boolean structure B, we have that CSP(B) is in PTIME if one of the following six cases holds:

1. each relation in B contains the tuple $\langle 0, \ldots, 0 \rangle$,
2. each relation in B contains the tuple $\langle 1, \ldots, 1 \rangle$,
3. each relation in B is *Horn* (i.e., definable by a CNF formula with at most one positive literal per clause),
4. each relation in B is *dual Horn* (i.e., definable by a CNF formula with at most one negative literal per clause),
5. each relation in B is *bijunctive* (i.e., definable by a 2-CNF formula),
6. each relation in B is *affine* (i.e., definable by a conjunction of linear equations).[1]

If B is not in any of these six classes, then CSP(B) is NP-complete.

We call a Boolean structure B a *Schaefer structure* if CSP(B) is solvable in polynomial time. We call the class of all Schaefer structures *Schaefer's class*, denoted $\mathcal{SC}$. Our main result in this section is that CSP($\mathcal{SC}$) is solvable in polynomial time, which means that Schaefer's tractability results completely uniformize. As a first step, we need to show that structures in $\mathcal{SC}$ can be recognized in polynomial time. This follows from results in [DP92, Sch78].

Theorem 3.1: *The class $\mathcal{SC}$ is recognizable in polynomial time.*

[1] A linear equation is a formula of the form $(p_{i_1} \oplus p_{i_2} \oplus \ldots \oplus p_{i_l}) \leftrightarrow$ **false** or $(p_{i_1} \oplus p_{i_2} \oplus \ldots \oplus p_{i_l}) \leftrightarrow$ **true**.

Proof: (Sketch) The first two cases are trivially recognizable. Schaefer showed that a Boolean relation R is bijunctive iff the following condition holds: if $t_1, t_2, t_3 \in R$, then $(t_1 \vee t_2) \wedge (t_2 \vee t_3) \wedge (t_1 \vee t_3) \in R$ (here Boolan operations are applied to tuples componentwise). Also Schaefer showed that a Boolean relation R is affine iff the following condition hold: if $t_1, t_2, t_3 \in R$, then $(t_1 \oplus t_2 \oplus t_3) \in R$. Finally, Dechter and Pearl [DP92] showed that a Boolean relation R is Horn (resp., dual Horn) if and only if the following condition holds: if $t_1, t_2 \in R$, then $t_1 \wedge t_2 \in R$ (resp., $t_1 \vee t_2 \in R$).[2] Clearly, these conditions can be checked in polynomial time. In the full paper, we will show that checking whether a relation R is Horn or dual Horn can be done in time $O(n \log n)$, using the congruence-closure algorithm of Paige and Tarjan [PT87]. ∎

We say that a relation R is a *trivial Schaefer relation* if it is covered by the first two cases of Schaefer's classification, and we say that R is a *nontrivial Schaefer relation* if it is covered by the four interesting cases of Schaefer's classification (i.e., Horn, dual Horn, bijunctive, and affine). In the latter cases, the relation R is definable by a formula δ_R with a certain syntactical structure. The next step is to show that given a nontrivial Schaefer relation R, we can construct a defining formula in polynomial time.

Theorem 3.2: *There is a polynomial algorithm that constructs for each nontrivial Schaefer relation R a defining formula δ_R.*

Proof: (Sketch) There are four cases to consider. Dechter and Pearl [DP92] showed how to construct δ_R in polynomial time, when R is Horn or dual Horn. It remains to deal with the cases that R is bijunctive or affine. Let R be a k-ary bijunctive relation. Consider a 2-clause c over $\{p_1, \ldots, p_k\}$. We say that R *satisfies* c, denoted $R \models c$, if $R \subseteq models(c)$. It can be shown that δ_R is $\bigwedge_{R \models c} c$, where the conjunction ranges over all 2-clauses c over $\{p_1, \ldots, p_k\}$. Clearly, δ_R can be constructed in time $O(||R||k^2)$.

Let R be a k-ary affine relation. Note that the linear formula $(p_{i_1} \oplus p_{i_2} \oplus \ldots \oplus p_{i_l}) \leftrightarrow$ **false** (resp., $\leftrightarrow$ **true**) can be viewed as the equation $p_{i_1} + p_{i_2} + \ldots + p_{i_l} = 0$ (resp., $= 1$) over the Boolean field. Let $R' = \{(\mathbf{t}, 1) \mid \mathbf{t} \in R\}$. Each linear equation satisfied by R corresponds to a Boolean $(k+1)$-vector $\mathbf{a} = (a_1, \ldots, a_{k+1})$ such that $a_1 t_1 + \ldots + a_{k+1} t_{k+1} = 0$, for each $\mathbf{t} = (t_1, \ldots, t_{k+1}) \in R'$. Thus, the set of such vectors $\mathbf{a}$ is the nullspace $N_{R'}$ of R'; that is, the vector space of solutions to the homogeneous linear equation system $R'\mathbf{a} = 0$ over the Boolean field, where R' is viewed as a Boolean $|R| \times (k+1)$ matrix (note that $|R| = |R'|$). By the Fundamental Theorem of Linear Algebra, $N_{R'}$ has a dimension that is at most $\min(k+1, |R|)$. By Gaussian elimination, we can convert R' to to a row-echelon matrix and recover an independent basis of $N_{R'}$. Each vector in the basis corresponds to a linear formula in δ_R. ∎

We can now prove the main result of this section.

Theorem 3.3: CSP($\mathcal{SC}$) *is solvable in polynomial time.*

Proof: (Sketch) Suppose we are given a pair A, B of relational structures, where $B \in \mathcal{SC}$. We have to determine whether there is a homomorphism from A to B. By Theorem 3.1, we can determine in polynomial time which of the six tractable cases in Schaefer's classification describes B. If

[2] For precursors of this result see [McK43, Riv74].

B is a trivial Schaefer structure, then there is a homomorphism from A to B, so we can assume that B is a nontrivial Schaefer structure. For each k-ary relation Q in A, let Q' be the corresponding relation in B (i.e., Q and Q' are the interpretations of the same relation symbol). Apply Theorem 3.2 to construct $\delta_{Q'}$ (recall that $\delta_{Q'}$ is a formula over $\{p_1, \ldots, p_k\}$.

We can view each element in Q as a propositional variable. For a tuple $\mathbf{t} = (t_1, \ldots, t_k) \in Q$, let $\delta_{Q'}(\mathbf{t})$ be the formula obtained from $\delta_{Q'}$ by substituting t_i for p_i, $1 \leq i \leq k$. Let $\varphi_Q = \bigwedge_{\mathbf{t} \in Q} \delta_{Q'}(\mathbf{t})$. Note that the length of φ_Q is $O(|Q||\delta_{Q'}|)$. Let $\varphi_A = \bigwedge_{Q \in A} \varphi_Q$. It can be shown that there is a homomorphism from A to B precisely when φ_A is satisfiable. Note, however, that δ_A is a conjunction of Horn clauses, dual Horn clauses, 2-clauses, or linear formula, depending on the type of B. Thus, satisfiability of δ_A can be checked in time that is linear in the length of φ_A in the first three cases, and cubic in the length of φ_A in the fourth case. ∎

When R is an affine relation, the length of the defining formula δ_R constructed above is bounded by the size of R. This is not the case when R is bijunctive, Horn, or dual Horn, where the length of δ_R is proportional to $O(k^2)$ (k is the arity of R), which can be quite larger than the size of R. Thus, the complexity of our algorithm in this cases is cubic. It is possible, however, to skip the formula-building stage of our algorithm and apply a direct algorithm. Essentially, this algorithm tests for satisfiability of φ_A without explicitly constructing it.

Theorem 3.4: *Let $\mathcal{B}$ be the class of Horn, dual Horn, or bijunctive structures. Then* CSP($\mathcal{B}$) *is solvable in quadratic time.*

Proof: (Sketch) We sketch the algorithm for the Horn case (an analogous algorithm works for the dual Horn case). Let R be a k-ary Horn relation. Take $[k] = \{1, \ldots, k\}$. For $X \subseteq [k]$ and $j \in [k]$, we say that R satisfies $X \rightarrow j$ if $R \subseteq models(\bigwedge_{i \in X} p_i \rightarrow p_j)$. To determine if there is a homomorphism from a structure A to a Horn structure B, the algorithm maintains a set One of elements of A that have to be mapped to 1. Initially, One is empty. Let $\mathbf{t}$ be a tuple in a relation Q of A. We define $One(\mathbf{t}) = \{i \mid t_i \in One\}$. The algorithm repeatedly selected a tuple $\mathbf{t}$ be a tuple in a relation Q of A, and add t_j to One if Q' satisfies $One(\mathbf{t}) \rightarrow j$, where Q' is the relation in B that corresponds to Q. When One cannot be enlarged anymore, there is a homomorphism from A to B iff for each tuple $\mathbf{t}$ in a relation Q of A, there is a tuple $\mathbf{t}'$ in the corresponding relation Q' of B such that $One(\mathbf{t}) \subseteq One(\mathbf{t}')$. This algorithm can be implemented to run in time $O(||A|| \cdot ||B||)$. For the bijunctive case, the complexity of our direct algorithm is $O(||A|| \cdot |B| + ||B||)$. ∎

What are the implications of Theorem 3.3 for conjunctive-query containment? At first sight, it seems that its applicability is limited, since Boolean constraint-satisfaction problems correspond to to testing whether $Q_1 \subseteq Q_2$, where Q_1 uses only two variables (corresponding to the Boolean values 0 and 1), which seems very restricted. The critical observation is that every conjunctive-query-containment problem can be converted, with a linear blow-up, to a Boolean constraint-satisfaction problem by encoding all variables in binary notation. If n is the maximum number of variables that occur in Q_1 or in Q_2, then we can encode every variables by a vector of length $\lceil \log n \rceil$ over the variables $\{x_0, x_1\}$. Thus, a k-ary relation becomes a $(k\lceil \log n \rceil)$-ary Boolean relation. Note that since one needs $n\lceil \log n \rceil$ bits to encode n elements, the blow-up of this conversion is linear. We refer to this process as the *binarization* of the original problem.

We now show an application of this technique. A *two-atom* conjunctive query is one in which every database predicate occurs at most twice in the body.

Proposition 3.5: [Sar91] *Testing whether a two-atom conjunctive query Q_1 is contained in a conjunctive query Q_2 can be done in time $O(||Q_1|| + ||Q_2||)$.*

Proof: (Sketch) By binarizing the problem we reduce it to testing the existence of homomorphism from a Boolean structure A to a Boolean structure B, where every relation in B has at most two tuples. By Theorem 3.1, B is bijunctive. By Theorem 3.3 and Theorem 3.4, the test can be done in time $O(||A|| + ||B||)$. Since the blow-up of the binarization process is linear, the claim follows. ∎

4 Datalog and Constraint Satisfaction

A Datalog program is a finite set of rules of the form

$$t_0 \text{ :- } t_1, \ldots, t_m,$$

where each t_i is an atomic formula $R(x_1, \ldots, x_n)$. The relational predicates that occur in the heads of the rules are the *intensional database* predicates (IDBs), while all others are the *extensional database* predicates (EDBs). One of the IDBs is designated as the *goal* of the program. Note that IDBs may occur in the bodies of rules and, thus, a Datalog program is a recursive specification of the IDBs with semantics obtained via least fixpoints of monotone operators (see [Ull89]). Each Datalog program defines a query which, given a set of EDB predicates, returns the value of the goal predicate. Moreover, this query is computable in polynomial time, since the bottom-up evaluation of the least fixpoint of the program terminates within a polynomial number of steps (in the size of the given EDBs). Thus, expressibility in Datalog is a sufficient condition for tractability of a query.

If B is a finite relational structure and $\mathcal{A}$ is a class of structures, then we write $\neg$CSP($\mathcal{A}, B$) for the *complement* of CSP($\mathcal{A}, B$), that is, the class of structures A such that there is no homomorphism $h : A \rightarrow B$. Feder and Vardi [FV93] provided a unifying explanation for the tractability of many non-uniform CSP(B) problems by showing that the complement of each of these problems is expressible in Datalog. Our aim in this section is to obtain stronger uniform tractability results for the collections of constraint satisfaction problems whose complements are expressible in Datalog with a bounded number of distinct variables.

For every positive integer k, let k-Datalog be the collection of all Datalog programs in which every rule has at most k distinct variables. For example, the query Non-2-Colorability is expressible in 4-Datalog, since it is definable by the goal predicate Q of the following Datalog program

$$\begin{aligned} P(X,Y) &\text{ :- } E(X,Y) \\ P(X,Y) &\text{ :- } P(X,Z), E(Z,W), E(W,Y) \\ Q &\text{ :- } P(X,X) \end{aligned}$$

It is well known that Datalog is a fragment of least fixpoint logic LFP, which on the class of all finite structures is, in turn, subsumed by the infinitary logic $\mathcal{L}^{\omega}_{\infty\omega}$ with a finite number of variables (see [KV92]). We are interested here in

fragments of $\mathcal{L}^{\omega}_{\infty\omega}$ that are suitable for the study of Datalog. For every $k \geq 1$, let $\exists\mathcal{L}^{k}_{\infty\omega}$ be the *existential positive* fragment of $\mathcal{L}^{\omega}_{\infty\omega}$ with k variables, that is, the collection of all formulas that have at most k distinct variables and are obtained from atomic formulas using infinitary disjunction, infinitary conjunction, and existential quantification only. The following result is implicit in [KV96] (see also [KV95]).

Theorem 4.1: *Let k be a positive integer. Every k-Datalog query is expressible in $\exists\mathcal{L}^{k}_{\infty\omega}$. Thus, k-Datalog $\subseteq \exists\mathcal{L}^{k}_{\infty\omega}$.*

It should be pointed out that on the class of all finite structures k-Datalog is properly contained in $\exists\mathcal{L}^{k}_{\infty\omega}$, since the latter can express non-computable queries.

Next, we describe certain combinatorial games that will play an important role in the sequel. Let A and B be two relational structures over a common relational vocabulary σ. The *existential k-pebble game on A and B* is played between two players, the *Spoiler* and the *Duplicator*. The Spoiler places k pebbles (one at a time) on elements of A; after each move of the Spoiler, the Duplicator responds by placing a pebble on an element of B. Once all pebbles have been placed, the Spoiler wins if the mapping $a_i \mapsto b_i$, $1 \leq i \leq k$, is not a homomorphism from the substructure of A with universe $\{a_1, \ldots, a_k\}$ to the substructure of B with universe $\{b_1, \ldots, b_k\}$, where a_i and b_i are the elements of A and B that have been pebbled in the i-th move of the Spoiler and the Duplicator. Otherwise, the Spoiler removes one or more pebbles and the game resumes. The Duplicator wins the existential k-pebble game if he has a strategy that allows him to continue playing "forever", that is, the Spoiler can never win a round of the game. The following result from [KV95] shows that expressibility in $\exists\mathcal{L}^{k}_{\infty\omega}$ can be characterized in terms of the existential k-pebble games.

Theorem 4.2: *Let k be a positive integer and Q a Boolean query on a class $\mathcal{C}$ of finite structures. Then the following two statements are equivalent:*

1. *Q is expressible in $\exists\mathcal{L}^{k}_{\infty\omega}$ on $\mathcal{C}$.*
2. *If A and B are two structures in $\mathcal{C}$ such that $A \models Q$ and the Duplicator wins the existential k-pebble game on A and B, then $B \models Q$.*

We will also need a result concerning the computational and descriptive complexity of existential k-pebble games.

Theorem 4.3: *Let k be a positive integer.*

1. *There is a polynomial-time algorithm such that, given two finite relational structures A and B, it determines whether the Spoiler wins the existential k-pebble game on A and B.*
2. *For each finite relational structure B, there is a k-Datalog program that expresses the query "Given A, does the Spoiler win the existential k-pebble game on A and B?".*

Proof: (Sketch) The existence of a polynomial-time algorithm for determining the winner of a (variant of) the existential k-pebble game was established in [KV95]. It can be shown that there is a formula of least fixpoint logic LFP with $2k$ variables that expresses the query: "Given A and B, does the Spoiler win the existential k-pebble game on A and B?". This formula involves existential quantifiers that range over the elements of A and universal quantifiers that range over the elements of B. If the structure B is fixed, then the universal quantifiers can be replaced by finitary conjunctions over the elements of B and, thus, the formula of least fixpoint logic can be transformed to a k-Datalog program that expresses the query: "Given A, does the Spoiler win the existential k-pebble game on A and B?" ∎

We now have all the necessary notation and machinery to establish the main results of this section.

Theorem 4.4: *Let k be a positive integer, B a finite relational structure, and $\mathcal{A}$ a class of finite relational structures such that $B \in \mathcal{A}$. Then the following statements are equivalent.*

1. *$\neg\mathrm{CSP}(\mathcal{A}, B)$ is expressible in k-Datalog on $\mathcal{A}$.*
2. *$\neg\mathrm{CSP}(\mathcal{A}, B)$ is expressible in $\exists\mathcal{L}^{k}_{\infty\omega}$ on $\mathcal{A}$.*
3. *$\neg\mathrm{CSP}(\mathcal{A}, B) = \{A \in \mathcal{A} :$ The Spoiler wins the existential k-pebble game on A and $B\}$.*

Proof: (Sketch) The implication (1) $\Longrightarrow$ (2) follows from Theorem 4.1. To show that (2) $\Longrightarrow$ (3), assume that ψ is an $\exists\mathcal{L}^{k}_{\infty\omega}$ sentence that defines $\neg\mathrm{CSP}(\mathcal{A}, B)$ on $\mathcal{A}$. Let A be a finite relational structure in $\mathcal{A}$. If $A \notin \mathrm{CSP}(\mathcal{A}, B)$, then $A \models \psi$ and, hence, the Spoiler wins the existential k-pebble game on A and B. Indeed, if the Duplicator wins this game, then, by Theorem 4.2, $B \models \psi$, which means that there is no homomorphism from B to B, a contradiction. Conversely, if the Spoiler wins the existential k-pebble game on A and B, then $A \in \neg\mathrm{CSP}(\mathcal{A}, B)$. Indeed, otherwise, there is a homomorphism $h : A \to B$, which will give the Duplicator a winning strategy for the existential k-pebble game on A and B. Finally, the implication (3) $\Longrightarrow$ (1) follows from Theorem 4.3. ∎

By combining Theorems 4.3 and 4.4, we obtain the following uniform tractability result for classes of constraint satisfaction problems expressible in Datalog.

Theorem 4.5: *Let k be a positive integer, $\mathcal{A}$ a class of finite relational structures, and*

$$\mathcal{B} \;=\; \{B \in \mathcal{A} : \neg\mathrm{CSP}(\mathcal{A}, B) \text{ is expressible in } k\text{-Datalog}\}.$$

Then the uniform constraint satisfaction problem $\mathrm{CSP}(\mathcal{A}, \mathcal{B})$ is solvable in polynomial time. Moreover, the running time of the algorithm is $O(n^{2k})$, where n is the maximum of the sizes of the input structures A and B.

We note that it is an open problem whether the class

$$\{B : \neg\mathrm{CSP}(\mathcal{A}, B) \text{ is expressible in } k - \mathrm{Datalog}\}$$

is recursive. In contrast, Schaefer's class ($\mathcal{SC}$), which was the basis for the tractability result of Theorem 3.3, is recursive (per Theorem 3.1).

Remark 4.6: Some remarks concerning the results of this section are in order now.

1. Feder and Vardi [FV93] showed that for every non-uniform $\mathrm{CSP}(B)$ problem there is a certain k-Datalog program π_B such that if the complement of $\mathrm{CSP}(B)$ is expressible in k-Datalog, then π_B expresses it. The preceding Theorems 4.3 and 4.4 give an alternative proof of this result; moreover, they reveal that as π_B we can take the k-Datalog program that expresses the query: "Given A, does the Spoiler win the existential k-pebble game on A and B?".

2. To illustrate an application of Theorem 4.5, consider a k-ary Horn structures B. Then it is easy to verify that CSP(B) is expressible in k-Datalog. Consequently, Theorem 4.5 yields a polynomial-time algorithm for CSP($\mathcal{F}, \mathcal{B}$), where $\mathcal{F}$ is the class of all finite structures and $\mathcal{B}$ is the class of k-ary Horn structures.

∎

5 Bounded Treewidth and Constraint Satisfaction

Up to this point, we found tractable cases of the uniform constraint satisfaction problem CSP($\mathcal{A}, \mathcal{B}$) by imposing restrictions on the class $\mathcal{B}$. In this section, we exhibit tractable cases of CSP($\mathcal{A}, \mathcal{B}$) that are obtained by imposing restrictions on the class $\mathcal{A}$. For this, we consider the concept of *treewidth* of a *relational structure*; this concept was introduced by Feder and Vardi [FV93] and generalizes the concept of treewidth of a graph (see [vL90, Bod93]).

A *tree decomposition* of a finite relational structure A is a labeled tree T such that the following conditions hold:

1. every node of T is labeled by a non-empty subset of the universe V of A,
2. for every relation R of A and every tuple $(a_1, \ldots, a_n)$ in R, there is a node of T whose label contains $\{a_1, \ldots, a_n\}$,
3. for every $a \in V$, the set of nodes X of T whose labels include a forms a subtree of T.

The *width* of a tree decomposition T is the maximum cardinality of a label of a node in T minus 1. Finally, we say that a structure A is of *treewidth* k if k is the smallest positive integer such that A has a tree decomposition of width k.

For every $k \geq 1$, let $\mathcal{A}(k)$ be the class of all finite relational structures of treewidth k. Feder and Vardi [FV93] showed for every finite relational structure B, there is a polynomial-time algorithm for the non-uniform constraint satisfaction problem CSP($\mathcal{A}(k), B$). Here, we show that these non-uniform tractability results do uniformize.

Theorem 5.1: *Let k be a positive integer, $\mathcal{A}(k)$ the class of finite relational structures of treewidth k, and $\mathcal{F}$ the class of all finite relational structures. Then the uniform constraint satisfaction problem* CSP($\mathcal{A}(k), \mathcal{F}$) *is solvable in polynomial time.*

Proof: (Sketch) Let A and B be two finite relational structure. From Theorem 2.1 and the accompanied remarks, it follows that the existence of a homomorphism $h : A \to B$ is equivalent to whether $Q^A(B)$ is true, where Q^A is the Boolean conjunctive query whose body consists of the conjunction of all facts in A. In the full paper, we show that if A is a finite relational structure of treewidth $k-1$, then the conjunctive query Q^A is expressible in FO^k, which the fragment of first-order logic with at most k distinct variables; moreover, an FO^k formula equivalent to Q^A can be found in time polynomial in the size of A. Thus, in this case, the existence of a homomorphism $h : A \to B$ reduces to evaluation of FO^k queries. As shown by Vardi [Var95], FO^k has polynomial-time combined complexity, which implies that CSP($\mathcal{A}(k), \mathcal{F}$) is solvable in polynomial time. ∎

In the full paper, we show that the treewidth of a structure A coincides with the treewidth of the *Gaifman* graph of A, that is, the graph that has the elements of the universe of A as nodes and is such that there is an edge between two nodes if and only if the corresponding elements appear in a tuple in one of the relations of A. Bodlaender [Bod93] showed that, for every $k \geq 1$, there is a polynomial-time algorithm that tests whether a given graph is of treewidth k. It follows that, for every $k \geq 1$, there is a polynomial-time algorithm that tests whether a given finite relational structure is of treewidth k; in other words, each class $\mathcal{A}(k)$ is recognizable in polynomial time.

As mentioned in the introduction, Chekuri and Ramajaran [CR97] showed that the conjunctive query containment problem "is $Q_1 \subseteq Q_2$?" is solvable in polynomial time, when Q_2 has bounded *querywidth*. In the full paper, we show that over vocabularies of bounded arities (which are precisely the vocabularies considered in [CR97]) the querywidth of a conjunctive query Q and the treewidth of the canonical database D^Q differ by a multiplicative constant factor. Thus, Theorem 5.1 can be used to essentially derive the main results of [CR97]. Moreover, there is an important advantage of the concept of bounded treewidth over the concept of bounded querywidth. Specifically, as seen above, the classes of structures of bounded treewidth are polynomial-time recognizable, whereas it is not known whether the same holds true for the classes of queries of bounded querywidth (see [CR97]).

6 Concluding Remarks

In this paper, we brought into center stage the close connection between conjunctive-query containment and constraint satisfaction. Moreover, we showed that several tractable cases of non-uniform CSP(B) problems uniformize and, thus, yield tractable cases of uniform constraint satisfaction and conjunctive query containment.

During the past several years, a group of researchers has pursued tractable cases of constraint satisfaction CSP($\mathcal{A}, \mathcal{B}$) by investigating the class of functions under which the relations in the structures in $\mathcal{B}$ are closed [JC95, JCG95, JCG96, Jea97] (see [PJ97] for a survey). In [FV93], a preliminary investigation has been carried out on the connection between expressibility of CSP(B) problems in Datalog and closure of the relations in B under certain functions. In a forthcoming paper [KV], we elaborate further on this connection and delineate the relationship between the two approaches.

Acknowledgement. We'd like to thank Leonid Libkin for useful discussions.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Addison-Wesley, 1995.

[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.

[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.

[CR97] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In Ph.G. Kolaitis and F. Afrati, editors, *Database Theory - ICDT '97*, volume 1186 of *Lecture Notes in Computer Science*, pages 56–70. Springer-Verlag, 1997.

[Dec90] R. Dechter. Decomposing a relation into a tree of binary relations. *Journal of Computer and System Sciences*, 41:1–24, 1990.

[Dec92] R. Dechter. Constraint networks. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 276–185. Wiley, New York, 1992.

[DP92] R. Dechter and J. Pearl. Structure identification in relational data. *Artificial Intelligence*, 48:237–270, 1992.

[FV93] T.A. Feder and M.Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 612–622, 1993. See full version in `http://www.cs.rice.edu/~vardi/papers`.

[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.

[GJC94] M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposition constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66:57–89, 1994.

[HN90] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[JC95] P. Jeavons and D. Cohen. An algebraic characterization of tractable constraints. In D.-Z. Du and M. Li, editors, *Proceedings of First Annual International Conference on Computing and Combinatorics COCOON '95*, pages 633–642. Springer-Verlag, 1995.

[JCG95] P. Jeavons, D. Cohen, and M. Gyssens. A unifying framework for tractable constraints. In U. Montanari and F. Rossi, editors, *Proceedings of 1st International Conference on Principles and Practice of Constraint Programming, CP95*, pages 276–291. Springer-Verlag, 1995.

[JCG96] P. Jeavons, D. Cohen, and M. Gyssens. A test for tractability. In E.C. Freuder, editor, *Proceedings of 2nd International Conference on Principles and Practice of Constraint Programming, CP96*, pages 267–281. Springer-Verlag, 1996.

[Jea97] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 1997. To appear.

[Kum92] V. Kumar. Algorithms for constraint-satisfaction problems. *AI Magazine*, 13:32–44, 1992.

[KV] Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. Forthcoming.

[KV92] Ph. G. Kolaitis and M. Y. Vardi. Infinitary logic and 0-1 laws. *Information and Computation*, 98:258–294, 1992. Special issue: Selections from the Fifth Annual IEEE Symposium on Logic in Computer Science.

[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):110–134, August 1995. Special Issue: Selections from Ninth Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Nashville, TN, USA, 2-4 April 1990.

[KV96] Ph.G. Kolaitis and M.Y. Vardi. On the expressive power of variable-confined logics. In *Proceedings of 11th Annual IEEE Symposium on Logic in Computer Science - LICS '96*, pages 348–59, 1996.

[LMSS95] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 95–104, 1995.

[McK43] J. McKinsey. The decision problem for some classes of sentence without quantifiers. *Journal of Symbolic Logic*, 8:61–76, 1943.

[Mes89] P. Meseguer. Constraint satisfaction problem: an overview. *AICOM*, 2:3–16, 1989.

[MF93] A.K. Mackworth and E.C. Freuder. The complexity of constraint satisfaction revisited. *Artificial Intelligence*, 59(1-2):57–62, 1993.

[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.

[PT87] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16:73–98, 1987.

[Qia96] X. Qian. Query folding. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 48–55, 1996.

[Riv74] I. Rival. Maximal sublattices of finite distributive lattices. *Proc. AMS*, 44:263–268, 1974.

[RSU95] A. Rajamaran, Y. Sagiv, and J.D. Ullman. Answering queries using templates with binding patterns. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 105–112, 1995.

[Sar91] Y. Saraiya. *Subtree elimination algorithms in deductive databases.* PhD thesis, Department of Computer Science, Stanford University, 1991.

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

[Tsa93] E.P.K. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[Ull89] J. D. Ullman. *Database and Knowledge-Base Systems, Volumes I and II.* Computer Science Press, 1989.

[Ull97] J.D. Ullman. Information integration using logical views. In Ph.G. Kolaitis and F. Afrati, editors, *Database Theory - ICDT '97*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 1997.

[Var95] M.Y. Vardi. On the complexity of bounded-variable queries. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 266–76, 1995.

[vL90] J. van Leeuwen. Graph algorithms. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science A: Algorithms and Complexity*, chapter 10, pages 525–631. Elsevier, Amsterdam, 1990.

[Yan81] M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM Journal of Computing*, 10:310–327, 1981.