Finite Model Theory and Its Applications

Springer

Contents

1 A Logical Approach to Constraint Satisfaction 5
1.1 Introduction 5
1.2 Preliminaries 6
1.3 Computational Complexity of Constraint Satisfaction 10
1.4 Non-Uniform Constraint Satisfaction 13
1.5 Monotone Monadic SNP and Non-Uniform Constraint Satisfaction 16
1.6 Datalog and Non-Uniform Constraint Satisfaction 18
1.7 Datalog, Games, and Constraint Satisfaction 21
1.8 Games and Consistency 23
1.9 Uniform Constraint Satisfaction and Bounded Treewidth 28

References 35

1

A Logical Approach to Constraint Satisfaction

Phokion G. Kolaitis and Moshe Y. Vardi

1.1 Introduction

Since the early 1970s, researchers in artificial intelligence (AI) 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 modelled as constraint-satisfaction problems in a natural way; these areas include Boolean satisfiability, temporal reasoning, belief maintenance, machine vision, and scheduling (cf. [20, 50, 56, 64]). In its full generality, constraint satisfaction is an NP-complete problem. For this reason, researchers in artificial intelligence have pursued both heuristics for constraint-satisfaction problems and tractable cases obtained by imposing various restrictions on the input (cf. [20, 24, 34, 54, 58]).

Over the past decade, it has become clear that there is an intimate connection between constraint satisfaction and various problems in database theory and finite-model theory. The goal of this chapter is to describe several such connections. We start in Section 1.2 by defining the constraint-satisfaction problem and showing how it can be phrased also as a homomorphism problem, conjunctive-query evaluation problem, or join-evaluation problem. In Section 1.3, we discuss the computational complexity of constraint satisfaction and show that it can be studied from two perspectives, a uniform perspective and a non-uniform perspective. We relate both perspectives to the study of the computational complexity of query evaluation. In Section 1.4, we focus on the non-uniform case and describe a Dichotomy Conjecture, asserting that every non-uniform constraint-satisfaction problem is either in PTIME or NP-complete. In Section 1.5, we examine the complexity of non-uniform constraint satisfaction from a logical perspective and show that it is related to

the data complexity of a fragment of existential second-order logic. We then go on in Section 1.6 and offer a logical approach, via definability in Datalog, to establishing the tractability of non-uniform constraint-satisfaction problems. In Section 1.7, we leverage the connection between Datalog and certain pebble games, and show how these pebble games offer an algorithmic approach to solving uniform constraint-satisfaction problems. In Section 1.8, we relate these pebble games to consistency properties of constraint-satisfaction instances, a well-known approach in constraint solving. Finally, in Section 1.9, we show how the same pebble games can be used to identify large "islands of tractability" in the constraint-satisfaction terrain that are based on the concept of bounded treewidth.

Much of the logical machinery used in this chapter is described in detail in Chapter **??**. For a book-length treatment of constraint satisfaction from the perspective of graph homomorphism, see [44]. Two books on constraint programming and constraint processing are [3, 23].

1.2 Preliminaries

The standard terminology in AI formalizes an instance $\mathcal{P}$ of constraint satisfaction as a triple $(V, D, \mathcal{C})$, where

1. V is a set of variables;
2. D is a set of values, referred to as the *domain*;
3. $\mathcal{C}$ is a collection of *constraints* $C_1, \ldots, C_q$, where each constraint C_i is a pair $(\mathbf{t}, R)$ with $\mathbf{t}$ a k-tuple over V, $k \geq 1$, referred to as the *scope* of the constraint, and R a k-relation on D.

A *solution* of such an instance is a mapping $h : V \to D$ such that, for each constraint $(\mathbf{t}, R)$ in $\mathcal{C}$, we have that $h(\mathbf{t}) \in R$, where h is defined on tuples component-wise, that is, if $\mathbf{t} = (a_1, \ldots, a_k)$, then $h(\mathbf{t}) = (h(a_1), \ldots, h(a_k))$. The CONSTRAINT-SATISFACTION PROBLEM asks whether a given instance is *solvable*, i.e., whether it has a solution. Note that, without loss of generality, we may assume that all constraints $(\mathbf{t}, R_i)$ involving the same scope $\mathbf{t}$ have been consolidated to a single constraint $(\mathbf{t}, R)$, where R is the intersection of all relations R_i constraining $\mathbf{t}$. Thus, we can assume that each tuple $\mathbf{t}$ of variables occurs at most once in the collection $\mathcal{C}$.

Consider the Boolean satisfiability problem 3-SAT: given a 3CNF-formula φ with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, is φ satisfiable? Such an instance of 3-SAT can be thought of as the constraint-satisfaction instance in which the set of variables is $V = \{x_1, \ldots, x_n\}$, the domain is $D = \{0, 1\}$, and the constraints are determined by the clauses of φ. For example, a clause of the form $(\neg x \vee \neg y \vee z)$ gives rise to the constraint $((x, y, z), \{0, 1\}^3 - \{(1, 1, 0)\})$. In an analogous manner, 3-COLORABILITY can be modelled as a constraint-satisfaction problem. Indeed, an instance $\mathbf{G} = (V, E)$ of 3-COLORABILITY can be thought of as the constraint-satisfaction instance in which the set

of variables is the set V of the nodes of the graph $\mathbf{G}$, 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 disequality relation on D.

Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures[1] 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 $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$, we have that $(h(a_1), \ldots, h(a_k)) \in R^{\mathbf{B}}$. The existence of a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is denoted by $\mathbf{A} \to \mathbf{B}$, or by $\mathbf{A} \to^h \mathbf{B}$, when we want to name the homomorphism h explicitly. An important observation made in [29][2] is that every such constraint-satisfaction instance $\mathcal{P} = (V, D, \mathcal{C})$ can be viewed as an instance of the HOMOMORPHISM PROBLEM, asking whether there is a homomorphism between two structures $\mathbf{A}_{\mathcal{P}}$ and $\mathbf{B}_{\mathcal{P}}$ that are obtained from $\mathcal{P}$ in the following way:

1. the universe of $\mathbf{A}_{\mathcal{P}}$ is V and the universe of $\mathbf{B}_{\mathcal{P}}$ is D;
2. the relations of $\mathbf{B}_{\mathcal{P}}$ are the distinct relations R occurring in $\mathcal{C}$;
3. the relations of $\mathbf{A}_{\mathcal{P}}$ are defined as follows: for each distinct relation R on D occurring in $\mathcal{C}$, we have the relation $R^{\mathbf{A}} = \{\mathbf{t} : (\mathbf{t}, R) \in \mathcal{C}\}$. Thus, $R^{\mathbf{A}}$ consists of all scopes associated with R.

We call $(\mathbf{A}_{\mathcal{P}}, \mathbf{B}_{\mathcal{P}})$ the *homomorphism instance* of $\mathcal{P}$. Conversely, it is also clear that every instance of the homomorphism problem between two structures $\mathbf{A}$ and $\mathbf{B}$ can be viewed as a constraint-satisfaction instance $\mathrm{CSP}(\mathbf{A}, \mathbf{B})$ by simply "breaking up" each relation $R^{\mathbf{A}}$ on $\mathbf{A}$ as follows: we generate a constraint $(\mathbf{t}, R^{\mathbf{B}})$ for each $\mathbf{t} \in R^{\mathbf{A}}$. We call $\mathrm{CSP}(\mathbf{A}, \mathbf{B})$ the *constraint-satisfaction instance* of $(\mathbf{A}, \mathbf{B})$. Thus, as pointed out in [29], the constraint-satisfaction problem can be identified with the homomorphism problem.

To illustrate the passage from the constraint-satisfaction problem to the homomorphism problem, let us consider 3-SAT. A 3CNF-formula φ with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$ gives rise to a homomorphism instance $(\mathbf{A}_\varphi, \mathbf{B}_\varphi)$ defined as follows:

- $\mathbf{A}_\varphi = (\{x_1, \ldots, x_n\}, R_0^\varphi, R_1^\varphi, R_2^\varphi, R_3^\varphi)$, where R_i^φ is the ternary relation consisting of all triples (x, y, z) of variables that occur in a clause of φ with i negated literals, $0 \leq i \leq 3$; for instance, R_2^φ consists of all triples (x, y, z) of variables such that $(\neg x \vee \neg y \vee z)$ is a clause of φ (here, we assume without loss of generality that the negated literals precede the positive literals).
- $\mathbf{B}_\varphi = (\{0, 1\}, R_0, R_1, R_2, R_3)$, where R_i consists of all triples that satisfy a 3-clause in which the first i literals are negated; for instance, $R_2 = \{0, 1\}^3 - \{1, 1, 0\}$.

Note that $\mathbf{B}_\varphi$ does not depend on φ. It is clear that φ is satisfiable if and only if there is a homomorphism from $\mathbf{A}_\varphi$ to $\mathbf{B}_\varphi$ (in symbols, $\mathbf{A}_\varphi \to \mathbf{B}_\varphi$).

[1] We consider only finite structures in this chapter.
[2] An early version appeared in [30].

As another example, 3-COLORABILITY is equivalent to the problem of deciding whether there is a homomorphism h from a given graph $\mathbf{G}$ 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 $\mathbf{G}$ to the complete graph $\mathbf{K}_k$ with k 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 $\mathbf{G}$ and an integer k, does $\mathbf{G}$ contain a clique of size k? As a homomorphism instance this is equivalent to asking if there is a homomorphism from the complete graph $\mathbf{K}_k$ to $\mathbf{G}$. As a constraint-satisfaction instance, the set of variables is $\{1, 2, \ldots, k\}$, the domain is the set V of nodes of $\mathbf{G}$, and the constraints are the pairs $((i, j), E)$ such that $i \neq j$, $1 \leq i, j \leq k$, and E is the edge relation of $\mathbf{G}$. For another example, consider the HAMILTONICITY PROBLEM: given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ does it have a Hamiltonian cycle? This is equivalent to asking if there is a homomorphism from the structure $(V, C_V, \neq)$ to the structure $(V, E, \neq)$, where C_V is some cycle on the set V of nodes of $\mathbf{G}$ and $\neq$ is the disequality relation on V. NP-completeness of the Homomorphism problem was pointed out explicitly in [53]. In this chapter, we use both the traditional AI formulation of constraint satisfaction and the formulation as the homomorphism problem, as each has its own advantages.

It turns out that in both formulations constraint satisfaction can be expressed as a database-theoretic problem. We start with the homomorphism formulation, which is intimately related to *conjunctive-query evaluation* [48]. A *conjunctive query* Q of arity n is a query definable by a positive existential first-order formula $\varphi(X_1, \ldots, X_n)$ having conjunction as its only propositional 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 (positive) atomic formulas. The free variables $X_1, \ldots, X_n$ of the defining formula are called the *distinguished variables* of Q. Such a conjunctive query is usually written as a 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 binary 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 the formula defining a conjunctive query Q has no free variables (i.e., if it is a sentence), then Q is a *Boolean* conjunctive query. For example, the sentence

$$\exists Z_1 \exists Z_2 \exists Z_3 (E(Z_1, Z_2) \wedge E(Z_2, Z_3) \wedge E(Z_3, Z_1))$$

defines the Boolean conjunctive query "is there a cycle of length 3?".

If D is a databaseand Q is a n-ary query, 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 (cf. Chapter **??**). The CONJUNCTIVE-QUERY EVALUATION PROBLEM asks: given a n-ary query Q, a database D, and a n-tuple $\mathbf{a}$ from D, does $\mathbf{a} \in Q(D)$? Let Q_1 and Q_2 be two n-ary queries having the same tuple of distinguished variables. We say that Q_1 is *contained in* Q_2, and write $Q_1 \subseteq Q_2$, if $Q_1(D) \subseteq Q_2(D)$ for every database D. The CONJUNCTIVE-QUERY CONTAINMENT PROBLEM asks: given two conjunctive queries Q_1 and Q_2, is $Q_1 \subseteq Q_2$? These concepts can be defined for Boolean conjunctive queries in an analogous manner. In particular, if Q is a Boolean query and D is a database, then $Q(D) = 1$ if D satisfies Q; otherwise, $Q(D) = 0$. Moreover, the containment problem for Boolean queries Q_1 and Q_2 is equivalent to asking whether Q_1 logically implies Q_2.

It is well known that conjunctive-query containment can be reformulated both as a *conjunctive-query evaluation* problem and as a *homomorphism* problem. What links these problems together is 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 the universe of 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 monadic predicate P_i (not occurring in Q). Every subgoal in the body of Q gives rise to a tuple in the corresponding predicate of D^Q; moreover, if X_i is a distinguished variable of Q, then $P_i(X_i)$ is also a (monadic) tuple of D^Q. Thus, returning to the preceding example, the canonical database of the conjunctive query $\exists Z_1 \exists Z_2 (P(X_1, Z_1, Z_2) \wedge R(Z_2, Z_3) \wedge R(Z_3, X_2))$ consists of the facts $P(X_1, Z_1, Z_2)$, $R(Z_2, Z_3)$, $R(Z_3, X_2)$, $P_1(X_1)$, $P_2(X_2)$. The relationship between conjunctive-query containment, conjunctive-query evaluation, and homomorphisms is provided by the following classical result, due to Chandra and Merlin.

Theorem 1.1. [11] *Let Q_1 and Q_2 be two conjunctive queries having the same tuple $(X_1, \ldots, X_n)$ of distinguished variables. Then the following statements are equivalent.*

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

It follows that the homomorphism problem can be viewed as a conjunctive-query evaluation problem or as a conjunctive-query containment problem. For this, with every structure $\mathbf{A}$, we view the universe $A = \{X_1, \ldots, X_n\}$ of $\mathbf{A}$ as a set of individual variables and associate with $\mathbf{A}$ the Boolean conjunctive query $\exists X_1 \ldots \exists X_n \wedge_{\mathbf{t} \in R^{\mathbf{A}}} R(\mathbf{t})$; we call this query the *canonical conjunctive query* of $\mathbf{A}$ and denote it by $Q_{\mathbf{A}}$. It is clear that $\mathbf{A}$ is isomorphic to the canonical database associated with $Q_{\mathbf{A}}$.

Corollary 1.2. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two structures over the same vocabulary. Then the following statements are equivalent.*

- $\mathbf{A} \rightarrow \mathbf{B}$.
- $\mathbf{B} \models Q_{\mathbf{A}}$.
- $Q_{\mathbf{B}} \subseteq Q_{\mathbf{A}}$.

As an illustration, we have that a graph $\mathbf{G}$ is 3-colorable iff $\mathbf{K}_3 \models Q_{\mathbf{G}}$ iff $Q_{\mathbf{K}_3} \subseteq Q_{\mathbf{G}}$.

A *relational join*, denoted by the symbol $\bowtie$, is a conjunctive query with no existentially quantified variables. Thus, relational-join evaluation is a special case of conjunctive-query evaluation. For example, $E(Z_1, Z_2) \wedge E(Z_2, Z_3) \wedge E(Z_3, Z_1)$ is a relational join that, when evaluated on a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, returns all triples of nodes forming a 3-cycle. There is a well known connection between the traditional AI formulation of constraint satisfaction and relational-join evaluation that we describe next. Suppose we are given a constraint-satisfaction instance $(V, D, \mathcal{C})$. We can assume without loss of generality that in every constraint $(\mathbf{t}, R) \in \mathcal{C}$ the elements in $\mathbf{t}$ are distinct. (Suppose to the contrary that $t_i = t_j$. Then we can delete from R every tuple in which the ith and jth entries disagree, and then project out that j-th column from t and R.) We can thus view every element of V as a relational *attribute*, every tuple of distinct elements of V as a *relational schema*, and every constraint $(\mathbf{t}, R)$ as a relation R over the schema t (cf. [1]). It now follows from the definition of constraint satisfaction that CSP can be viewed as a relational-join evaluation problem.

Proposition 1.3. [6, 42] *A constraint-satisfaction instance* $(V, D, \mathcal{C})$ *is solvable if and only if* $\bowtie_{(\mathbf{t},R)\in\mathcal{C}} R$ *is nonempty.*

Note that Proposition 1.3 is essentially the same as Corollary 1.2. Indeed, the condition $\mathbf{B} \models Q_{\mathbf{A}}$ amounts to the non-emptiness of the relational join obtained from $Q_{\mathbf{A}}$ by dropping all existential quantifiers and using the relations from $\mathbf{B}$ as interpretations of the relational symbols in $Q_{\mathbf{A}}$. Moreover, the homomorphisms from $\mathbf{A}$ to $\mathbf{B}$ are precisely the tuples in the relational join associated with the constraint-satisfaction instance $\mathrm{CSP}(\mathbf{A}, \mathbf{B})$.

1.3 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 [36], 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-Colorability restricted to graphs of bounded treewidth is solvable in polynomial time (see [26]). 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. Using the fact that the Constraint-Satisfaction Problem can be identified with the Homomorphism Problem, we begin by introducing some terminology and notation that will enable us to formalize the concept of an "island of tractability" of constraint satisfaction.

In general, an instance of the Homomorphism Problem consists of two relational structures $\mathbf{A}$ and $\mathbf{B}$. Thus, all restricted cases of this problem can be obtained by imposing restrictions on the input structures $\mathbf{A}$ and $\mathbf{B}$.

Definition 1.4. *Let $\mathcal{A}$, $\mathcal{B}$ be two classes of relational structures. We write* $\mathrm{CSP}(\mathcal{A}, \mathcal{B})$ *to denote the restriction of the* Homomorphism Problem *to input structures from $\mathcal{A}$ and $\mathcal{B}$. In other words,*

$$\mathrm{CSP}(\mathcal{A}, \mathcal{B}) = \{(\mathbf{A}, \mathbf{B}) : \mathbf{A} \in \mathcal{A},\ \mathbf{B} \in \mathcal{B} \text{ and } \mathbf{A} \to \mathbf{B}\}.$$

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

(A more general definition of islands of tractability of constraint satisfaction would consider classes of pairs $(\mathbf{A}, \mathbf{B})$ of structures, cf. [28]; we do not pursue this more general definition here.)

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

Clearly, for each fixed $\mathbf{A}$, the decision problem $\mathrm{CSP}(\{\mathbf{A}\}, \mathit{All})$ can be solved in polynomial time, because, given a structure $\mathbf{B}$, the existence of a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ can be checked by testing all functions h from the universe A of $\mathbf{A}$ to the universe B of $\mathbf{B}$ (the total number of such functions is $|B|^{|A|}$, which is a polynomial number in the size of the structure $\mathbf{B}$ when $\mathbf{A}$ is fixed). Thus, having a singleton structure "on the left' is of little interest.

At the other extreme, however, the situation is quite different, since the computational complexity of $\mathrm{CSP}(\mathit{All}, \{\mathbf{B}\})$ may very well depend on the

particular structure $\mathbf{B}$. Indeed, $\mathrm{CSP}(All, \{\mathbf{K}_3\})$ is NP-complete, because it is the 3-Colorability problem; in contrast, $\mathrm{CSP}(All, \{\mathbf{K}_2\})$ is in P, because it is the 2-Colorability problem. For simplicity, in what follows, for every fixed structure $\mathbf{B}$, we define $\mathrm{CSP}(\mathbf{B}) = \mathrm{CSP}(All, \{\mathbf{B}\})$ and call this the *non-uniform* constraint-satisfaction problem associated with $\mathbf{B}$. For such problems, we refer to $\mathbf{B}$ as the *template*. Thus, the first major goal in the study of the computational complexity of constraint satisfaction is to identify those templates $\mathbf{B}$ for which $\mathrm{CSP}(\mathbf{B})$ is in PTIME. This goals gives rise to an important open decision problem.

The Tractability Classification Problem: Given a relational structure $\mathbf{B}$, decide if $\mathrm{CSP}(\mathbf{B})$ is in PTIME.

In addition to the family of non-uniform constraint-satisfaction problems $\mathrm{CSP}(\mathbf{B})$, where $\mathbf{B}$ is a relational structure, we also study decision problems of the form $\mathrm{CSP}(\mathcal{A}, All)$, where $\mathcal{A}$ is a class of structures. We refer to such problems as *uniform* constraint-satisfaction problems.

It is illuminating to consider the complexity of uniform and non-uniform constraint satisfaction from the perspective of query evaluation. As argued in [67] (see Chapter **??**), there are three ways to measure the complexity of evaluating queries (we focus here on Boolean queries) expressible in a query language L:

- The *combined complexity* of L is the complexity of the following decision problem: given an L-query Q and a structure $\mathbf{A}$, does $\mathbf{A} \models Q$? In symbols,

$$\{\langle Q, \mathbf{A}\rangle : Q \in L \text{ and } \mathbf{A} \models Q\}.$$

- The *expression complexity* of L is the complexity of the following decision problems, one for each fixed structure $\mathbf{A}$:

$$\{Q : Q \in L \text{ and } \mathbf{A} \models Q\}.$$

- The *data complexity* of L is the complexity of the following decision problems, one for each fixed query $Q \in L$:

$$\{\mathbf{A} : \mathbf{A} \models Q\}.$$

As discussed in Chapter **??**, the data complexity of first-order logic is in LOGSPACE, which means that, for each first-order query Q, the problem $\{\mathbf{A} : \mathbf{A} \models Q\}$ is in LOGSPACE. In contrast, the combined complexity for first-order logic is PSPACE-complete. Furthermore, the expression complexity for first-order logic is also PSPACE-complete. In fact, for all but trivial structures $\mathbf{A}$, the problem $\{Q : Q \in FO \text{ and } \mathbf{A} \models Q\}$ is PSPACE-complete. This exponential gap between data complexity, on one hand, and combined and expression complexity, on the other hand, is typical [67]. For conjunctive queries, on the other hand, both combined and expression complexity are NP-complete.

Consider now the uniform constraint-satisfaction problem $\mathrm{CSP}(\mathcal{A}, All) = \{(\mathbf{A}, \mathbf{B}) : \mathbf{A} \in \mathcal{A}, \text{ and } \mathbf{A} \to \mathbf{B}\}$, where $\mathcal{A}$ is a class of structures. By Corollary 1.2, we have that

$$\mathrm{CSP}(\mathcal{A}, All) = \{(\mathbf{A}, \mathbf{B}) : \mathbf{A} \in \mathcal{A}, \ \mathbf{B} \text{ is a structure and } \ \mathbf{B} \models Q_{\mathbf{A}}\}.$$

Thus, studying the complexity of uniform constraint satisfaction amounts to studying the combined complexity for a class of conjunctive queries, as, for example, in [12, 39, 62]. In contrast, consider the non-uniform constraint-satisfaction problem $\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \to \mathbf{B}\}$. By Corollary 1.2 we have that $\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{B} \models Q_{\mathbf{A}}\}$. Thus, studying the complexity of non-uniform constraint satisfaction amounts to studying the expression complexity of conjunctive queries with respect to different structures. This is a problem that has not been studied in the context of database theory.

1.4 Non-Uniform Constraint Satisfaction

The first major result in the study of non-uniform constraint-satisfaction problems was obtained by Schaefer [63], 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 $\mathbf{B} = (\{0, 1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$. A *Boolean non-uniform constraint-satisfaction problem* is a problem of the form $\mathrm{CSP}(\mathbf{B})$ with a Boolean template $\mathbf{B}$. 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 [36]. 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, as seen earlier, 3-Sat is $\mathrm{CSP}(\mathbf{B})$, where $\mathbf{B} = (\{0, 1\}, R_0, R_1, R_2, R_3)$ and R_i is the set of all triples 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 $\mathrm{CSP}(\mathbf{B})$, where $\mathbf{B} = (\{0, 1\}, R_0, R_3)$.

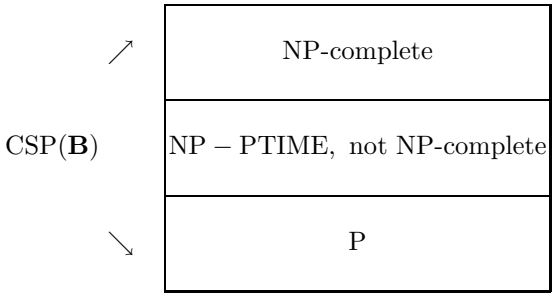
Ladner [51] showed that if PTIME $\neq$ NP, then there are decision problems in NP that are neither NP-complete, nor belong to PTIME. Such problems are called *intermediate* problems. Consequently, it is conceivable that a given family of NP-problems contains intermediate problems. Schaefer [63], however, showed that the family of all Boolean non-uniform constraint-satisfaction problems contains no intermediate problems.

Theorem 1.5. (Schaefer's Dichotomy Theorem [63])

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

- *The* Tractability Classification Problem *for Boolean structures is decidable; in fact, there is a polynomial-time algorithm to decide, given a Boolean structure* $\mathbf{B}$*, whether* CSP($\mathbf{B}$) *is in* PTIME *or is* NP*-complete.*

Schaefer's Dichotomy Theorem can be described pictorially as follows:



Schaefer [63] actually showed that there are exactly six types of Boolean structures such that CSP($\mathbf{B}$) is in PTIME, and provided explicit descriptions of them. Specifically, he showed that CSP($\mathbf{B}$) is in PTIME precisely when at least one of the following six conditions is satisfied:

- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is 0-*valid*, that is, $R_i^{\mathbf{B}}$ contains the all-zeroes tuple $(0, \ldots, 0)$.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is 1-*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 $\mathbf{B}$ is *bijunctive*, that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some 2-CNF formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *Horn*, that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some Horn formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *dual Horn*, that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some dual Horn formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *affine*, that is, $R_i^{\mathbf{B}}$ is the set of solutions to a system of linear equations over the two-element field.

Schaefer's Dichotomy Theorem established a dichotomy and a decidable classification of the complexity of CSP($\mathbf{B}$) for Boolean templates $\mathbf{B}$. After this, Hell and Nešetřil [43] established a dichotomy theorem for CSP($\mathbf{B}$) problems in which the template $\mathbf{B}$ is an *undirected* graph: if $\mathbf{B}$ is bipartite, then CSP($\mathbf{B}$) is solvable in polynomial time; otherwise, CSP($\mathbf{B}$) is NP-complete. To illustrate this dichotomy theorem, let $\mathbf{C}_n$, $n \geq 3$, be a cycle with n elements. Then CSP($\mathbf{C}_n$) is in PTIME if n is even, and is NP-complete if n is odd.

The preceding two dichotomy results raise the challenge of classifying the computational complexity of CSP($\mathbf{B}$) for arbitrary relational templates $\mathbf{B}$. Addressing this question, Feder and Vardi [29] formulated the following conjecture.

Conjecture 1.6. (Dichotomy Conjecture) [29]
If $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ is an arbitrary relational structure, then either CSP($\mathbf{B}$) is in PTIME or CSP($\mathbf{B}$) is NP-complete.

In other words, the Dichotomy Conjecture says that the picture above describes the complexity of non-uniform constraint-satisfaction problems CSP($\mathbf{B}$) for arbitrary structures $\mathbf{B}$. The basis for the conjecture is not only the evidence from Boolean constraint satisfaction and undirected constraint satisfaction, but also from the seeming inability to carry out the diagonalization argument of [51] using the constraint-satisfaction machinery [27].

The Dichotomy Conjecture inspired intensive research efforts that significantly advanced our understanding of the complexity of non-uniform constraint satisfaction. In particular, Bulatov confirmed two important cases of this conjecture. We say that a structure $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ is a *3-element* structure if B contains at most three element. We say that $\mathbf{B}$ is *conservative* if all possible monadic relations on the universe included, that is, every non-empty subset of B is one of the relations $R_i^{\mathbf{B}}$ of $\mathbf{B}$.

Theorem 1.7. [8, 9] *If* $\mathbf{B}$ *a 3-element structure or a conservative structure, then either* CSP($\mathbf{B}$) *is in* PTIME *or* CSP($\mathbf{B}$) *is* NP*-complete. Moreover, in both cases the* Tractability Classification Problem *is decidable in polynomial time.*

In spite of the progress made, the Dichotomy Conjecture remains unresolved in general. The research efforts towards this conjecture, however, have also resulted into the discovery of broad sufficient conditions for tractability and intractability of non-uniform constraint satisfaction that have provided unifying explanations for numerous seemingly disparate tractability and intractability results and have also led to the discovery of new islands of tractability of CSP($\mathbf{B}$). These broad sufficient conditions are based on concepts and techniques from two different areas: universal algebra and logic.

The approach via universal algebra yields sufficient conditions for tractability of CSP($\mathbf{B}$) in terms of *closure* properties of the relations in $\mathbf{B}$ under certain functions on its universe B. Let R be a n-ary relation on a set B and $f : B^k \to B$ a k-ary function. We say that R *is closed under* f, if whenever $\mathbf{t}_1 = (t_1^1, t_1^2, \ldots, t_1^n), \ldots, \mathbf{t_k} = (t_k^1, t_k^2, \ldots, t_k^n)$ are k (not necessarily distinct) tuples in R, then the tuple

$$(f(t_1^1, \ldots, t_k^1), f(t_1^2, \ldots, t_k^2), \ldots, f(t_1^n, \ldots, t_k^n))$$

is also in R. We say that $f : B^k \to B$ is a *polymorphism* of a structure $\mathbf{B} = (B, R_1, \ldots, R_m)$ if each of the relations R_j, $1 \leq j \leq m$, is closed under f. It is easy to see that f is a polymorphism of $\mathbf{B}$ if and only if f is a homomorphism from $\mathbf{B}^k$ to $\mathbf{B}$, where $\mathbf{B}^k$ is the k-th *power* of $\mathbf{B}$. By definition, the k-th power $\mathbf{B}^k$ is the structure $(B^k, R'_1 \ldots, R'_m)$ over the same vocabulary as $\mathbf{B}$ with universe B^k and relations R'_j, $1 \leq j \leq m$, defined as follows: if R_j is of arity n, then $R'_j(\mathbf{s}_1, \ldots, \mathbf{s}_n)$ holds in $\mathbf{B}^k$ if and only if $R_j(s_1^i, \ldots, s_n^i)$ holds in $\mathbf{B}$ for $1 \leq i \leq n$.

We write Pol($\mathbf{B}$) for the set of all polymorphisms of $\mathbf{B}$. As it turns out, the complexity of CSP($\mathbf{B}$) is intimately connected to the kinds of functions

that Pol($\mathbf{B}$) contains. This connection was first unveiled in [29], and explored in depth by Jeavons and his collaborators; for a recent survey see [10]. In particular, they showed that if Pol($\mathbf{B_1}$) = Pol($\mathbf{B_2}$) for two structures $\mathbf{B_1}$ and $\mathbf{B_2}$ (over *finite* vocabularies), then CSP($\mathbf{B_1}$) and CSP($\mathbf{B_2}$) are polynomially reducible to each other. Thus, the polymorphisms of a template $\mathbf{B}$ characterize the complexity of CSP($\mathbf{B}$). The above mentioned dichotomy results for 3-element and conservative constraint satisfaction are based on a rather deep analysis of the appropriate sets of polymorphisms.

1.5 Monotone Monadic SNP and Non-Uniform Constraint Satisfaction

We discussed earlier how non-uniform constraint satisfaction is related to the study of the expression complexity of conjunctive queries. We now show that it can also be viewed as the study of the data complexity of second-order logic. This will suggest a way to identify islands of tractability via logic.

As described in Chapters **??** and **??**, existential second-order logic ESO defines, by Fagin's Theorem, precisely the complexity class NP. The class SNP (for *strict* NP) [46,57] is a fragment of ESO, consisting of all existential second-order sentences with a universal first-order part, namely, sentences of the form $(\exists S')(\forall \mathbf{x})\Phi(\mathbf{x}, S, S')$, where Φ is a first-order quantifier-free formula. We refer to the relations over the input vocabulary S as *input relations*, while the relations over the quantified vocabulary S' are referred to as *existential relations*. 3-SAT is an example of an SNP problem. The input structure consists of four ternary relations C_0, C_1, C_2, C_3, on the universe $\{0, 1\}$, where C_i corresponds to a clause on three variables with the first i of them negated. There is a single existential monadic relation T describing a truth assignment. The condition that must be satisfied states that for all x_1, x_2, x_3, if $C_0(x_1, x_2, x_3)$ then $T(x_1)$ or $T(x_2)$ or $T(x_3)$, and similarly for the remaining C_i by negating $T(x_j)$ if $j \leq i$. Formally, we can express 3-SAT with the SNP sentence:

$$\begin{aligned}(\exists T)(\forall x_1, x_2, x_3)\ ((&C_0(x_1, x_2, x_3) \rightarrow T(x_1) \vee T(x_2) \vee T(x_3)) \wedge \\ &(C_1(x_1, x_2, x_3) \rightarrow \neg T(x_1) \vee T(x_2) \vee T(x_3)) \wedge \\ &(C_2(x_1, x_2, x_3) \rightarrow \neg T(x_1) \vee \neg T(x_2) \vee T(x_3)) \wedge \\ &(C_3(x_1, x_2, x_3) \rightarrow \neg T(x_1) \vee \neg T(x_2) \vee \neg T(x_3))).\end{aligned}$$

It is easy to see that CSP($\mathbf{B}$) is in SNP for each structure $\mathbf{B}$. For each element a in the universe of $\mathbf{B}$, we introduce an existentially quantified monadic relation T_a; intuitively, $T_a(x)$ indicates that a variable x has been assigned value a by the homomorphism. The sentence $\varphi_{\mathbf{B}}$ says that the sets T_a cover all elements in the universe[3], and that the tuples in the input relations satisfy the constraints imposed by the structure $\mathbf{B}$. Thus, if $R(a_1, \ldots, a_n)$ does *not* hold in $\mathbf{B}$, then $\varphi_{\mathbf{B}}$ contains the conjunct $\neg(R(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} T_{a_i}(x_i))$. For

[3] It is not necessary to require disjointness.

example, 3-COLORABILITY over a binary input relation E can be expressed by the following sentence:

$$\begin{aligned}(\exists C_1, C_2, C_3)(\forall x, y)\ ((C_1(x) \vee C_2(x) \vee C_3(x)) \wedge \\ \neg(E(x,y) \wedge C_1(x) \wedge C_1(y)) \wedge \\ \neg(E(x,y) \wedge C_2(x) \wedge C_2(y)) \wedge \\ \neg(E(x,y) \wedge C_3(x) \wedge C_3(y))).\end{aligned}$$

It follows that $\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \models \varphi_{\mathbf{B}}\}$. Thus, the study of the complexity of non-uniform constraint satisfaction can be viewed as the study of the *data complexity* of certain SNP sentences.

A close examination of $\varphi_{\mathbf{B}}$ above shows that it actually resides in a syntactic fragment of SNP. For *monotone* SNP, we require that all occurrences of an input relation C_i in Φ have the same polarity (the polarity of a relation is positive if it is contained in an even number of subformulas with a negation applied to it, and it is negative otherwise); by convention, we assume that this polarity is negative, so that the C_i can be interpreted as constraints, in the sense that imposing C_i on more elements of the input structure can only make the instance "less satisfiable". For *monadic* SNP we require that the existential structure S' consist of monadic relations only. Normally we assume that the language contains also the equality relation, so both equalities and inequalities are allowed in Φ, unless we say *without inequality*, which means that the $\neq$ relation cannot be used (note that equalities can always be eliminated here). We refer to the class when all restrictions hold, that is, monotone monadic SNP without inequality, as MMSNP. It is clear then that non-uniform constraint satisfaction can be expressed in MMSNP.

What is the precise relationship between non-uniform constraint satisfaction and MMSNP? It is easy to see that MMSNP is more expressive than non-uniform constraint satisfaction. The property asserting that the input graph is triangle-free is clearly in MMSNP (in fact, it can be expressed by a universal first-order sentence), but it can be easily shown that there is no graph $\mathbf{G}$ such that $\mathrm{CSP}(\mathbf{G})$ consists of all triangle-free graphs [29]. From a computational point of view, however, MMSNP and non-uniform constraint satisfaction turn out to be equivalent.

Theorem 1.8. [29] *Every problem in MMSNP is polynomially equivalent to* $\mathrm{CSP}(\mathbf{B})$ *for some template* $\mathbf{B}$*. The equivalence is by a randomized Turing reduction*[4] *from CSP to MMSNP and by a deterministic Karp reduction from MMSNP to CSP.*

An immediate corollary is that the Dichotomy Conjecture holds for CSP if and only if it holds for MMSNP. At the same time, MMSNP seems to be a maximal class with this property. Specifically, any attempt to relax the syntactical restrictions of MMSNP yields a class that is polynomially equivalent to NP, and, consequently, a class for which dichotomy fails.

[4] G. Kun recently announced a derandomization of this reduction.

Theorem 1.9. [29]

- *Every problem in NP has a polynomially equivalent problem in monotone monadic SNP with inequality.*
- *Every problem in NP has a polynomially equivalent problem in monadic SNP without inequality.*
- *Every problem in NP has a polynomially equivalent problem in monotone SNP without inequality.*

By Ladner's Theorem it follows that if PTIME $\neq$ NP, then there are intermediate problems, which are neither in PTIME nor NP-complete, in each of monotone monadic SNP with inequality, monadic SNP without inequality, and monotone SNP without inequality. This is the sense in which MMSNP is a maximal class for which we would expect a dichotomy theorem to hold.

The fact that each constraint-satisfaction problem CSP($\mathbf{B}$) can be expressed by the MMSNP sentence $\varphi_{\mathbf{B}}$ suggests a way to identify templates $\mathbf{B}$ for which CSP($\mathbf{B}$) is tractable: characterize those templates $\mathbf{B}$ for which $\varphi_{\mathbf{B}}$ is equivalent to a sentence in a logic whose data complexity is in PTIME. We discuss this approach in the next section.

1.6 Datalog and Non-Uniform Constraint Satisfaction

Consider all tractable problems of the form CSP($\mathbf{B}$). In principle, it is conceivable that every such problem requires a completely different algorithm. In practice, however, there seem to be two basic algorithmic approaches for solving tractable constraint-satisfaction problems: one based on a logical framework and one based on an algebraic framework.[5] Feder and Vardi [29] conjectured that these two algorithmic approaches cover all tractable constraint-satisfaction problems. Their group-theoretic approach, which extended the algorithm used to solve affine Boolean constraint-satisfaction problems [63], has more recently been subsumed by a universal-algebraic approach [8,9]. We discuss here the logical approach.

As described in Chapter **??**, a Datalog program is a finite set of rules of the form t_0 :- $t_1, \ldots, t_m$, 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 fixed-points of monotone operators. 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

[5] The two approaches, however, are not alwasy cleanly separated; in fact, they can be fruitfully combined to yield new tractable classes, cf. [17].

evaluation of the least fixed-point of the program terminates within a polynomial number of steps (in the size of the given EDBs). It follows that Datalog has data complexity in PTIME. Thus, expressibility in Datalog is a sufficient condition for tractability of a query. This suggests trying to identify those templates $\mathbf{B}$ for which the MMSNP sentence $\varphi_{\mathbf{B}}$ is equivalent to a Boolean Datalog query.

It should be noted, however, that Datalog queries are *preserved under homomorphisms*. This means that if $\mathbf{A} \rightarrow^h \mathbf{A}'$ and $\mathbf{t} \in P(\mathbf{A})$ for a Datalog program M, with goal predicate P, then $h(\mathbf{t}) \in P(\mathbf{A}')$. In contrast, constraint-satisfaction problems are not preserved under homomorphism; however, their complements are. If $\mathbf{B}$ is a relational structure, then we write $\overline{\mathrm{CSP}(\mathbf{B})}$ for the *complement* of $\mathrm{CSP}(\mathbf{B})$, that is, the class of all structures $\mathbf{A}$ such that there is no homomorphism $h : \mathbf{A} \rightarrow \mathbf{B}$. If $\mathbf{A} \rightarrow^h \mathbf{A}'$ and $\mathbf{A} \in \mathrm{CSP}(\mathbf{B})$, then it does not follow that $\mathbf{A}' \in \mathrm{CSP}(\mathbf{B})$. On the other hand, if $\mathbf{A} \rightarrow^h \mathbf{A}'$ and $\mathbf{A} \in \overline{\mathrm{CSP}(\mathbf{B})}$, then $\mathbf{A}' \in \overline{\mathrm{CSP}(\mathbf{B})}$, since homomorphisms compose. Thus, rather then try to identify those templates $\mathbf{B}$ for which $\varphi_{\mathbf{B}}$ is equivalent to a Boolean Datalog query, we try to identify those templates $\mathbf{B}$ for which the negated sentence $\neg\varphi_{\mathbf{B}}$ is equivalent to a Boolean Datalog query.

Along this line of investigation, Feder and Vardi [29] provided a unifying explanation for the tractability of many non-uniform $\mathrm{CSP}(\mathbf{B})$ problems by showing that the complement of each of these problems is expressible in Datalog. It should be pointed out, however, that Datalog does not cover all tractable constraint-satisfaction problems. For example, it is shown in [29] that Datalog cannot express the complement of affine Boolean constraint-satisfaction problems; see also [5]. Affine Boolean constraint-satisfaction problems and their generalizations require algebraic techniques to establish their tractability [8, 29]).

For every positive integer k, let k-Datalog be the collection of all Datalog programs in which the body of every rule has at most k distinct variables and also the head of every rule has at most k variables (the variables of the body may be different from the variables of the head). For example, the query Non-2-Colorability is expressible in 3-Datalog, since it is definable by the goal predicate Q of the following Datalog program, which asserts that a cycle of odd length exists:

$$\begin{aligned} P_1(X,Y) &:- E(X,Y) \\ P_0(X,Y) &:- P_1(X,Z), E(Z,Y) \\ P_1(X,Y) &:- P_0(X,Z), E(Z,Y) \\ Q &:- P_1(X,X). \end{aligned}$$

The fact that expressibility in Datalog, and, more specifically, expressibility in k-Datalog provide sufficient conditions for tractability, gives rise to two classification problems:

- The k-Datalog Classification Problem: Given a relational structure $\mathbf{B}$ and $k > 1$, decide if $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog?

- The Datalog Classification Problem: Given a relational structure $\mathbf{B}$, decide if $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog for some $k > 1$.

The universal-algebraic approach does offer some sufficient conditions for $\overline{\mathrm{CSP}(\mathbf{B})}$ to be expressible in Datalog. We mention here two examples. A k-ary function $f : B^k \to B$ with $k \geq 3$ is a *near-unanimity function* if $f(a_1, \ldots, a_k) = b$, for every k-tuple $(a_1, \ldots, a_k)$ such that at least $k - 1$ of the a_i's are equal to b. Note that the ternary majority function from $\{0, 1\}^3$ to $\{0, 1\}$ is a near-unanimity function.

Theorem 1.10. [29] *Let* $\mathbf{B}$ *be relational structure, and* $k \geq 3$. *If* $\mathrm{Pol}(\mathbf{B})$ *contains a* k*-ary near-unanimity function, then* $\overline{\mathrm{CSP}(\mathbf{B})}$ *is expressible in* k*-Datalog.*

Since the number of k-ary functions over the universe B of $\mathbf{B}$ is finite, checking the condition of the preceding theorem for a given k is clearly decidable. It is not known, however, whether it is decidable to check, given $\mathbf{B}$, if $\mathrm{Pol}(\mathbf{B})$ contains a k-ary near-unanimity function for *some* k.

A special class of Datalog consists of those programs whose IDB predicates are all monadic. We refer to such Datalog programs as *monadic* Datalog programs. It can easily be seen that the Horn case of Boolean constraint satisfaction can be dealt with by monadic programs. Consider, for example, a Boolean template with three relations: H_1 is a monadic relation corresponding to positive Horn clauses ("facts"), H_2 is a ternary relation corresponding to Horn clauses of the form $p \wedge q \to r$, and H_3 is a ternary relation corresponding to negative Horn clause of the form $\neg p \vee \neg q \vee \neg r$. Then, unsatisfiability of Horn formulas with at most three literals per clause is expressed by the following monadic Datalog program:

$$\begin{aligned} H(X) &:- H_1(X) \\ H(X) &:- H(X), H_2(Y, Z, X) \\ Q &:- H(X), H(Y), H(Z), H_2(X, Y, Z) \end{aligned}$$

It turns out that we can fully characterize expressibility in monadic Datalog. A k-ary function f is a *set function* if $f(a_1, \ldots, a_k) = f(b_1, \ldots, b_k)$ whenever $\{a_1, \ldots, a_k\} = \{b_1, \ldots, b_k\}$. In other words, a set function depends only the set of its arguments. As a concrete example, the binary Boolean functions $\wedge$ and $\vee$ are set functions.

Theorem 1.11. [29] *Let* $\mathbf{B}$ *be relational structure with universe* B. *Then the following two statements are equivalent.*

- $\overline{\mathrm{CSP}(\mathbf{B})}$ *is expressible in monadic Datalog.*
- $\mathrm{Pol}(\mathbf{B})$ *contains a* $|B|$*-ary set function.*

Since the number of $|B|$-ary functions over the universe B of $\mathbf{B}$ is finite, checking the condition of the theorem is clearly decidable; in fact, it is in

NEXPTIME. Thus, the classification problem for monadic Datalog is decidable.

The main reason for the focus on Datalog as a language to solve constraint-satisfaction problems is that its data complexity is in PTIME. Datalog, however, is not the only logic with this property. We know, for example, that the data complexity of first-order logic is in LOGSPACE. Thus, it would be interesting to characterize the templates $\mathbf{B}$ such that CSP($\mathbf{B}$) is expressible in first-order logic. This turns out to have an intimate connection to expressibility in (non-recursive) Datalog.

Theorem 1.12. [5, 60] *Let* $\mathbf{B}$ *be a relational structure. The following are equivalent:*

- $\overline{\text{CSP}(\mathbf{B})}$ *is expressible in first-order logic.*
- $\overline{\text{CSP}(\mathbf{B})}$ *is expressible by a finite union of conjunctive queries.*

It is known that a Datalog program is always equivalent to a (possibly infinite) union of conjunctive queries. A Datalog program is *bounded* if it is equivalent to a finite union of conjunctive queries [35]. It is known that a Datalog program is bounded if and only if it is equivalent to a first-order formula [2, 61]. Thus, expressibility of non-uniform CSP in first-order logic is a special case of expressibility in Datalog. Concerning the classification problem, Larose, Loten, and Tardif [52] have shown that there is an algorithm to decide, given a structure $\mathbf{B}$, whether CSP($\mathbf{B}$) is expressible in first-order logic; actually, this problem turns out to be NP-complete.

In another direction, we may ask if there are constraint-satisfaction problems that cannot be expressed by Datalog, but can be expressed in least fixed-point logic LFP, whose data complexity is also in PTIME. This is an open question. It is conjectured in [29] that if $\overline{\text{CSP}(\mathbf{B})}$ is expressible in LFP, then it is also expressible in Datalog.

1.7 Datalog, Games, and Constraint Satisfaction

So far, we focused on using Datalog to obtain tractability for non-uniform constraint satisfaction. Kolaitis and Vardi [48] showed how the logical framework also provides a unifying explanation for the uniform tractability of constraint-satisfaction problems. Note that, in general, non-uniform tractability results do not uniformize. Thus, tractability results for each problem in a collection of non-uniform CSP($\mathbf{B}$) problems do not necessarily yield a tractable case of the uniform constraint-satisfaction problem. The reason is that both structures $\mathbf{A}$ and $\mathbf{B}$ are part of the input to the constraint-satisfaction problem, and the running times of the polynomial-time algorithms for CSP($\mathbf{B}$) may very well be exponential in the size of $\mathbf{B}$. We now leverage the intimate connection between Datalog and pebble games to shed new light on expressibility in Datalog, and show how tractability via k-Datalog does uniformize.

As discussed in Chapter **??**, Datalog can be viewed as a fragment of least fixed-point logic LFP; furthermore, on the class *All* of all finite structures, LFP is subsumed by the finite-variable infinitary logic $\mathcal{L}^{\omega}_{\infty\omega} = \bigcup_{k>0} \mathcal{L}^{k}_{\infty\omega}$ (see Chapter **??**). Here we are interested in the existential positive fragments of $\exists\mathcal{L}^{k}_{\infty\omega}$, k a positive integer, which are tailored for the study of Datalog

Theorem 1.13. [48] *Let k be a positive integer. Every k-*Datalog *query over finite structures is expressible in $\exists\mathcal{L}^{k}_{\infty\omega}$. Thus, k-*Datalog $\subseteq \exists\mathcal{L}^{k}_{\infty\omega}$ *on finite structures.*

We make use here of the $(\exists, k)$-pebble games discussed in Chapter **??**. We saw there that if k is a positive integer and Q a Boolean query on a class $\mathcal{C}$ of finite structures, then Q is expressible in $\exists\mathcal{L}^{k}_{\infty\omega}$ on $\mathcal{C}$ iff for all $\mathbf{A}, \mathbf{B} \in \mathcal{C}$ such that $\mathbf{A} \models Q$ and the Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$, we have that $\mathbf{B} \models Q$. The next theorem establishes a connection between expressibility in k-Datalog and $(\exists, k)$-pebble games. (A closely related, but somewhat less precise, such connection was established in [29]). In what follows, if $\mathcal{A}$ is a class of structures and $\mathbf{B}$ is a structure, we write $\mathrm{CSP}(\mathcal{A}, \mathbf{B})$ to denote the class of structures $\mathbf{A}$ such that $\mathbf{A} \in \mathcal{A}$ and $\mathbf{A} \to \mathbf{B}$.

Theorem 1.14. [48] *Let k be a positive integer, $\mathbf{B}$ a relational structure, and $\mathcal{A}$ a class of relational structures such that $\mathbf{B} \in \mathcal{A}$. Then the following statements are equivalent.*

1. $\overline{\mathrm{CSP}(\mathcal{A}, \mathbf{B})}$ *is expressible in k-*Datalog *on $\mathcal{A}$.*
2. $\overline{\mathrm{CSP}(\mathcal{A}, \mathbf{B})}$ *is expressible in $\exists\mathcal{L}^{k}_{\infty\omega}$ on $\mathcal{A}$.*
3. $\overline{\mathrm{CSP}(\mathcal{A}, \mathbf{B})}$ *is equal to the class*

$$\{\mathbf{A} \in \mathcal{A} : \textit{The Spoiler wins the } (\exists, k)\textit{-pebble game on } \mathbf{A} \textit{ and } \mathbf{B}\}.$$

Recall also from Chapter **??** that the query "Given two structures $\mathbf{A}$ and $\mathbf{B}$, does the Spoiler win the $(\exists, k)$-pebble on $\mathbf{A}$ and $\mathbf{B}$?" is definable in LFP; as a result, there is a polynomial-time (in fact, $O(n^{2k})$) algorithm that, given two structures $\mathbf{A}$ and $\mathbf{B}$, determines whether the Spoiler wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$.

By combining Theorem 1.14 with the results of Chapter **??**, we obtain the following uniform tractability result for classes of constraint-satisfaction problems expressible in Datalog.

Theorem 1.15. [48] *Let k be a positive integer, $\mathcal{A}$ a class of relational structures, and $\mathcal{B} = \{\mathbf{B} \in \mathcal{A} : \neg\mathrm{CSP}(\mathcal{A}, \mathbf{B})$ is expressible in k-*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 $\mathbf{A}$ and $\mathbf{B}$.*

Intuitively, if we consider the class of all templates $\mathbf{B}$ for which k-Datalog solves $\mathrm{CSP}(\mathbf{B})$, then computing the winner in the existential k-pebble game

offers a *uniform* polynomial-time algorithm. That is, the algorithm determining the winner in the existential k-pebble game is a uniform algorithm for all (non-uniform) constraint-satisfaction problems that can be expressed in k-Datalog.

The characterization in terms of pebble games turns also sheds light on non-uniform constraint satisfaction. As described in Chapter **??**, for every relational structure $\mathbf{B}$ and every positive integer k, there is a k-Datalog program $\rho^k_{\mathbf{B}}$ that expresses the query "Given a structure $\mathbf{A}$, does the Spoiler win the $(\exists, k)$ pebble game on $\mathbf{A}$ and $\mathbf{B}$?". As an immediate consequence of this fact, we get that $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog if and only if it is expressible by a *specific* k-Datalog program.

Theorem 1.16. [29, 48] $\overline{\mathrm{CSP}(\mathbf{B})}$ *is expressible in k-Datalog if and only if it is expressible by $\rho^k_{\mathbf{B}}$.*

It follows that $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog if and only if $\neg\varphi_{\mathbf{B}}$ is logically equivalent to $\rho^k_{\mathbf{B}}$, where $\varphi_{\mathbf{B}}$ is the MMSNP sentence expressing $\mathrm{CSP}(\mathbf{B})$. Unfortunately, it is not known if equivalence of complemented MMSNP to Datalog is decidable.

1.8 Games and Consistency

One of the most fruitful approaches to coping with the intractability of constraint satisfaction has been the introduction and use of various *consistency* concepts that make explicit additional constraints implied by the original constraints. The connection between consistency properties and tractability was first described in [31, 32]. In a similar vein, the relationship between *local consistency* and *global consistency* is investigated in [21, 65, 66]. Intuitively, local consistency means that any partial solution on a set of variables can be extended to a partial solution containing an additional variable, whereas global consistency means that any partial solution can be extended to a global solution. Note that if the inputs are such that local consistency implies global consistency, then there is a polynomial-time algorithm for constraint satisfaction; moreover, in this case a solution can be constructed via a backtrack-free search. We now describe this approach from the Datalog perspective. The crucial insight is that the key concept of *strong k-consistency* [21] is equivalent to a property of winning strategies for the Duplicator in the $(\exists, k)$-pebble game. Specifically, an instance of a constraint-satisfaction problem is strongly k-consistent if and only if the family of *all* k-partial homomorphisms f is a winning strategy for the Duplicator in the $(\exists, k)$-pebble game on the two relational structures that represent the given instance.

The connection between pebble games and consistency properties, however, is deeper than just a mere reformulation of the concept of strong k-consistency. Indeed, as mentioned earlier, consistency properties underly the

process of making explicit new constraints that are implied by the original constraints. A key technical step in this approach is the procedure known as "establishing strong k-consistency", which propagates the original constraints, adds implied constraints, and transforms a given instance of a constraint-satisfaction problem to a strongly k-consistent instance with the same solution space [15, 21]. In fact, strong k-consistency can be established if and only if the Duplicator wins the $(\exists, k)$-pebble game. Moreover, whenever strong k-consistency can be established, one method for doing this is to first compute the largest winning strategy for the Duplicator in the $(\exists, k)$-pebble game and then modify the original problem by augmenting it with the constraints expressed by the largest winning strategy; this method gives rise to the least constrained instance that establishes strong k-consistency and, in addition, satisfies a natural *coherence* property. By combining this result with known results concerning the definability of the largest winning strategy, it follows that the algorithm for establishing strong k-consistency in this way (with k fixed) is actually expressible in least fixed-point logic; this strengthens the fact that strong k-consistency can be established in polynomial time, when k is fixed. If we consider non-uniform constraint satisfaction, it follows that for every relational structure $\mathbf{B}$, the complement of CSP($\mathbf{B}$) is expressible by a Datalog program with k variables if and only if CSP($\mathbf{B}$) coincides with the collection of all relational structures $\mathbf{A}$ such that establishing strong k-consistency on $\mathbf{A}$ and $\mathbf{B}$ implies that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

We start the formal treatment by returning first to $(\exists, k)$-pebble games. Recall from Chapter **??** that a winning strategy for the Duplicator in the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$ is a nonempty family of k-partial homomorphisms (that is, partial homomorphisms defined on at most k elements) from $\mathbf{A}$ to $\mathbf{B}$ that is closed under subfunctions and has the forth property up to k. A *configuration* for the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$ is a $2k$-tuple $\mathbf{a}, \mathbf{b}$, where $\mathbf{a} = (a_1, \ldots, a_k)$ and $\mathbf{b} = (b_1, \ldots, b_k)$ are elements of A^k and B^k, respectively, such that if $a_i = a_j$, then $b_i = b_j$; this means that the correspondence $a_i \mapsto b_i$, $1 \leq i \leq k$, is a partial function from A to B, which we denote by $h_{\mathbf{a},\mathbf{b}}$. A *winning configuration* for the Duplicator in the existential k-pebble game on $\mathbf{A}$ and $\mathbf{B}$ is a configuration $\mathbf{a}, \mathbf{b}$ for this game such that $h_{\mathbf{a},\mathbf{b}}$ is a member of some winning strategy for the Duplicator in this game. We denote by $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$ the set of all such configurations. The following results show that expressibility in $\exists\mathcal{L}^k_{\infty\omega}$ can be characterized in terms of the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$.

Proposition 1.17. [49] *If $\mathcal{F}$ and $\mathcal{F}'$ are two winning strategies for the Duplicator in the $(\exists, k)$-pebble game on two structures $\mathbf{A}$ and $\mathbf{B}$, then also the union $\mathcal{F} \cup \mathcal{F}'$ is a winning strategy for the Duplicator. Consequently, there is a largest winning strategy for the Duplicator in the $(\exists, k)$-pebble game, namely the union of all winning strategies, which is precisely the set $\mathcal{H}^k(\mathbf{A}, \mathbf{B}) = \{h_{\overline{a},\overline{b}} : (\overline{a}, \overline{b}) \in \mathcal{W}^k(\mathbf{A}, \mathbf{B})\}$.*

Corollary 1.18. [48] *Let k be a positive integer and Q a k-ary 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 $\mathbf{A}$, $\mathbf{B}$ are two structures in $\mathcal{C}$, $(\mathbf{a}, \mathbf{b}) \in \mathcal{W}^k(\mathbf{A}, \mathbf{B})$, and $\mathbf{A} \models Q(\mathbf{a})$, then $\mathbf{B} \models Q(\mathbf{b})$.*

The following lemma is a crucial definability result.

Lemma 1.19. [48] *There is a positive-in-S first-order formula $\varphi(\overline{x}, \overline{y}, S)$, where $\overline{x}$ and $\overline{y}$ are k-tuples of variables, such that the complement of its least fixed-point on a pair $(\mathbf{A}, \mathbf{B})$ of structures defines the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$ of all winning configurations for the Duplicator in the $(\exists, k)$-pebble game on $\mathbf{A}, \mathbf{B}$.*

We now formally define the concepts of *i-consistency* and *strong k-consistency*.

Definition 1.20. *Let $\mathcal{P} = (V, D, \mathcal{C})$ be a constraint-satisfaction instance.*

- *A* partial solution *on a set $V' \subset V$ is an assignment $h : V' \to D$ that satisfies all the constraints whose scope is contained in V'.*
- *$\mathcal{P}$ is* i-consistent *if for every $i-1$ variables $v_1, \ldots, v_{i-1}$, for every partial solution on these variables, and for every variable $v_i \notin \{v_1, \ldots, v_{i-1}\}$, there is a partial solution on the variables $v_1, \ldots, v_{i-1}, v_i$ extending the given partial solution on the variables $v_1, \ldots, v_{i-1}$.*
- *$\mathcal{P}$ is* strongly k-consistent *if it is i-consistent for every $i \leq k$.*

∎

To illustrate these concepts, consider the Boolean formula

$$(\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3).$$

It is easy to verify that this formula, viewed as a constraint-satisfaction instance, is strongly 3-consistent. For instance, the partial solution $x_2 = 0$, $x_3 = 0$ can be extended to the solution $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, while the partial solution $x_1 = 1$, $x_3 = 1$ can be extended to the solution $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. In contrast, the Boolean formula

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

is satisfiable and strongly 2-consistent, but not 3-consistent (hence, it is not strongly 3-consistent either). The reason is that the partial solution $x_2 = 0$, $x_3 = 0$ cannot be extended to a solution, since the only solutions of this formula are $x_1 = 0$, $x_2 = 1$, $x_3 = 1$ and $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. We note that the concepts of strong 2-consistency and strong 3-consistency were first studied in the literature under the names of *arc consistency* and *path consistency* (see [23]).

A key insight is that the concepts of i-consistency and strong k-consistency can be naturally recast in terms of existential pebble games.

Proposition 1.21. [49] *Let $\mathcal{P}$ be a CSP instance, and let $(\mathbf{A}_\mathcal{P}, \mathbf{B}_\mathcal{P})$ be the associated homomorphism instance.*

- *$\mathcal{P}$ is i-consistent if and only if the family of all partial homomorphisms from $\mathbf{A}_\mathcal{P}$ to $\mathbf{B}_\mathcal{P}$ with $i-1$ elements in their universe has the i-forth property.*
- *$\mathcal{P}$ is strongly k-consistent if and only if the family of all k-partial homomorphisms from $\mathbf{A}_\mathcal{P}$ to $\mathbf{B}_\mathcal{P}$ is a winning strategy for the Duplicator in the $(\exists, k)$-pebble game on $\mathbf{A}_\mathcal{P}$ and $\mathbf{B}_\mathcal{P}$.*

Let us now recall the concept of *establishing strong k-consistency*, as defined, for instance, in [15,21]. This concept has been defined rather informally in the AI literature to mean that, given a constraint-satisfaction instance $\mathcal{P}$, we associate with it another instance $\mathcal{P}'$ that has the following properties: (1) $\mathcal{P}'$ has the same set of variables and the same set of values as $\mathcal{P}$ (2) $\mathcal{P}'$ is strongly k-consistent; (3) $\mathcal{P}'$ is at least as constrained as $\mathcal{P}$; and (4) $\mathcal{P}$ and $\mathcal{P}'$ have the same space of solutions. The next definition formalizes the above concept in the context of the homomorphism problem (cf. [19, 49]).

Definition 1.22. *Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures over a k-ary vocabulary σ (i.e., every relation symbol in σ has arity at most k).* Establishing strong k-consistency for $\mathbf{A}$ and $\mathbf{B}$ *means that we associate two relational structures $\mathbf{A}'$ and $\mathbf{B}'$ with the following properties:*

1. *$\mathbf{A}'$ and $\mathbf{B}'$ are structures over some k-ary vocabulary σ' (in general, different than σ); moreover, the universe of $\mathbf{A}'$ is the universe A of $\mathbf{A}$, and the universe of $\mathbf{B}'$ is the universe B of $\mathbf{B}$.*
2. *$\mathrm{CSP}(\mathbf{A}', \mathbf{B}')$ is strongly k-consistent.*
3. *if h is a k-partial homomorphism from $\mathbf{A}'$ to $\mathbf{B}'$, then h is a k-partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*
4. *If h is a function from A to B, then h is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ if and only if h is a homomorphism from $\mathbf{A}'$ to $\mathbf{B}'$.*

If the structures $\mathbf{A}'$ and $\mathbf{B}'$ have the above properties, then we say that $\mathbf{A}'$ and $\mathbf{B}'$ establish strong k-consistency for $\mathbf{A}$ and $\mathbf{B}$. ∎

A constraint-satisfaction instance $\mathcal{P}$ is *coherent* if every constraint $(\mathbf{t}, R)$ of $\mathcal{P}$ completely determines all constraints $(\mathbf{u}, Q)$ in which all variables occurring in $\mathbf{u}$ are among the variables of $\mathbf{t}$. We formalize this concept as follows.

Definition 1.23. *An instance $\mathbf{A}, \mathbf{B}$ of the homomorphism problem is* coherent *if its associated constraint-satisfaction instance $\mathrm{CSP}(\mathbf{A}, \mathbf{B})$ has the following property: for every constraint $(\mathbf{a}, R)$ of $\mathrm{CSP}(\mathbf{A}, \mathbf{B})$ and every tuple $\mathbf{b} \in R$, the mapping $h_{\mathbf{a},\mathbf{b}}$ is well defined and is a partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$.* ∎

Note that a constraint-satisfaction instance can be made coherent in polynomial-time by constraint propagation.

The main result of this section is that strong k-consistency can be established precisely when the Duplicator wins the $(\exists, k)$-pebble game. Moreover, one method for establishing strong k-consistency is to first compute the largest winning strategy for the Duplicator in this game and then generate an instance of the constraint-satisfaction problem consisting of all the constraints embodied in the largest winning strategy. Furthermore, this method gives rise to the largest coherent instance that establishes strong k-consistency (and, hence, the least constrained such instance).

Theorem 1.24. [49] *Let k be a positive integer, let σ be a k-ary vocabulary, and let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures over σ with universes A and B, respectively. It is possible to establish strong k-consistency for $\mathbf{A}$ and $\mathbf{B}$ if and only if $\mathcal{W}^k(\mathbf{A}, \mathbf{B}) \neq \emptyset$. Furthermore, if $\mathcal{W}^k(\mathbf{A}, \mathbf{B}) \neq \emptyset$, then the following sequence of steps gives rise to two structures $\mathbf{A}'$ and $\mathbf{B}'$ that establish strong k-consistency for $\mathbf{A}$ and $\mathbf{B}$:*

1. *Compute the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$.*
2. *For every $i \leq k$ and for every i-tuple $\mathbf{a} \in A^i$, form the set $R_{\mathbf{a}} = \{\mathbf{b} \in B^i : (\mathbf{a}, \mathbf{b}) \in \mathcal{W}^k(\mathbf{A}, \mathbf{B})\}$.*
3. *Form the constraint-satisfaction instance $\mathcal{P}$ with A as the set of variables, B as the set of values, and $\{(\mathbf{a}, R_{\mathbf{a}}) : \mathbf{a} \in \cup_{i=1}^{k} A^i\}$ as the collection of constraints.*
4. *Let $(\mathbf{A}', \mathbf{B}')$ be the homomorphism instance of $\mathcal{P}$.*

In addition, the structures $\mathbf{A}'$ and $\mathbf{B}'$ obtained above constitute the largest coherent instance establishing strong k-consistency for $\mathbf{A}$ and $\mathbf{B}$, that is, if $(\mathbf{A}'', \mathbf{B}'')$ is another such coherent instance, then for every constraint $(\mathbf{a}, R)$ of $\mathrm{CSP}(\mathbf{A}'', \mathbf{B}'')$, we have that $R \subseteq R_{\mathbf{a}}$.

The key step in the procedure described in Theorem 1.24 is the first step, in which the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$ is computed. The other steps simply "reformat" $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$. From Lemma 1.19 it follows that we can establish strong k-consistency by computing the fixed-point of a monotone first-order formula. We can now relate the concept of strong k-consistency to the results in [29] regarding Datalog and non-uniform CSP.

Theorem 1.25. [49] *Let $\mathbf{B}$ be a relational structure over a vocabulary σ. Then the following two statements are equivalent.*

- *$\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog.*
- *For every structure $\mathbf{A}$ over σ, establishing strong k-consistency for $\mathbf{A}, \mathbf{B}$ implies that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*

Given the fundamental role that the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$ plays here, it is natural to ask about the complexity of computing it. To turn it into a decision problem, we just ask about the non-emptiness of this set.

Theorem 1.26. [45] *The problem* $\{(\mathbf{A}, \mathbf{B}, k) : \mathcal{W}^k(\mathbf{A}, \mathbf{B}) \neq \emptyset\}$*, with* k *encoded in unary, is* EXPTIME*-complete. In words, the following problem is* EXPTIME*-complete: given a positive integer* k *and two structures* $\mathbf{A}$ *and* $\mathbf{B}$*, does the Duplicator win the* $(\exists, k)$*-pebble game on* $\mathbf{A}$ *and* $\mathbf{B}$*?*

This result is rather surprising. After all, the complexity of constraint satisfaction is "only" NP-complete. In contrast, the complexity of establishing strong k-consistency is provably exponential and not in PTIME. This offers an a posteriori justification of the practice of establishing only a "low degree" of consistency, such as *arc consistency* or *path consistency* [3, 23].

1.9 Uniform Constraint Satisfaction and Bounded Treewidth

So far, we focused on the pursuit of islands of tractability of non-uniform constraint satisfaction, that is, islands of the form $\mathrm{CSP}(\mathbf{B}) = \mathrm{CSP}(\mathit{All}, \{\mathbf{B}\})$, where $\mathbf{B}$ is a fixed template. Even when we discussed uniform constraint satisfaction, it was with respect to tractable templates. In this section we focus on uniform constraint satisfaction of the form $\mathrm{CSP}(\mathcal{A}, \mathit{All})$, where $\mathcal{A}$ is a class of structures. The goal is to identify conditions on $\mathcal{A}$ that ensure uniform tractability.

As is well known, many algorithmic problems that are "hard" on arbitrary structures become "easy" on trees. This phenomenon motivated researchers to investigate whether the concept of a tree can be appropriately relaxed while maintaining good computational behavior. As part of their seminal work on graph minors, Robertson and Seymour introduced the concept of *treewidth*, which, intuitively, measures how "tree-like" a structure is; moreover, they showed that graphs of *bounded treewidth* exhibit such good behavior, cf. [59].

Definition 1.27. *A* tree decomposition *of a relational structure* $\mathbf{A}$ *is a labelled tree* T *such that the following conditions hold:*

1. *every node of* T *is labelled by a non-empty subset of the universe* A *of* $\mathbf{A}$*,*
2. *for every relation* R *of* $\mathbf{A}$ *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* $a \in A$*, the set of nodes 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. The* treewidth *of* $\mathbf{A}$*, denoted* $\mathrm{tw}(\mathbf{A})$*, is the smallest positive integer* k *such that* A *has a tree decomposition of width* k*. We write* $\mathcal{T}(k)$ *to denote the class of all structures* $\mathbf{A}$ *such that* $\mathrm{tw}(\mathbf{A}) < k$*.*

Clearly, if $\mathbf{T}$ is a tree, then $\mathrm{tw}(\mathbf{T}) = 1$. Similarly, if $n \geq 3$ and $\mathbf{C}_n$ is the n-element (directed) cycle, then $\mathrm{tw}(\mathbf{C}) = 2$. At the other end of the scale, $\mathrm{tw}(\mathbf{K}_k) = k - 1$, for every $k \geq 2$. Computing the treewidth of a structure is

an intractable problem. Specifically, the following problem is NP-complete [4]: given a graph $\mathbf{H}$ and an integer $k \geq 1$, is $\mathrm{tw}(\mathbf{H}) \leq k$? Nonetheless, Bodlaender [7] showed that for every fixed integer $k \geq 1$, there is a linear-time algorithm such that, given a structure $\mathbf{A}$, it determines whether or not $\mathrm{tw}(\mathbf{A}) < k$. In other words, each class $\mathcal{T}(k)$ is recognizable in polynomial time.

Dechter and Pearl [25] and Freuder [33] showed that the classes of structures of bounded treewidth give rise to large islands of tractability of uniform constraint satisfaction.

Theorem 1.28. [25, 33] *If $k \geq 2$ is a positive integer, then* $\mathrm{CSP}(\mathcal{T}(k), All)$ *is in* PTIME.

The polynomial-time algorithm for $\mathrm{CSP}(\mathcal{T}(k), All)$ in the above theorem is often described as a *bucket-elimination algorithm* [22]. 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 [48], and Dalmau, Kolaitis and Vardi [18] 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 $\mathrm{CSP}(\mathcal{T}(k), All)$ can be explained in purely logical terms. Moreover, it led to the discovery of larger islands of tractability of uniform constraint satisfaction.

Definition 1.29. *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* FO^k*-formulas built using atomic formulas, conjunction, and existential first-order quantification only.*

Intuitively, queries expressible in FO^k and L^k are simply first-order queries and conjunctive queries, respectively, with a bound k on the number of distinct variables (each variable, however, may be reused any number of times).

As an example, it is easy to see that if $\mathbf{C}_n$ is the n-element cycle, $n \geq 3$, then the canonical conjunctive query $Q_{\mathbf{C}_n}$ 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 $\mathrm{tw}(\mathbf{C}_n) = 2$.

The logics FO^k and L^k are referred to as *variable-confined logics* [47]. The complexity of query evaluation for such queries has been studied in [68]. Since in data complexity the queries are fixed, bounding the number of variables does not change data complexity. The change in expression and combined complexity, however, is quite dramatic, as the combined complexity of FO^k has been shown to be in PTIME [68]. (More generally, the exponential gap

between data complexity and expression and combined complexity shrinks when the number of variables is bounded.)

The next result shows that the relationship we just saw in the example between treewidth and number of variables needed to express the canonical conjunctive query of a cycle is not an accident.

Theorem 1.30. [48] *Let $k \geq 2$ be a positive integer. If $\mathbf{A} \in \mathcal{T}(k)$, then the canonical conjunctive query $Q_{\mathbf{A}}$ is expressible in L^k.*

Corollary 1.31. $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ *can be solved in polynomial time by determining, given a structure $\mathbf{A} \in \mathcal{T}(k)$ and an arbitrary structure $\mathbf{B}$, whether $\mathbf{B} \models Q_{\mathbf{A}}$.*

A precise complexity analysis of $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ is provided in [37], where it is shown that the problem is LOGFCL-complete; by definition, LOGCFL is the class of decision problems that are logspace-reducible to a context-free language. Note that, in contrast, the combined complexity of evaluating FO^k-queries, for $k > 3$, is PTIME-complete [68].

Theorem 1.30 can be viewed as a logical recasting of the bucket-elimination algorithm. It derives the tractability of $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ from the fact that the canonical conjunctive query $Q_{\mathbf{A}}$ can be written using at most k variables. Consequently, evaluating this query amounts to solving a join-evaluation problem in which all intermediate relations are of bounded arity. For an investigation of how the ideas underlying Theorem 1.30 can be used to solve practical join-evaluation problems, see [55].

It turns out, however, that we can also approach solving $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ from the perspective of k-Datalog and $(\exists, k)$-pebble games. This is because L^k is a fragment of $\exists\mathcal{L}^k_{\infty\omega}$, whose expressive power, as seen earlier, can be characterized in terms of such games.

Theorem 1.32. [18] *Let $k \geq 2$ be a positive integer.*

- *If $\mathbf{B}$ is an arbitrary, but fixed, structure, then $\mathcal{T}(k) \cap \overline{\mathrm{CSP}(\mathcal{T}(k), \{\mathbf{B}\})}$ is expressible in k-Datalog*[6]*.*
- $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ *can be solved in polynomial time by determining whether, given a structure $\mathbf{A} \in \mathcal{T}(k)$ and an arbitrary structure $\mathbf{B}$, the Duplicator wins the $(\exists, k)$-pebble on $\mathbf{A}$ and $\mathbf{B}$.*

The situation for bounded treewidth structures, as described by Theorem 1.32, should be contrasted with the situation for bounded *cliquewidth* structures [16]. Let $\mathcal{C}(k)$ be the class of structures of cliquewidth bounded by k. It is shown in [16] that $\mathrm{CSP}(\mathcal{C}(k), \{\mathbf{B}\})$ is in PTIME for each structure $\mathbf{B}$. Since, however, complete graphs have bounded cliquewidth, it follows that the Clique problem can be reduced to $\mathrm{CSP}(\mathcal{C}(k), \mathit{All})$, implying NP-hardness of the latter.

[6] The intersection with $\mathcal{T}(k)$ ensures that only structures with treewidth bounded by k are considered.

As a consequence of Theorem 1.32, we see that $\mathrm{CSP}(\mathcal{T}(k), All)$ can be solved in polynomial time using a constraint-propagation algorithm that is quite different from the bucket-elimination algorithm in Theorem 1.28. It should be noted, however, that this requires knowing that we are given an instance $\mathbf{A}, \mathbf{B}$ where $\mathrm{tw}(\mathbf{A}) \leq k$. In contrast, the bucket-elimination algorithm can be used for arbitrary constraint-satisfaction instances (with no tractability guarantee, in general).

The classes $\mathrm{CSP}(\mathcal{T}(k), All)$ enjoy also nice tractability properties from the perspective of *Parameterized Complexity Theory* [26], as they are *fixed-parameter tractable*, and, in a precise technical sense, are maximal with this property under a certain complexity-theoretic assumption (see [41]).

The development so far shows that $\mathcal{T}(k)$ provides an island of tractability for uniform constraint satisfaction. We now show that this island can be expanded.

Definition 1.33. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two relational structures.*

- *We say that* $\mathbf{A}$ *and* $\mathbf{B}$ *are* homomorphically equivalent*, denoted* $\mathbf{A} \sim_h \mathbf{B}$*, if both* $\mathbf{A} \to \mathbf{B}$ *and* $\mathbf{B} \to \mathbf{A}$ *hold.*
- *We say that* $\mathbf{B}$ *is the* core *of* $\mathbf{A}$*, and write* $\mathrm{core}(\mathbf{A}) = \mathbf{B}$*, if* $\mathbf{B}$ *is a substructure of* $\mathbf{A}$*,* $\mathbf{A} \to \mathbf{B}$ *holds, and* $\mathbf{A} \to \mathbf{B}'$ *fails for each proper substructure* $\mathbf{B}'$ *of* $\mathbf{B}$*.*

Clearly, $\mathrm{core}(\mathbf{K}_k) = \mathbf{K}_k$ and $\mathrm{core}(\mathbf{C}_n) = \mathbf{C}_n$. On the other hand, if $\mathbf{H}$ is a 2-colorable graph with at least one edge, then $\mathrm{core}(\mathbf{H}) = \mathbf{K}_2$. It should be noted that cores play an important role in database query processing and optimization (see [11]). The next result shows that they can also be used to characterize when the canonical conjunctive query is definable in L^k.

Theorem 1.34. [18] *Let* $k \geq 2$ *be a positive integer and* $\mathbf{A}$ *a relational structure. Then the following are equivalent:*

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

The tight connection between definability in L^k and the boundedness of the treewidth of the core suggests a way to expand the "island" $\mathcal{T}(k)$.

Definition 1.35. *If* $k \geq 2$ *is a positive integer, then* $\mathcal{H}(\mathcal{T}(k))$ *is the class of relational structures* $\mathbf{A}$ *such that* $\mathrm{core}(\mathbf{A})$ *has treewidth less than* k*.*

It should noted that $\mathcal{T}(k)$ is properly contained in $\mathcal{H}(\mathcal{T}(k))$, for every $k \geq 2$. Indeed, it is known that there are 2-colorable graphs of arbitrarily large treewidth. In particular, *grids* are known to have these properties (see [26]). Yet, these graphs are members of $\mathcal{H}(\mathcal{T}(2))$, since their core is $\mathbf{K}_2$.

Theorem 1.36. [18] *Let* $k \geq 2$ *be a positive integer.*

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

The preceding Theorem 1.36 yields new islands of tractability for uniform constraint satisfaction, which properly subsume the islands of tractability constituted by the classes of structures of bounded treewidth. This expansion of the tractability landscape comes, however, at a certain price. Specifically, as seen earlier, for every fixed $k \geq 2$, there is a polynomial-time algorithm for determining membership in $\mathcal{T}(k)$ [7]. In contrast, it has been shown that, for every fixed $k \geq 2$, determining membership in $\mathcal{H}(\mathcal{T}(k))$ is an NP-complete problem [18]. Thus, these new islands of tractability are, in some sense, "inaccessible".

Since $\mathcal{H}(\mathcal{T}(k))$ 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 1.36 also shows that determining the winner of the $(\exists, k)$-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 $\mathcal{A}$ of relational structures that are larger than the classes $\mathcal{H}(\mathcal{T}(k))$ and $\mathrm{CSP}(\mathcal{A}, All)$ is solvable in polynomial time. A remarkable result by Grohe [40] essentially shows that, if we fix the vocabulary, *no* such classes exist, provided a certain complexity-theoretic hypothesis is true.

Theorem 1.37. [40] *Assume that* FPT $\neq W[1]$*. If* $\mathcal{A}$ *is a recursively enumerable class of relational structures over some fixed vocabulary such that* $\mathrm{CSP}(\mathcal{A}, All)$ *is in* PTIME*, then there is a positive integer* k *such that* $\mathcal{A} \subseteq \mathcal{H}(\mathcal{T}(k))$*.*

The hypothesis FPT $\neq W[1]$ is a statement in *parameterized complexity* that is analogous to the hypothesis PTIME $\neq$ NP, and it is widely accepted as being true (see [26]). In effect, Grohe's Theorem 1.37 is a converse to Theorem 1.36 for fixed vocabularies. Together, these two theorems yield a complete characterization of all islands of tractability of the form $\mathrm{CSP}(\mathcal{A}, All)$, where $\mathcal{A}$ is a class of structures over some fixed vocabulary. Moreover, they reveal that all tractable cases of the form $\mathrm{CSP}(\mathcal{A}, All)$ 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 $\mathrm{CSP}(\mathcal{A}, All)$ can be solved in polynomial time using constraint propagation.

It is important to emphasize that the classes $\mathcal{H}(\mathcal{T}(k))$ are the largest islands of tractability for uniform constraint satisfaction only under the assumption in Theorem 1.37 of a fixed vocabulary. For variable vocabularies, there is

a long line of research, studying the impact of the "topology" of conjunctive queries on the complexity of their evaluation; this line of research goes back to the study of *acyclic* joins in [69], The connection between acyclic joins and acyclic constraints was pointed out in [42]. This is still an active research area. Chekuri and Ramajaran [12] showed that the uniform constraint-satisfaction problem $\mathrm{CSP}(\mathcal{Q}(k), All)$ is solvable in polynomial time, where $\mathcal{Q}(k)$ is the class of structures of *querywidth* k. Gottlob, Leone, and Scarcello [39] define another notion of width, called *hypertree width*. They showed that the querywidth of a structure $\mathbf{A}$ provides a strict upper bound for the hypertree width of $\mathbf{A}$, but that the class $\mathcal{H}(k)$ of structures of hypertree width at most k is polynomially recognizable (unlike the class $\mathcal{Q}(k)$), and that $\mathrm{CSP}(\mathcal{H}(k), All)$ is tractable. For further discussion on the relative merit of various notions of "width", see [38]. This is an active area of research (see [13, 14]).

Acknowledgements: We are grateful to Benoit Larose and Scott Weinstein for helpful comments on a previous draft of this chapter. This work was supported in part by NSF grants CCR-9988322, CCR-0124077, CCR-0311326, ANI-0216467, and a Guggenheim Fellowship. Part of this work was done while the second author was visiting the Isaac Newton Institute for Mathematical Science, as part of a Special Programme on Logic and Algorithms.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. M. Ajtai and Y. Gurevich. Datalog vs first-order logic. *J. Comput. Syst. Sci.*, 49(3):562–588, 1994.
3. K. Apt. *Principles of Constraint Programming*. Cambridge Univ. Press, 2003.
4. S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. of Algebraic and Discrete Methods*, 8:277–284, 1987.
5. A. Atserias. On digraph coloring problems and treewidth duality. In *Proc. 20th IEEE Symp. on Logic in Computer Science*, pages 106–115. IEEE Computer Society, 2005.
6. W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence*, 35:401–413, 1988.
7. 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.
8. A.A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. 43rd Symp. on Foundations of Computer Science*, pages 649–658. IEEE Computer Society, 2002.
9. A.A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proc. 18th IEEE Symp. on Logic in Computer Science*, pages 321–330. IEEE Computer Society, 2003.
10. A.A. Bulatov, P. Jeavons, and A.A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.
11. 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.
12. 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.
13. H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *Proc. 11th Int'l Conf. on Principles and Practice of Constraint Programming*, volume 3709 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2005.

14. D. A. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction and spread cut decomposition. In *Proc. 19th Int'l Joint Conf. on Artificial Intelligence*, pages 72–77, 2005.
15. M.C. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41(1):89–95, 1989.
16. B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33:125–150, 2000.
17. V. Dalmau. Generalized majority-minority operations are tractable. In *Proc. 20th IEEE Symp. on Logic in Computer Science*, pages 438–447, 2005.
18. V. Dalmau, P.G. Kolaitis, and M.Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In P. Van Hentenryck, editor, *Proc. 8th Int'l Conf. on Constraint Programming (CP'02)*, Lecture Notes in Computer Science 2470, pages 310–326. Springer-Verlag, 2002.
19. V. Dalmau and J. Pearson. Closure functions and width 1 problems. In *Proc. 5th Int'l Conf. on Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 1999.
20. R. Dechter. Constraint networks. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 276–185. Wiley, New York, 1992.
21. R. Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–107, May 1992.
22. R. Dechter. Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.
23. R. Dechter. *Constraint Processing*. Morgan Kaufmman, 2003.
24. R. Dechter and I. Meiri. Experimental evaluation of preprocessing algorithms for constraint satisfaction problems. *Artificial Intelligence*, 68:211–241, 1994.
25. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.
26. R.G. Downey and M.R. Fellows. *Parametrized Complexity*. Springer-Verlag, 1999.
27. T. Feder. Constraint satisfaction: A personal perspective. Technical report, Electronic Colloquium on Computational Complexity, 2006. Report TR06-021.
28. T. Feder and D. Ford. Classification of bipartite boolean constraint satisfaction through delta-matroid intersection, 2005.
29. 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.
30. 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.
31. E.C. Freuder. Synthesizing constraint expressions. *Communications of the ACM*, 21(11):958–966, November 1978.
32. E.C. Freuder. A sufficient condition for backtrack-free search. *Journal of the Association for Computing Machinery*, 29(1):24–32, 1982.
33. E.C Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proc. AAAI-90*, pages 4–9, 1990.
34. D.H. Frost. *Algorithms and Heuristics for Constraint Satisfaction Problems*. PhD thesis, Department of Computer Science, University of California, Irvine, 1997.

35. H. Gaifman, H. Mairson, Y. Sagiv, and M. Y. Vardi. Undecidable optimization problems for database logic programs. In *Proc. 2nd IEEE Symp. on Logic in Computer Science*, pages 106–115, 1987.
36. M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., 1979.
37. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. In *Proc. 39th IEEE Symp. on Foundation of Computer Science*, pages 706–715, 1998.
38. G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. In *Proc. 16th Int'l Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 394–399, 1999.
39. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In *Proc. 18th ACM Symp. on Principles of Database Systems*, pages 21–32, 1999.
40. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proc. 44th IEEE Symp. on Foundations of Computer Science*, pages 552–561. IEEE Computer Society, 2003.
41. M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 657–666, 2001.
42. M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposition constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66:57–89, 1994.
43. P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.
44. P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford Lecture Series in Mathematics and Its applications 28. Oxford Univ. Press, 2004.
45. P.G. Kolaitis and J. Panttaja. On the complexity of existential pebble games. In *Proc. 12th Conf. Computer Science Logic*, volume 2803 of *Lecture Notes in Computer Science*, pages 314–329. Springer, 2003.
46. Ph. G. Kolaitis and M. Y. Vardi. The decision problem for the probabilities of higher-order properties. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 425–435, 1987.
47. Ph. G. Kolaitis and M. Y. Vardi. On the expressive power of variable-confined logics. In *Proc. 11th IEEE Symp. on Logic in Computer Science*, pages 348–359, 1996.
48. 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).
49. Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 175–181, 2000.
50. V. Kumar. Algorithms for constraint-satisfaction problems. *AI Magazine*, 13:32–44, 1992.
51. R.E. Ladner. On the structure of polynomial time reducibility. *J. Assoc. Comput. Mach.*, 22:155–171, 1975.
52. B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, pages 201–210, 2006.

53. L. A. Levin. Universal sorting problems. *Problemy Peredaci Informacii*, 9:115–116, 1973. In Russian. English translation in *Problems of Information Transmission* 9:265–266.
54. A.K. Mackworth and E.C. Freuder. The complexity of constraint satisfaction revisited. *Artificial Intelligence*, 59(1-2):57–62, 1993.
55. B.J. McMahan, G. Pan, P. Porter, and M.Y. Vardi. Projection pushing revisited. In *Proc. 9th Int'l Conf. on on Extending Database Technology*, volume 2992 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2004.
56. P. Meseguer. Constraint satisfaction problems: an overview. *AICOM*, 2:3–16, 1989.
57. C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43:425–440, 1991.
58. J. Pearson and P. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway University of London, 1997.
59. N. Robertson and P. D. Seymour. Graph minors IV: Tree-width and well-quasi-ordering. *J. Combinatorial Theory, Ser. B*, 48(2):227–254, 1990.
60. E. Rosen. *Finite Model Theory and Finite Variable Logics*. PhD thesis, University of Pennsylvania, 1995.
61. B. Rossman. Existential positive types and preservation under homomorphisms. In *Proc. 20th IEEE Symp. on Logic in Computer Science*, pages 467–476. IEEE Computer Society, 2005.
62. Y. Saraiya. *Subtree elimination algorithms in deductive databases*. PhD thesis, Department of Computer Science, Stanford University, 1991.
63. T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th ACM Symp. on Theory of Computing*, pages 216–226, 1978.
64. E.P.K. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.
65. P. van Beek. On the inherent tightness of local consistency in constraint networks. In *Proc. of National Conference on Artificial Intelligence (AAAI-94)*, pages 368–373, 1994.
66. P. van Beek and R. Dechter. Constraint tightness and looseness versus local and global consistency. *Journal of the ACM*, 44(4):549–566, 1997.
67. M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 137–146, 1982.
68. M.Y. Vardi. On the complexity of bounded-variable queries. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 266–76, 1995.
69. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. 7 Int'l Conf. on Very Large Data Bases*, pages 82–94, 1981.