The complexity of constraint satisfaction: an algebraic approach

Andrei KROKHIN

Department of Computer Science
University of Warwick
Coventry CV4 7AL
UK

Andrei BULATOV

Computing Laboratory
University of Oxford
Oxford OX1 3QD
UK

Peter JEAVONS

Computing Laboratory
University of Oxford
Oxford OX1 3QD
UK

Notes taken by Alexander SEMIGRODSKIKH

Abstract

Many computational problems arising in artificial intelligence, computer science and elsewhere can be represented as constraint satisfaction and optimization problems. In this survey paper we discuss an algebraic approach that has proved to be very successful in studying the complexity of constraint problems.

1 Constraint satisfaction problems

The constraint satisfaction problem (CSP) is a powerful general framework in which a variety of combinatorial problems can be expressed [21, 60, 62, 79]. The aim in a constraint satisfaction problem is to find an assignment of values to the variables, subject to specified constraints. In artificial intelligence, this framework is widely acknowledged as a convenient and efficient way of modelling and solving a number of real-world problems such as planning [48] and scheduling [76], frequency assignment problems [28], image processing [64], programming language analysis [66] and natural language understanding [2]. In database theory, it has been shown that the key problem of conjunctive-query evaluation can be viewed as a constraint satisfaction problem [36, 54]. Furthermore, some central problems in combinatorial optimization can be represented as constraint problems [21, 31, 43, 50]. Finally, CSPs have attracted much attention in complexity theory because various versions of CSPs lie at the heart of many standard complexity classes, and because, despite their great expressiveness, they tend to avoid "intermediate" complexity; that is, they tend to be either tractable or complete for standard complexity classes [7, 8, 11, 13, 14, 21, 31, 51, 56, 74]. On a more practical side, constraint programming is a rapidly developing area with its own international journal and an annual international conference, and with new programming languages being specifically designed (see, e.g., [62]).

The standard toy example of a problem modelled as a constraint satisfaction problem is the "8-queens" problem: place eight queens on a chess board so that no queen can capture any other one [79]. One can think of the horizontals of the board as variables, and the verticals as the possible values, so that assigning a value to a variable means placing a queen on the corresponding square of the board. The fact that no queen must be able to capture any other queen can be represented as a collection of binary constraints C_{ij}, one for each pair of variables i, j, where the constraint C_{ij} allows only those pairs (k, l) such that a queen at position (i, k) cannot capture a queen at position (j, l). It is easy to see that every solution of this constraint satisfaction problem corresponds to a "legal" placing of the 8 queens.

We now give a formal definition of the general CSP.

1.1 Definition An instance of a constraint satisfaction problem is a triple $(V, D, \mathcal{C})$ where

- V is a finite set of variables,
- D is a set of values (sometimes called a domain), and
- $\mathcal{C}$ is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair $\langle s_i, \varrho_i \rangle$ with s_i a list of variables of length m_i, called *the constraint scope*, and ϱ_i an m_i-ary relation over the set D called *the constraint relation*.

The question is whether there exists a *solution* to $(V, D, \mathcal{C})$, that is, a function from V to D such that, for each constraint in $\mathcal{C}$, the image of the constraint scope is a member of the constraint relation.

Now we give some examples of natural problems and their representations as CSPs.

1.2 Example The most obvious algebraic example of a CSP is the problem of solving a system of equations: given a system of linear equations over a finite field F, does it have a solution? Clearly, in this example each individual equation is a constraint, where the variables in the equation form the scope, and the set of all tuples corresponding to solutions of this equation is the constraint relation.

1.3 Example An instance of the standard propositional 3-*Satisfiability* problem [33, 67] is specified by giving a formula in propositional logic consisting of a conjunction of clauses, each containing three literals (that is, variables or negated variables), and asking whether there are values for the variables which make the formula true.

Suppose that $\Phi = \phi_1 \wedge \cdots \wedge \phi_n$ is such a formula, where the ϕ_i are clauses. The satisfiability question for Φ can be expressed as the instance $(V, \{0, 1\}, \mathcal{C})$ of CSP, where V is the set of all variables appearing in the formula, and $\mathcal{C}$ is the set of constraints $\{\langle s_1, \varrho_1\rangle, \ldots, \langle s_n, \varrho_n\rangle\}$, where each constraint $\langle s_k, \varrho_k\rangle$ is constructed as follows: s_k is a list of the variables appearing in ϕ_k and ϱ_k consists of all tuples that make ϕ_k true. The solutions of this CSP instance are exactly the assignments which make the formula ϕ true. Hence, any instance of 3-*Satisfiability* can be expressed as an instance of CSP.

Example 1.3 suggests that any instance of CSP can be represented in a logical form. Indeed, using the standard correspondence between relations and predicates, one can rewrite an instance of CSP as a first-order formula $\varrho_1(s_1) \wedge \cdots \wedge \varrho_q(s_q)$ where the ϱ_i $(1 \leq i \leq q)$ are predicates on D and $\varrho_i(s_i)$ means ϱ_i applied to the tuple s_i of variables. The question then would be whether this formula is satisfiable [75]. In this paper we will sometimes use this alternative logical form for the CSP. This form is commonly used in database theory because it corresponds so closely to conjunctive query evaluation [54], as the next example indicates.

1.4 Example A *relational database* is a finite collection of *tables*. A table consists of a *scheme* and an *instance*, where

A scheme is a finite set of *attributes*, where each attribute has an associated set of possible values, referred to as a *domain*.

An instance is a finite set of *rows*, where each row is a mapping that associates with each attribute of the scheme a value in its domain.

A standard problem in the context of relational databases is the *Conjunctive Query Evaluation* problem [36, 54]. In this problem we are asked if a *conjunctive query* to a relational database, that is, a query of the form $\varrho_1 \wedge \cdots \wedge \varrho_n$ where the $\varrho_1, \ldots, \varrho_n$ are atomic formulas, has a solution.

A conjunctive query over a relational database corresponds to an instance of CSP by a simple translation of terms: 'attributes' have to be replaced with 'variables', 'tables' with 'constraints', 'scheme' with 'scope', 'instance' with 'constraint relation', and 'rows' with 'tuples'. Hence a conjunctive query is equivalent to a CSP instance whose variables are the variables of the query. For each atomic formula ϱ_i in the query, there is a constraint C such that the scope of C is the list of variables of ϱ_i and the constraint relation of C is the set of models of ϱ_i.

Another important reformulation of the CSP is the *Homomorphism* problem: the question of deciding whether there exists a homomorphism between two relational structures (see [31, 36, 54]). Let $\tau = (R_1, \ldots, R_k)$ be a signature, that is, a list of relation names with a fixed arity assigned to each name. Let $\mathcal{A} = (A; R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}})$ and $\mathcal{B} = (B; R_1^{\mathcal{B}}, \ldots, R_k^{\mathcal{B}})$ be relational structures of signature τ. A mapping $h : A \to B$ is called a *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ if, for all $1 \leq i \leq k$, $(h(a_1), \ldots, h(a_m)) \in R_i^{\mathcal{B}}$ whenever $(a_1, \ldots, a_m) \in R_i^{\mathcal{A}}$. In this case we write $h : \mathcal{A} \to \mathcal{B}$. To see that the *Homomorphism* problem is the same as the CSP, think of the elements in A as variables, the elements in B as values, tuples in the relations of $\mathcal{A}$ as constraint scopes, and the relations of $\mathcal{B}$ as constraint relations. Then, clearly, the solutions to this CSP instance are precisely the homomorphisms from $\mathcal{A}$ to $\mathcal{B}$.

We now give some more examples of well-known combinatorial problems and their representations as a CSP. For the sake of brevity, we use the homomorphism form of the CSP.

1.5 Example For any positive integer k, an instance of the *Graph k-Colorability* problem consists of a graph G. The question is whether the vertices of G can be coloured with k colours in such a way that adjacent vertices receive different colours.

It follows that every instance of *Graph colorability* can be expressed as a CSP instance where $\mathcal{A} = G$ and $\mathcal{B}$ is the complete graph on k vertices, K_k.

1.6 Example An instance of the *Clique* problem consists of an undirected graph G and an integer k. The question is whether G has a clique of size k (that is, a subgraph isomorphic to the complete graph K_k).

It follows that every instance of the *Clique* problem can be expressed as a CSP instance where $\mathcal{A}$ is K_k and $\mathcal{B}$ is the graph G.

1.7 Example An instance of the *Hamiltonian Circuit* problem consists of a graph $G = (V; E)$. The question is whether there is a cyclic ordering of V such that every pair of successive nodes in V is adjacent in G.

It follows that every instance of the *Hamiltonian Circuit* problem can be expressed as a CSP instance with $\mathcal{A} = (V; C_V, \neq_V)$ and $\mathcal{B} = (V; E, \neq_V)$, where $\neq_V$ denotes the inequality relation on V and C_V is the graph of an arbitrary cyclic permutation on V.

1.8 Example An instance of the *Graph Isomorphism* problem consists of two graphs, $G = (V; E)$ and $G' = (V'; E')$, with $|V| = |V'|$. The question is whether there is a bijection between V and V' such that adjacent vertices in G are mapped to adjacent vertices in G', and non-adjacent vertices are mapped to non-adjacent vertices.

It follows that every instance of the *Graph Isomorphism* problem can be expressed as a CSP instance with $\mathcal{A} = (V; E, \overline{E})$ and $\mathcal{B} = (V'; E', \overline{E'})$, where $\overline{E}$ denotes the set of all pairs in $\neq_V$ that are not in E.

Many other examples of well-known problems expressed as CSPs can be found further on in this paper, and also in [43].

2 Related constraint problems

As with many other computational problems, it is not only the standard version of the CSP (that is, deciding whether a CSP instance has a solution or not) which is of interest. There are many related problems that have been studied, and in this section we give a brief overview of some of these.

- **Counting Problem**
  How many solutions does a given CSP instance have?

  A standard natural problem associated with many computational decision problems [21].

- **Quantified Problem**
  Given a fully quantified instance of CSP, is it true?

  Problems of this form have provided several fundamental examples of **PSPACE**-complete problems [21, 23, 75]. Any instance of the ordinary CSP can be viewed as an instance of this problem in which all the quantifiers are existential. We note that certain games involving the construction of graph colorings can easily be expressed in this form [5].

- **Minimal Solution**
  Given a CSP instance and some solution to it, is there a solution that is strictly less (point-wise) than the given one?

  This problem is connected with circumscription, a framework used in artificial intelligence to formalize common-sense reasoning [53]. It was also studied as "minimal model checking" in [52].

- **Circumscriptive Inference**
  Given two CSP instances with the same set of variables, is every minimal solution to the first one also a solution to the second one?

  This is a popular problem in nonmonotonic reasoning, an area of artificial intelligence, related to the previous version of the CSP. It was studied in [51, 52].

- **Equivalence**
  Given two CSP instances, do they have the same sets of solutions?

  In database theory, this corresponds to the question of whether or not two queries are equivalent [6].

- **Isomorphism**
  Given two CSP instances, can one permute the variables in them so that they become equivalent in the above sense?

  This is a more general form of the Equivalence problem which is of interest in some contexts. The complexity of the Boolean case of this problem is classified in [7].

- **Inverse Satisfiability**
  Given a set of n-tuples, is it the set of all solutions to a CSP instance of some certain type?

  This problem is related to efficient knowledge representation issues in artificial intelligence [49].

- **Listing Problem**
  Generate all solutions of a given CSP instance.

  A standard natural problem associated with many computational decision problems [21].

- **Max CSP**
  Maximize the number of satisfied constraints in a CSP instance.

  For over-constrained problems, where it is impossible to satisfy all of the constraints, it may be appropriate to try to find a solution satisfying as many constraints as possible [32]. A number of standard optimization problems, e.g., maximum cut, can also be expressed as Max CSP problems [21, 50].

- **Maximum Solution**
  Maximize the sum of values in a solution of a CSP instance.

  Many optimization problems including maximum clique are of this form; in the Boolean case this problem is known as MAX ONES [21, 50].

- **Maximum Hamming Distance**
  Find two solutions to a CSP instance that are distinct in a maximal number of variables.

  The "world difference" in the blocks world problem from knowledge representation can be modelled in this way [22].

- **Lex Max CSP**
  Given a CSP instance where the variables are linearly ordered, find a solution that is lexicographically maximal.

  This form of CSP is used when variables in instances have priorities according to some preference list [74].

- **Unique Solution**
  Does a given instance of CSP have a unique solution?

  This problem is studied in [47]. A related problem concerning partially unique solutions (that is, solutions that are unique on some subsets of variables) was studied in [56].

3 Parameterization of the CSP

The main object of our interest is the computational complexity of constraint problems of various kinds. We refer the reader to [33, 67] for a general background in complexity theory and the definitions of standard complexity classes. In general, the standard decision-problem form of the CSP is **NP**-complete, as one can see from Example 1.3, so it is unlikely to be computationally tractable. However, certain restrictions on the form of the problems can ensure tractability, that is, solvability in polynomial time (see, e.g., [68]).

With any CSP instance one can associate two natural parameters, which represent, informally, the following two features of the instance: which variables constrain which others, and the way in which the values are constrained.

(1) The first feature (that is, which variables constrain which others) can be captured in two ways: one of these is by giving a hypergraph defined on the set of variables used in the instance, where each hyperedge consists of the set of variables appearing together in some constraint scope. The other, finer, way is by specifying the left-hand-side structure, $\mathcal{A}$, in the homomorphism form of the CSP.

(2) The second feature (that is, the way in which the values are constrained), can be captured by specifying the set of constraint relations used in the instance, or alternatively by specifying the right-hand-side structure, $\mathcal{B}$, in the homomorphism form of the CSP.

It follows from these observations that the general CSP can be restricted by fixing either the set of allowed hypergraphs (or left-hand-side structures) or else the set of allowed constraint relations (or right-hand-side structures).

The case when the set of hypergraphs is fixed has been studied in connection with databases [36, 54]. Moreover, in [37], there is a complete classification of the complexity of the CSP in the case when the set of possible left-hand-side structures is fixed, and there are no restrictions on the right-hand-side structures.

In this paper we concentrate on the case when *the set of constraint relations allowed in instances is fixed*, but there is no restriction on the form of the associated hypergraphs (or left-hand-side structures). Let $R_D^{(n)}$ denote the set of all n-ary relations (or predicates) on a set D, and let $R_D = \bigcup_{n=1}^{\infty} R_D^{(n)}$.

3.1 Definition A *constraint language* over D is a subset Γ of R_D. The *constraint satisfaction problem over* Γ, denoted CSP(Γ), is the subclass of the CSP defined by the following property: any constraint relation in any instance must belong to Γ.

Of course, such a parameterization can also be considered for all of the related constraint problems discussed in Section 2 above.

3.2 Definition A constraint language Γ is called *globally tractable* if CSP(Γ) is tractable, and it is called *tractable* if, for every finite $\Gamma_0 \subseteq \Gamma$, CSP($\Gamma_0$) is tractable. It is called **NP**-*complete* if, for some finite $\Gamma_0 \subseteq \Gamma$, CSP($\Gamma_0$) is **NP**-complete.

Of course, every finite tractable constraint language is also globally tractable, but for infinite constraint languages this implication is not immediate (see [15, 17]), so it is technically necessary to distinguish the notions of tractability and global tractability. In fact, all known tractable constraint languages are globally tractable, and it seems plausible that the two notions coincide, though at present this is an open problem. In this paper, we will consider only the question of determining which constraint languages are tractable, and we will not make any further use of the notion of global tractability.

When the set $\Gamma \subset R_D$ is finite, let $\mathcal{B}_\Gamma$ denote the relational structure over the universe D whose relations are precisely the relations of Γ (listed in some order). Then the problem CSP(Γ) corresponds exactly with the problem Hom($\mathcal{B}_\Gamma$), defined as follows: given a structure $\mathcal{A}$ similar to $\mathcal{B}_\Gamma$ (i.e., of the same signature), is it true that $\mathcal{A} \to \mathcal{B}_\Gamma$? Note that the order in which the relations from Γ are listed in $\mathcal{B}_\Gamma$ does not affect the complexity of this problem.

We now give some examples of well-known problems expressible as CSP(Γ) for suitable sets Γ.

3.3 Example An instance of *Linear Equations* consists of a system of linear equations over a field.

Following Example 1.2, it is easy to see that this problem can be expressed as CSP(Γ) where Γ consists of all relations expressible by a linear equation. This problem is clearly tractable because it can be solved by a straightforward polynomial-time algorithm, such as Gaussian elimination.

Moreover, systems of equations can be considered not only over fields, but also over other algebraic structures. For example, systems of polynomial equations over a (fixed) finite group (that is, equations of the form $a_1 x_1 a_2 \cdots x_n a_{n+1} = b_1 y_1 b_2 \cdots y_m b_{m+1}$ where the a_i's and the b_i's are constants and the x_i's and y_i's are variables) are studied in [34] where it is proved that solving such systems is tractable if the underlying group is Abelian, and is **NP**-complete otherwise. This result is generalised in [65] to solving systems of equations over finite monoids: this problem is tractable if the underlying monoid is a union of groups and commutative; otherwise it is **NP**-complete. A more general setting, when systems of polynomial equations are considered over an arbitrary finite (universal) algebra, is studied in [58], which gives a generalization of the results on groups and monoids mentioned above.

3.4 Example The *Not-All-Equal Satisfiability* problem [33, 75] is a restricted version of the standard 3-*Satisfiability* problem (Example 1.3) which remains **NP**-complete. In this problem the clauses are ternary, and each clause is satisfied by any assignment in which the variables of the clause do not all receive the same truth value.

This problem corresponds to the problem CSP($\{N\}$) where N is the following ternary relation on $\{0,1\}$:

$$N = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}.$$

3.5 Example Let $H = (V,E)$ be a finite graph. An instance of the *Graph H-coloring* problem consists of a finite graph G. The question is whether G can be homomorphically mapped to H.

This problem precisely corresponds to the problem CSP($\{E\}$). If we consider only undirected graphs H, then the complexity of *Graph H-coloring* has been completely characterised [40]: it is tractable if H is bipartite or contains a loop; otherwise it is **NP**-complete. However, if we allow H and G to be directed graphs, then the complexity of *Graph H-coloring* has not yet been fully characterised. Moreover, it was shown in [31] that every problem CSP(Γ) with finite Γ is polynomial-time equivalent to *Graph H-coloring* for some suitable directed graph H.

Following a seminal work by Schaefer in 1978 [75], many researchers have studied the following problem:

3.6 Problem Determine the complexity of a given constraint problem for *all possible* values of the parameter Γ.

Most progress has been made in the Boolean case (that is, when the set of values D is $\{0,1\}$), such problems are sometimes called "generalized satisfiability problems" [33]. Schaefer obtained a complete classification for the standard decision-problem form of the CSP over $\{0,1\}$ [75], which is described in Section 4.3, below. Over the last decade, classifications for many related Boolean constraint problems, including all of the problems mentioned in Section 2, have been completed (see references in Section 2). Some of these classifications are also described in Section 4.3.

Classifying the complexity in the non-Boolean case has proved to be a very difficult task. Three main approaches to this problem have been considered; two of them are based on the homomorphism form of the CSP.

(1) The homomorphism problem for *graphs* has been extensively studied (see, e.g., [39]), and thus one can try to develop some methods of graph theory to apply in the more general context of constraint satisfaction.

(2) The problem Hom($\mathcal{B}$) can be seen as the membership problem for the class of all relational structures $\mathcal{A}$ such that $\mathcal{A} \to \mathcal{B}$, and hence methods of finite model theory can applied to study the definability of this class in various logics (from which one can then derive information about the complexity of the problem [30]).

Elements of these two approaches are present in [24, 26, 31, 54].

In the remainder of this paper, we will discuss the third, *algebraic*, approach to the complexity classification problem. This approach has proved to be the most fruitful so far; it has made it possible to obtain very strong complexity classification results for a wide variety of cases.

4 The finite-valued CSP

In this section we consider the case when the set of possible values for the variables in a constraint satisfaction problem is finite.

4.1 Expressive power of constraint languages

In any CSP instance some of the required relationships between variables are given explicitly in the constraints, whilst others generally arise implicitly from interactions among different constraints. For any instance in CSP(Γ), the explicit constraint relations must be elements of Γ, but there may be implicit restrictions on some subsets of the variables for which the corresponding relations are not elements of Γ, as the next example indicates.

4.1 Example Let Γ be the set containing a single binary relation, χ, over the set $\{0, 1, 2\}$, where χ is defined as follows:

$$\chi = \{(0, 0), (0, 1), (1, 0), (1, 2), (2, 1), (2, 2)\}.$$

One element of CSP(Γ) is the instance

$$\mathcal{P} = (\{v_1, v_2, v_3, v_4\}, \{0, 1, 2\}, \{C_1, C_2, C_3, C_4, C_5\}),$$

where $C_1 = ((v_1, v_2), \chi)$, $C_2 = ((v_1, v_3), \chi)$, $C_3 = ((v_2, v_3), \chi)$, $C_4 = ((v_2, v_4), \chi)$, $C_5 = ((v_3, v_4), \chi)$.

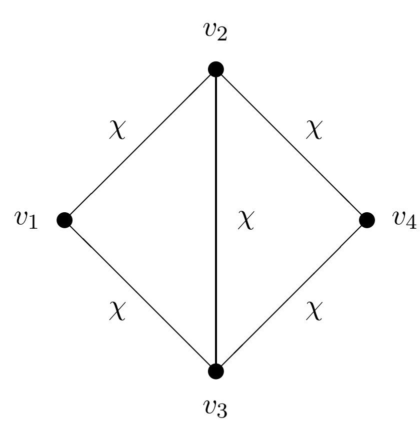

Figure 1: The CSP instance $\mathcal{P}$ defined in Example 4.1.

Note that there is no explicit constraint on the pair (v_1, v_4). However, by considering all solutions to $\mathcal{P}$, it can be shown that the possible pairs of values which can be taken by this pair of variables are precisely the elements of the relation $\chi' = \chi \cup \{(1, 1)\}$.

We now define exactly what it means to say that a constraint relation can be expressed in a constraint language.

4.2 Definition A relation ϱ can be *expressed* in a constraint language Γ over D if there exists a problem instance (V, D, C) in $\mathrm{CSP}(\Gamma)$, and a list, s, of variables, such that the solutions to (V, D, C) restricted to s give precisely the tuples of ϱ.

For any constraint language Γ, the set of all relations which can be expressed in Γ will be called the *expressive power* of Γ.

The expressive power of a constraint language Γ can be characterised in a number of different ways [46]. For example, it is equal to the set of all relations that may be obtained from the relations in Γ using the *relational join* and *project* operations from relational database theory [38]. Alternatively, it can be shown to be equal to the set of relations definable by *primitive positive formulas* involving the relations of Γ and equality, which is defined as follows.

4.3 Definition For any set of relations Γ over D, the set $\langle\Gamma\rangle$ consists of all relations that can be expressed using

(1) relations from Γ, together with the binary equality relation on D (denoted $=_D$),

(2) conjunction, and

(3) existential quantification.

4.4 Example Example 4.1 demonstrates that the relation χ' belongs to the expressive power of the constraint language $\Gamma = \{\chi\}$. It is easy to deduce from the construction given in Example 4.1 that

$$\chi'(x, y) \equiv \exists\, u\ \exists\, v\ (\chi(x, u) \wedge \chi(x, v) \wedge \chi(u, v) \wedge \chi(u, y) \wedge \chi(v, y)).$$

Hence, $\chi' \in \langle\{\chi\}\rangle$.

4.2 Polymorphisms and complexity

In this section we shall explore how the notion of expressive power may be used to simplify the analysis of the complexity of the constraint satisfaction problem.

We first note that any relation that can be expressed in a language Γ can be added to Γ without changing the complexity of $\mathrm{CSP}(\Gamma)$.

4.5 Proposition *For any constraint language* Γ *and any relation* ϱ *belonging to the expressive power of* Γ*,* $\mathrm{CSP}(\Gamma \cup \{\varrho\})$ *is reducible in polynomial time to* $\mathrm{CSP}(\Gamma)$*.*

This result can be established simply by noting that, given an arbitrary problem instance in $\mathrm{CSP}(\Gamma \cup \{\varrho\})$, we can obtain an equivalent instance in $\mathrm{CSP}(\Gamma)$ by replacing each constraint C that has constraint relation ϱ with a collection of constraints that have constraint relations chosen from Γ and that together express the constraint C.

By iterating this procedure we can obtain the following corollary.

4.6 Corollary *For any constraint language* Γ*, and any finite constraint language* Γ_0*, if* Γ_0 *is contained in the expressive power of* Γ*, then* $\mathrm{CSP}(\Gamma_0)$ *is reducible to* $\mathrm{CSP}(\Gamma)$ *in polynomial time.*

Corollary 4.6 implies that for any finite constraint language Γ, the complexity of $\mathrm{CSP}(\Gamma)$ is determined, up to polynomial-time reduction, by the expressive power of Γ, and hence by $\langle\Gamma\rangle$. This raises an obvious question: how can we obtain sufficient information about the set $\langle\Gamma\rangle$ to determine the complexity of $\mathrm{CSP}(\Gamma)$?

A very successful approach to this question has been developed in [18, 43, 45], using techniques from universal algebra [63, 71]. To describe this approach, we need to consider finitary *operations* on D. We will use $O_D^{(n)}$ to denote the set of all n-ary operations on the set D (that is, the set of mappings $f\colon D^n \to D$), and O_D to denote the set $\bigcup_{n=1}^{\infty} O_D^{(n)}$.

An operation $f \in O_D^{(n)}$ will be called *essentially unary* if there exists some i in the range $1 \le i \le n$, and some operation $g \in O_D^{(1)}$ such that the following identity is satisfied

$$f(x_1, x_2, \ldots, x_n) = g(x_i).$$

An essentially unary operation for which g is the identity operation is called a *projection*. Any operation (of whatever arity) which is *not* essentially unary will be called *essentially non-unary*.

Any operation on D can be extended in a standard way to an operation on tuples over D, as follows. For any operation $f \in O_D^{(n)}$, and any collection of tuples $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n \in D^m$, where $\vec{a}_i = (a_{i1}, \ldots, a_{im})$ $(i = 1 \ldots n)$, define $f(\vec{a}_1, \ldots, \vec{a}_n)$ by setting

$$f(\vec{a}_1, \ldots, \vec{a}_n) = (f(a_{11}, \ldots, a_{n1}), \ldots, f(a_{1m}, \ldots, a_{nm})).$$

4.7 Definition For any relation $\varrho \in R_D^{(m)}$, and any operation $f \in O_D^{(n)}$, if $f(\vec{a}_1, \ldots, \vec{a}_n) \in \varrho$ for all choices of $\vec{a}_1, \ldots, \vec{a}_n \in \varrho$, then ϱ is said to be *invariant* under f, and f is called a *polymorphism* of ϱ.

The set of all relations that are invariant under each operation from some set $C \subseteq O_D$ will be denoted $\mathsf{Inv}(C)$. The set of all operations that are polymorphisms of every relation from some set $\Gamma \subseteq R_D$ will be denoted $\mathsf{Pol}(\Gamma)$. The operators Inv and Pol form a Galois correspondence between R_D and O_D (see [71, Proposition 1.1.14]). A basic introduction to this correspondence can be found in [69], and a comprehensive study in [71].

Sets of operations of the form $\mathsf{Pol}(\Gamma)$ are known as *clones* and sets of relations of the form $\mathsf{Inv}(C)$ are known as *relational clones* [71]. Moreover, the following useful characterisation of sets of the form $\mathsf{Inv}(\mathsf{Pol}(\Gamma))$ can be found in [71].

4.8 Theorem *For every set* $\Gamma \subseteq R_D$*,* $\mathsf{Inv}(\mathsf{Pol}(\Gamma)) = \langle\Gamma\rangle$*.*

This result was combined with Corollary 4.6 to obtain the following result in [43].

4.9 Theorem *For any constraint languages* $\Gamma, \Gamma_0 \subseteq R_D$*, with* Γ_0 *finite, if* $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Gamma_0)$*, then* $\mathrm{CSP}(\Gamma_0)$ *is reducible to* $\mathrm{CSP}(\Gamma)$ *in polynomial time.*

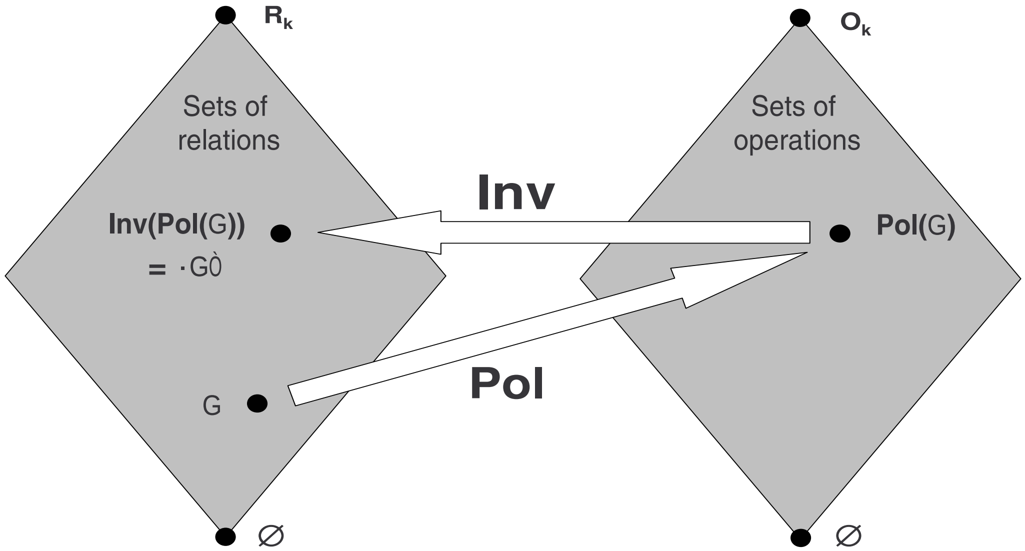

Figure 2: The operators Inv and Pol.

This result implies that, for any finite constraint language Γ over a finite set, the complexity of CSP(Γ) is determined, up to polynomial-time reduction, by the polymorphisms of Γ.

We now apply this result to obtain a sufficient condition for **NP**-completeness of CSP(Γ). A constraint language Γ is said to be *strongly rigid* if $\mathsf{Pol}(\Gamma)$ consists of projections only.

4.10 Proposition *If* Γ *is strongly rigid then* CSP(Γ) *is* **NP***-complete.*

This proposition follows from Theorem 4.9 by setting $\Gamma_0 = \{N\}$ (see Example 3.4), assuming $\{0, 1\} \subseteq D$, and using the fact that every relation on D is invariant under any projection.

Proposition 4.10 was used in [59] to show that most non-trivial problems of the form CSP(Γ), with finite Γ, are **NP**-complete. More precisely, let $R(n, k)$ denote a random k-ary relation on the set $\{1, \ldots, n\}$, for which the probability that $(a_1, \ldots, a_k) \in R(n, k)$ is equal to 1/2 independently for each k-tuple $(a_1, \ldots, a_k)$ where not all a_i's are equal; also, set $(a, \ldots, a) \notin R(n, k)$ for all a (this is necessary to ensure that CSP($R(n, k)$) is non-trivial). It is shown in [59] that the probability that $\{R(n, k)\}$ is strongly rigid tends to 1 as either n or k tends to infinity.

4.3 Complexity of Boolean problems

In this section we describe some of the results that have been obtained concerning the complexity of Boolean constraint problems, that is, problems over a two-valued domain.

The first result of this kind was a complete classification of the complexity of the ordinary Boolean constraint satisfaction problem obtained by Schaefer in 1978 [75]. Recall that a computational problem is called *tractable* if there is a polynomial-time algorithm deciding every instance of the problem. The class of all tractable problems is denoted **PTIME**.

4.11 Theorem *For any constraint language $\Gamma \subseteq R_{\{0,1\}}$, $\mathrm{CSP}(\Gamma)$ is tractable when (at least) one of the following conditions holds:*

(1) *Every ϱ in Γ contains the tuple $(0, 0, \ldots, 0)$.*

(2) *Every ϱ in Γ contains the tuple $(1, 1, \ldots, 1)$.*

(3) *Every ϱ in Γ is definable by a CNF formula in which each conjunct has at most one negated variable.*

(4) *Every ϱ in Γ is definable by a CNF formula in which each conjunct has at most one unnegated variable.*

(5) *Every ϱ in Γ is definable by a CNF formula in which each conjunct has at most two literals.*

(6) *Every ϱ in Γ is definable by a system of linear equations over the two-element field.*

In all other cases $\mathrm{CSP}(\Gamma)$ *is* **NP***-complete.*

This result establishes a dichotomy for versions of this problem parameterized by the choice of constraint language: they are all either tractable or **NP**-complete. Dichotomy theorems of this kind are of particular interest because, on the one hand, they determine the precise complexity of particular constraint problems, and, on the other hand, they demonstrate that no problems of intermediate complexity can occur in this context. Note that the existence of constraint problems of intermediate complexity cannot be ruled out *a priori* due to the result [57] that if $\mathbf{PTIME} \neq \mathbf{NP}$ then the class **NP** contains (infinitely many pairwise inequivalent) problems which are neither tractable nor **NP**-complete.

Using the algebraic approach described in the previous sections, together with the knowledge of possible clones on a two-element set obtained in [72], Schaefer's result can be reformulated in the following much more concise form.

4.12 Theorem *For any set of relations $\Gamma \subseteq R_{\{0,1\}}$, $\mathrm{CSP}(\Gamma)$ is tractable when $\mathsf{Pol}(\Gamma)$ contains any essentially non-unary operation or a constant operation. Otherwise it is* **NP***-complete.*

4.13 Example Recall the relation N over $\{0, 1\}$ defined in Example 3.4. Using general results from [72], it can be shown that $\mathsf{Pol}(\{N\})$ contains essentially unary operations only, and hence, by Theorem 4.12, $\mathrm{CSP}(\{N\})$ is **NP**-complete.

Schaefer's result has inspired a series of analogous investigations for many related constraint problems, including those listed in Section 2. We will now list some complexity classification results that have recently been obtained for these problems in the Boolean case. Surprisingly, for a wide variety of such related problems it turns out that the polymorphisms of the constraint language are highly relevant to the study of the computational complexity.

4.14 Theorem *Let $\Gamma \subseteq R_{\{0,1\}}$ be a Boolean constraint language. The following facts are known to hold for constraint problems parameterized by Γ:*

- *The **Counting Problem** is tractable if $\mathsf{Pol}(\Gamma)$ contains the unique affine operation on $\{0, 1\}$, $x - y + z$. Otherwise it is $\#\mathbf{P}$-complete* [21].

- *The* ***Quantified Problem*** *is tractable if* $\mathsf{Pol}(\Gamma)$ *contains an essentially non-unary operation. Otherwise it is* **PSPACE**-*complete* [21, 23].

- *The* ***Equivalence*** *problem is tractable if* $\mathsf{Pol}(\Gamma)$ *contains an essentially non-unary operation or a constant operation. Otherwise it is* **coNP**-*complete* [6].

- *The* ***Inverse Satisfiability*** *problem is tractable if* $\mathsf{Pol}(\Gamma)$ *contains an essentially non-unary operation. Otherwise it is* **coNP**-*complete* [49].

- *The* ***Maximum Hamming Distance*** *problem is tractable if* $\mathsf{Pol}(\Gamma)$ *contains either a constant operation, or the affine operation and the negation operation on* $\{0, 1\}$ [22].

A full description of these results requires the careful definition of the relevant complexity classes and reductions, which is beyond the scope of this paper, so we refer the reader to the cited papers for details.

4.15 Example Recall the relation N over $\{0, 1\}$ defined in Example 3.4. Using general results from [72], it can be shown that $\mathsf{Pol}(\{N\})$ contains essentially unary operations only. Hence, by Theorem 4.14, we can immediately conclude that:

- Counting the number of solutions to an instance of CSP($\{N\}$) is #**P**-complete;

- Deciding whether a quantified Boolean formula, whose quantifier-free part involves only conjunctions of the predicate N, is true is **PSPACE**-complete.

- Deciding whether two instances of CSP($\{N\}$) have the same solutions is **coNP**-complete;

- Deciding whether a given set of n-tuples is the set of solutions to some instance of CSP($\{N\}$) is **coNP**-complete.

4.4 From the CSP to algebras and varieties

Most of the results presented in this section were first obtained in [15, 17, 18].

With any constraint language $\Gamma \subseteq R_D$ one can associate an algebra $\mathbb{A}_\Gamma = (D; \mathsf{Pol}(\Gamma))$. In this section we show that the complexity of the problem CSP(Γ) is completely determined by certain properties of $\mathbb{A}_\Gamma$. (We refer the reader to [63] for a general background in universal algebra.)

Recall that algebras are said to be *term equivalent* if they have the same set of term operations. Since, the term operations of $\mathbb{A}_\Gamma$ are precisely the operations in $\mathsf{Pol}(\Gamma)$, Theorem 4.9 implies that term equivalent algebras give rise to problem classes of the same complexity.

4.16 Proposition *Let* $\Gamma_1, \Gamma_2 \subseteq R_D$*, where* D *is finite. If* $\mathbb{A}_{\Gamma_1}$ *and* $\mathbb{A}_{\Gamma_2}$ *are term equivalent then* Γ_1 *and* Γ_2 *are tractable or* **NP**-*complete simultaneously.*

This allows us to introduce the notion of a tractable algebra.

4.17 Definition An algebra $\mathbb{A} = (D; F)$ is said to be *tractable* if the constraint language $\mathsf{Inv}(F)$ is tractable. It is said to be **NP**-*complete* if $\mathsf{Inv}(F)$ is **NP**-complete.

Thus, the complexity classification problem for constraint languages reduces to the complexity classification problem for finite algebras. Furthermore, the next results show that it is possible to significantly restrict the class of algebras which need to be classified.

Let $\mathbb{A} = (D; F)$ be an algebra, and $U \subseteq D$. Let $\mathbb{A}|_U$ denote the algebra $\mathbb{A}|_U = (U, F')$, where F' consists of all operations of the form $f|_U$ (the restriction of f to U), for each term operation f of $\mathbb{A}$ such that $f \in \mathsf{Pol}(U)$.

4.18 Proposition *Let $\mathbb{A}$ be a finite algebra, f a unary term operation such that $f(f(x)) = f(x)$ and $U = f(D)$. Then $\mathbb{A}$ is tractable if and only if $\mathbb{A}|_U$ is tractable.*

Hence, by choosing a unary term operation with a minimal range, we may restrict ourselves to considering only *surjective* algebras, that is, algebras all of whose term operations are surjective.

Recall that an operation f is called *idempotent* if it satisfies the identity $f(x, \ldots, x) = x$, and the *full idempotent reduct* of an algebra $\mathbb{A} = (D; F)$ is the algebra $\mathsf{Id}(\mathbb{A}) = (D, F')$ where F' consists of all idempotent term operations of $\mathbb{A}$.

4.19 Proposition *A surjective finite algebra $\mathbb{A}$ is tractable if and only if its full idempotent reduct is tractable.*

It follows that to classify the complexity of arbitrary finite algebras it is sufficient to consider only idempotent algebras, that is, algebras whose operations are all idempotent.

Next, we show that the standard algebraic constructions preserve the tractability of an algebra.

4.20 Theorem *Let $\mathbb{A}$ be a finite algebra. If $\mathbb{A}$ is tractable, then all of its subalgebras, homomorphic images and finite direct powers are also tractable. Conversely, if $\mathbb{A}$ has an* **NP***-complete subalgebra, homomorphic image, or finite direct power, then it is* **NP***-complete itself.*

For an algebra $\mathbb{A}$, we denote the *pseudo-variety* and the *variety* generated by $\mathbb{A}$ by $\mathsf{pvar}(\mathbb{A})$ and $\mathsf{var}(\mathbb{A})$, respectively.

4.21 Corollary *A finite algebra $\mathbb{A}$ is tractable if and only if every algebra from $\mathsf{pvar}(\mathbb{A})$ is tractable.*

As is well known, if $\mathfrak{A}$ is a finite class of finite algebras, then the pseudo-variety generated by $\mathfrak{A}$ equals the class of finite algebras from the variety generated by $\mathfrak{A}$.

4.22 Corollary *A finite algebra $\mathbb{A}$ is tractable if and only if every finite algebra from $\mathsf{var}(\mathbb{A})$ is tractable.*

4.23 Corollary *If $\mathbb{A}$ is a finite algebra, and $\mathsf{var}(\mathbb{A})$ contains a finite* **NP***-complete algebra, then $\mathbb{A}$ is* **NP***-complete.*

Thus, the tractability of an algebra is a property which can be determined by identities.

We call an algebra a *set* if it contains more than one element and all of its operations are projections. By combining Proposition 4.10 with Corollary 4.23, we get the following result.

4.24 Corollary *If the pseudovariety generated by a finite idempotent algebra $\mathbb{A}$ contains a set then $\mathbb{A}$ is* **NP***-complete.*

A homomorphic image of a subalgebra of an algebra $\mathbb{A}$ is called a *factor* of $\mathbb{A}$.

4.25 Proposition *If $\mathbb{A}$ is an idempotent algebra and* $\mathsf{pvar}(\mathbb{A})$ *contains a set then some factor of $\mathbb{A}$ is a set.*

Remarkably, the presence of a set as a factor is the only known reason for an idempotent algebra to be **NP**-complete. This prompts us to suggest the following conjecture.

4.26 Conjecture A finite idempotent algebra $\mathbb{A}$ is tractable if and only if

$$\text{none of the factors of } \mathbb{A} \text{ is a set;} \tag{No-Set}$$

otherwise it is **NP**-complete.

It was shown in [17] that if one weakens Conjecture 4.26 by removing the condition of idempotency, or replacing "factor" by either "subalgebra" or "homomorphic image", then the resulting conjecture is false.

It was proved in [78] that a variety $\mathcal{V}$ generated by a finite idempotent algebra contains no set if and only if there is an n-ary term f (called a *Taylor term*) in $\mathcal{V}$ such that $\mathcal{V}$ satisfies n identities of the form

$$f(x_{i1}, \dots, x_{in}) = f(y_{i1}, \dots, y_{in}), \quad i = 1, \dots, n,$$

where $x_{ij}, y_{ij} \in \{x, y\}$ and $x_{ii} \neq y_{ii}$ for all i, j. Therefore, Corollary 4.24 can be restated as follows.

4.27 Corollary *If a finite idempotent algebra has no Taylor term, then it is* **NP***-complete.*

This corollary was used in [58] to study systems of polynomial equations over finite algebras, where it was proved, in particular, that solving systems of equations over a non-trivial algebra from a congruence-distributive variety is **NP**-complete, and, furthermore, solving systems of equations over a Mal'tsev algebra is tractable if this algebra is polynomially equivalent to a module, otherwise it is **NP**-complete.

Using the result from [78] mentioned above, Conjecture 4.26 can be restated in terms of identities.

4.28 Conjecture A finite idempotent algebra $\mathbb{A}$ is tractable if it has a Taylor term; otherwise it is **NP**-complete.

Finally, the condition (No-Set) from Conjecture 4.26 can be expressed in terms of tame congruence theory [41]: a finite idempotent algebra satisfies this condition if and only if the variety it generates "omits type **1**" [15].

4.5 Tractable algebras, classification results and tractability tests

4.5.1 Tractable algebras

During the last decade several particular identities (particular forms of the Taylor term) have been identified that guarantee the tractability of algebras satisfying one of these identities (that is, having a Taylor term of one of these special forms) [9, 12, 27, 43, 44, 45].

Recall that a binary operation $\cdot$ is called a 2-*semilattice* operation if it satisfies the identities $x \cdot x = x$, $x \cdot y = y \cdot x$ and $(x \cdot x) \cdot y = x \cdot (x \cdot y)$. Note that a semilattice operation is a particular case of a 2-semilattice operation. A ternary operation f satisfying the identities $f(x, y, y) = f(y, y, x) = x$ is called a *Mal'tsev* operation, and an n-ary operation g is called a *near-unanimity* operation if it satisfies the identities

$$f(y, x, \ldots, x) = f(x, y, x, \ldots, x) = \cdots = f(x, \ldots, x, y) = x.$$

An n-ary operation is called *totally symmetric* if, for all $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ such that $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\}$, it satisfies the identities

$$f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n).$$

(Note that, in [27], a family $(f_n)_{n \geq 2}$ of totally symmetric operations, where f_n is n-ary, was called a *set function*).

4.29 Theorem *A finite algebra is tractable if it has (at least) one the following:*

- *a* 2-*semilattice term operation;*
- *a Mal'tsev term operation;*
- *a near-unanimity term operation;*
- n*-ary totally symmetric term operations for all* $n \geq 2$.

Another class of algebras which has been shown to be tractable [25] is the class of *para-primal* algebras, which are defined as follows. Let ϱ an n-ary relation on D, and $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ with $i_1 < \cdots < i_k$. By the *projection* of ϱ onto I we mean the relation $\varrho|_I = \{(a_{i_1}, \ldots, a_{i_k}) \mid (a_1, \ldots, a_n) \in \varrho\}$. The set I is said to be ϱ-*reduced* if it is minimal with the property that the natural mapping $\varrho \mapsto \varrho|_I$ is one-to-one. A finite algebra $\mathbb{A}$ is called *para-primal* if it is finite, and for every $n \in \mathbb{N}$, every subuniverse ϱ of $\mathbb{A}^n$, and every ϱ-reduced set I, we have $\varrho|_I = \prod_{i \in I} \varrho|_{\{i\}}$.

The following result was obtained in [25].

4.30 Theorem *Every para-primal algebra is tractable.*

The para-primal algebras provide a class of tractable algebras that cannot be characterized by identities. (Therefore, this class of algebras is probably contained in a larger class of tractable algebras, but this larger class has not yet been identified.)

4.5.2 Classification results

Algebras of several special types have been completely classified with respect to the complexity of the corresponding constraint satisfaction problems.

Strictly simple algebras A finite algebra is said to be *strictly simple* if it is simple and has no subalgebras with more than one element. Strictly simple algebras are completely described in [77].

4.31 Proposition [17, 18] *A finite idempotent strictly simple algebra is tractable if it is not a set; otherwise it is* **NP***-complete.*

Homogeneous algebras An algebra is called *homogeneous* if every permutation on its base set is an automorphism of the algebra. Finite homogeneous algebras are completely described in [61].

4.32 Proposition [25] *A finite homogeneous algebra is tractable if it satisfies the condition (No-Set); otherwise it is* **NP***-complete.*

Finite semigroups A semigroup is called a *left-* [*right-*]*zero semigroup* if $x \cdot y = x$ [$x \cdot y = y$] for all x, y. It is called a *block-group* if none of its subsemigroups is a left- or right-zero semigroup. As is easily seen, block-groups are exactly those semigroups that have no factor which is a set.

4.33 Proposition [16] *A finite semigroup is tractable if it is a block-group; otherwise it is* **NP***-complete.*

Small algebras Conjecture 4.26 has been proved for 2- and 3-element algebras.

4.34 Theorem [75, 11]

(1) *An idempotent two-element algebra is tractable if it is not a set; otherwise it is* **NP***-complete.*

(2) *An idempotent three-element algebra is tractable if it satisfies the condition (No-Set); otherwise it is* **NP***-complete.*

Conservative algebras An algebra is said to be *conservative* if every subset of its universe is a subalgebra, or, equivalently, if $f(x_1, \ldots, x_n) \in \{x_1, \ldots, x_n\}$ for every term operation f and all $x_1, \ldots, x_n$.

4.35 Theorem [13] *A conservative algebra $\mathbb{A}$ is tractable if every 2-element subalgebra $\mathbb{B}$ has a term operation of one of the following types: a semilattice operation, a ternary near-unanimity operation (that is, a majority operation), or a Mal'tsev operation; otherwise $\mathbb{A}$ is* **NP***-complete.*

It is not hard to check that the conditions stated in Theorem 4.35 are equivalent to (No-Set).

4.5.3 Testing tractability

We now consider the problem of deciding whether a given constraint language or idempotent algebra is tractable. Following [21], we call such a problem a *meta-problem*. A priori, there is no upper complexity bound for this problem; it may even be undecidable. However, if Conjecture 4.26 is true, then, given the basic operations of an idempotent algebra, one can straightforwardly check whether any factor of the algebra is a set. If we are given a finite constraint language Γ on a finite set A, then the presence of a factor which is a set can be detected by examining all polymorphisms of Γ of arity at most $|A|$. Thus, the meta-problem is decidable, assuming Conjecture 4.26 holds. In this section we study its complexity.

For a constraint language Γ, let $f \in \mathsf{Pol}(\Gamma)$ be a unary operation with minimal range U, and let $f(\Gamma) = \{f(\varrho) \mid \varrho \in \Gamma\}$ where $f(\varrho) = \{f(\vec{a}) \mid \vec{a} \in \varrho\}$. Denote by $\overline{\Gamma}$ the constraint language $f(\Gamma) \cup \{\{a\} \mid a \in U\}$. It follows from Propositions 4.18 and 4.19 that Γ is tractable if and only if $\overline{\Gamma}$ is tractable. Moreover, the algebra $\mathbb{A}_{\overline{\Gamma}} = (U, \mathsf{Pol}(\overline{\Gamma})$ is idempotent.

We consider three combinatorial decision problems related to the condition (No-Set).

CSP-Tractability-of-algebra

Instance: A finite set A and operation tables of idempotent operations $f_1, \dots, f_n$ on A.
Question: Does the algebra $\mathbb{A} = (A; \{f_1, \dots, f_n\})$ satisfy (No-set)?

CSP-Tractability

Instance: A finite set A and a finite constraint language Γ on A.
Question: Does the algebra $\mathbb{A} = (U, \mathsf{Pol}(\overline{\Gamma}))$ satisfy (No-set)?

CSP-Tractability(k)

Instance: A finite set A, $|A| \leq k$, and a finite constraint language Γ on A.
Question: Does the algebra $\mathbb{A} = (U, \mathsf{Pol}(\overline{\Gamma}))$ satisfy (No-set)?

4.36 Theorem [15]

(1) *The problem* CSP-Tractability-of-algebra *is tractable.*

(2) *The problem* CSP-Tractability(k) *is tractable.*

(3) *The problem* CSP-Tractability *is* **NP***-complete.*

The algorithm solving *CSP-Tractability-of-algebra* is, in fact, an adapted version of the algorithm presented in [3] for finding the type set of a finite algebra. Since this algorithm uses operations of arity bounded by the size of the algebra, it can be further transformed to an algorithm for solving *CSP-Tractability*(k). Finally, it is possible to show that *CSP-Tractability* is in **NP** and to reduce the **NP**-complete problem *Not-All-Equal-Satisfiability* (see Example 3.4) to *CSP-Tractability* in polynomial time.

4.6 The counting CSP

In this section we discuss Problem 3.6 for the counting constraint satisfaction problem (#CSP), which is the problem of counting solutions to an instance of CSP. Using the logical and the homomorphism forms of the CSP (see Section 1) this problem can also be formulated as the problem of counting satisfying assignments to a conjunctive formula, that is, a formula of the form $\varrho_1 \wedge \cdots \wedge \varrho_n$, where each ϱ_i is an atomic formula, in a given interpretation, or alternatively as the problem of finding the number of homomorphisms between two finite relational structures. For any constraint language Γ, giving rise to the decision constraint satisfaction problem CSP(Γ), we also define the corresponding class #CSP(Γ) of counting problems.

The results of this section were first obtained in [14].

4.37 Example An instance of the *#3-SAT* problem [20, 21, 80, 81] is specified by giving an instance of the 3-*satisfiability* problem (see Example 1.3) and asking how many assignments satisfy it. Therefore, *#3-SAT* is equivalent to #CSP(Γ) where Γ is the set of ternary Boolean relations which are expressible by clauses.

4.38 Example In the problem *Antichain* [73], we are given a finite poset $(P;\leq)$, and we aim to compute the number of antichains in P. This problem can be expressed in the #CSP-form as follows. Let $\varrho_{\leq}$ be the predicate of the natural order on $\{0,1\}$. We assign a variable x_a to each element $a \in P$. Then the #CSP($\{\varrho_{\leq}\}$) instance $\Phi = \bigwedge_{a\leq b} \varrho_{\leq}(x_a, x_b)$ can be shown to be equivalent to the original *Antichain* instance.

To show this, notice that every model φ to Φ satisfies the following condition: if $\varphi(x_a) = 1$ and $a \leq b$ then $\varphi(x_b) = 1$. This means that the set $F_\varphi = \{a \in P \mid \varphi(x_a) = 1\}$ is a filter of P. Hence the models of Φ correspond one-to-one to the filters of P, and consequently, to the antichains of P.

On the other hand, any #CSP($\{\varrho_{\leq}\}$) instance is reducible to an *Antichain* instance, though not so straightforwardly (see [14]). Thus *Antichain* is equivalent to #CSP($\{\varrho_{\leq}\}$).

The general #CSP is known to be #**P**-complete, as follows from Theorem 4.14, or the results of [81] and the examples above. We call a constraint language Γ *#-tractable* if, for every finite $\Gamma_0 \subseteq \Gamma$, the problem #CSP($\Gamma_0$) is polynomial time solvable. The language Γ is said to be #**P**-*complete* if #CSP(Γ_0) is #**P**-complete for a certain finite $\Gamma_0 \subseteq \Gamma$.

The expressive power and the polymorphisms of a constraint language again play crucial roles in determining the complexity of #CSP(Γ).

4.39 Proposition *For any constraint languages,* Γ, Γ_0*, on a finite set* D*, with* Γ_0 *finite, if* $\Gamma_0 \subseteq \langle\Gamma\rangle$*, then* #CSP($\Gamma_0$) *is reducible to* #CSP(Γ) *in polynomial time.*

4.40 Proposition *For any constraint languages* Γ, Γ_0*, on a finite set* D*, with* Γ_0 *finite, if* $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Gamma_0)$*, then* #CSP($\Gamma_0$) *is reducible to* #CSP(Γ) *in polynomial time.*

Proposition 4.40 implies that, as with the decision CSP, the algebra $\mathbb{A}_\Gamma$ fully determines the counting complexity of a constraint language Γ. We will say that a finite algebra $\mathbb{A} = (D;F)$ is *#-tractable* [*♯P-complete*] if so is the constraint language $\mathsf{Inv}(F)$.

The next result shows that, once again, standard constructions preserve tractability.

4.41 Theorem *Let $\mathbb{A}$ be a finite algebra. If $\mathbb{A}$ is #-tractable, then all of its subalgebras, homomorphic images and finite direct powers are also #-tractable. Conversely, if $\mathbb{A}$ has a #**P**-complete subalgebra, homomorphic image, or finite direct power, then $\mathbb{A}$ is #**P**-complete itself.*

4.42 Theorem *A finite algebra is #-tractable (#**P**-complete) if and only if its full idempotent reduct is #-tractable (#**P**-complete).*

The benchmark hard counting problems arise from binary reflexive non-symmetric relations.

4.43 Proposition *If ϱ is a binary reflexive non-symmetric relation on a finite set, then* #CSP($\{\varrho\}$) *is #**P**-complete.*

Theorem 4.41, Proposition 4.43, and the results from [41], provide a link between the complexity of #CSP and Mal'tsev operations, which we will now investigate. The next statement follows from [41, Theorem 9.13].

4.44 Theorem *For a finite algebra $\mathbb{A}$ the following conditions are equivalent:*

(1) *$\mathbb{A}$ does not have a Mal'tsev term operation.*

(2) *There is a finite algebra $\mathbb{B} = (B; F) \in \mathsf{var}(\mathbb{A})$, such that $\mathsf{Inv}(F)$ contains a binary reflexive non-symmetric relation.*

By Proposition 4.43, the algebra $\mathbb{B}$ from Theorem 4.44 (2) is #**P**-complete. Furthermore, Theorem 4.41 implies that $\mathbb{A}$ is also #**P**-complete.

4.45 Corollary *Every finite algebra having no Mal'tsev term operation is#**P**-complete.*

By making use of Corollary 4.45 we can obtain a very easy proof of the dichotomy theorem for the Boolean #CSP ([20], see Theorem 4.14). First, it follows from the results of [72], that any Boolean relation which is invariant under some Mal'tsev operation on $\{0, 1\}$ is also invariant under the unique affine operation on $\{0, 1\}$, $x-y+z$. Hence, by Corollary 4.45, any Boolean constraint language is either #**P**-complete, or else a subset of $\mathsf{Inv}(\{x-y+z\})$. Any relation belonging to $\mathsf{Inv}(\{x-y+z\})$ is the solution space of a system of linear equations over the 2-element field, so it is possible to find a basis for this set in polynomial time. Furthermore, the number of solutions in this set equals 2^n, where n is the number of vectors in the basis.

4.46 Example The *#H-coloring* problem is the counting version of the *Graph H-coloring* problem (see Example 3.5). In this problem, the goal is to find the number of homomorphisms from a given graph G to the fixed graph H.

If the *#H-coloring* problem is restricted to undirected graphs then, as proved in [29], the problem is tractable if every connected component of H is either an isolated vertex, or a complete graph with all loops, or a complete unlooped bipartite graph; otherwise the problem is #**P**-complete. The tractability part of this result is easy, and the hardness part can be easily derived from Corollary 4.45, since symmetric relations (or graphs) invariant under a Mal'tsev operation must be of the form specified above.

Mal'tsev operations on a three-element set, and the structure of relations invariant with respect to them, are exhaustively studied in [10]. Making use of these results, three-element algebras have been completely classified with respect to counting complexity [14].

4.47 Theorem *A three-element algebra is #-tractable if it is Mal'tsev; otherwise it is #**P**-complete.*

An algebra is said to be *uniform* if, for any subalgebra $\mathbb{B}$, the blocks of every congruence of $\mathbb{B}$ are of the same size. Clearly, all two-element algebras, groups and quasi-groups are uniform.

4.48 Theorem *Every uniform Mal'tsev algebra is #-tractable.*

Theorems 4.47 and 4.48 prompt a natural conjecture: a finite algebra is #-tractable if and only if it is Mal'tsev; otherwise it is #**P**-complete. However, this conjecture does not hold, since there is a 5-element Mal'tsev algebra that can be proved to be #**P**-complete.

4.7 The Quantified CSP

The standard constraint satisfaction problem over an arbitrary finite domain can be expressed as follows: given a first-order sentence of the form $\exists x_1 \ldots \exists x_l(\varrho_1 \wedge \ldots \varrho_q)$, where each ϱ_i is an atomic formula, and $x_1, \ldots, x_l$ are the variables appearing in the ϱ_i, determine whether the sentence is true (see Section 1). In this subsection we consider a more general framework which allows arbitrary quantifiers over constrained variables, rather than just existential quantifiers. This form of the CSP is called the *quantified* CSP, or QCSP for short. The Boolean QCSP (also known as QSAT or QBF), and some of its restrictions (such as Q3SAT), have always been standard examples of **PSPACE**-complete problems [33, 67, 75].

All the results presented in this section were first obtained in [8, 19].

4.49 Definition For a constraint language $\Gamma \subseteq R_D$, an *instance* of QCSP(Γ) is a first-order sentence $\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_l x_l\ (\varrho_1 \wedge \cdots \wedge \varrho_q)$, where each ϱ_i is an atomic formula involving a predicate from Γ, $x_1, \ldots, x_l$ are the variables appearing in the ϱ_i, and $\mathcal{Q}_1, \ldots, \mathcal{Q}_l$ are arbitrary quantifiers. The *question* is whether the sentence is true.

Clearly, an instance of CSP(Γ) corresponds to an instance of QCSP(Γ) in which all the quantifiers happen to be existential.

We note that in the Boolean case, the complexity of QCSP(Γ) has been completely classified (see Theorem 4.11). For problems over larger domains no complete classification has yet been obtained, but there are a number of known results concerning the complexity of special cases.

4.50 Example Consider the following *Coloring Construction Game* played by two players, Player 1 and Player 2: given an undirected graph $G = (V, E)$, a linear ordering on V (i.e., a bijection $f : V \to \{1, \ldots, |V|\}$), an ownership function $w : V \to \{1, 2\}$, and a finite set of colours D with $|D| \geq 3$. In the i-th move, the player who owns vertex $f^{-1}(i)$ (that is, Player $w(f^{-1}(i))$) colours it in such a way that its colour is different from the colours of all its neighbours that are already coloured. Player 1 wins if all vertices are coloured at the end of the game.

Deciding whether Player 1 has a winning strategy in an instance of this game can be translated into an instance of the quantified version of the *Graph* $|D|$*-colorability* problem, QCSP($\{\neq_D\}$). To make this translation we view elements from V as variables, elements of E as constraint scopes, the relation $\neq_D$ as the only available constraint relation, the variables from $w^{-1}(1)$ as existentially quantified, the variables from $w^{-1}(2)$ as universally quantified, and the order of quantification as specified by the function f.

Since the problem of determining whether Player 1 has a winning strategy in this game was shown to be **PSPACE**-complete in [5], it follows that QCSP($\{\neq_D\}$) is also **PSPACE**-complete.

It can be shown that, for quantified constraint satisfaction problems, *surjective* polymorphisms play a similar role to that played by arbitrary polymorphisms for ordinary CSPs (cf. Theorem 4.9). Let $\mathsf{s\text{-}Pol}(\Gamma)$ denote the set of all surjective operations from $\mathsf{Pol}(\Gamma)$.

4.51 Theorem *For any constraint languages* $\Gamma, \Gamma_0 \subseteq R_D$*, with* Γ_0 *finite, if* $\mathsf{s\text{-}Pol}(\Gamma) \subseteq \mathsf{s\text{-}Pol}(\Gamma_0)$*, then* QCSP($\Gamma_0$) *is reducible to* QCSP(Γ) *in polynomial time.*

This theorem follows immediately from the next two propositions.

4.52 Definition For any set $\Gamma \subseteq R_D$, the set $[\Gamma]$ consists of all predicates that can be expressed using:

(1) predicates from Γ, together with the binary equality predicate $=_D$ on D,

(2) conjunction,

(3) existential quantification,

(4) universal quantification.

4.53 Proposition *For any constraint languages* $\Gamma, \Gamma_0 \subseteq R_D$*, with* Γ_0 *finite, if* $[\Gamma_0] \subseteq [\Gamma]$*, then* QCSP($\Gamma_0$) *is reducible to* QCSP(Γ) *in polynomial time.*

4.54 Proposition *For any constraint language* Γ *over a finite set,* $[\Gamma] = \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))$.

Note that Proposition 4.54 intuitively means that the expressive power of constraints in the QCSP is determined by their surjective polymorphisms. Hence, in order to show that some relation ϱ belongs to $[\Gamma]$, one does not have give an explicit construction, but instead one can show that ϱ is invariant under all surjective polymorphisms of Γ, which often turns out to be significantly easier.

We remark that the operators $\mathsf{Inv}()$ and $\mathsf{s\text{-}Pol}()$ used in Proposition 4.54 form a Galois connection between R_D and the set of all surjective members of O_D which has not previously been investigated (see, e.g., survey [70]).

Using Theorem 4.51, together with Example 4.50, we can obtain a sufficient condition for **PSPACE**-completeness of QCSP(Γ), in terms of the surjective polymorphisms of Γ.

4.55 Theorem *For any finite set* D *with* $|D| \geq 3$*, and any* $\Gamma \subseteq R_D$*, if every* $f \in \mathsf{s\text{-}Pol}(\Gamma)$ *is of the form* $f(x_1, \ldots, x_n) = g(x_i)$ *for some* $1 \leq i \leq n$ *and some permutation* g *on* D*, then* QCSP(Γ) *is* **PSPACE***-complete.*

The next example uses this result to show that even predicates that give rise to trivial constraint satisfaction problems can give rise to intractable quantified constraint satisfaction problems. This can happen because non-surjective polymorphisms, which may guarantee the tractability of the CSP, do not affect the complexity of the QCSP.

4.56 Example Let τ_s be the s-ary "not-all-distinct" predicate holding on a tuple $(a_1, \ldots, a_s)$ if and only if $|\{a_1, \ldots, a_s\}| < s$. Note that $\tau_s \supseteq \{(a, \ldots, a) \mid a \in D\}$, so every instance of CSP($\{\tau_s\}$) is trivially satisfiable by assigning the same value to all variables.

However, by [71, Lemma 2.2.4], the set $\mathsf{Pol}(\{\tau_{|D|}\})$ consists of all non-surjective operations on D, together with all operations of the form given in Theorem 4.55. Hence, $\{\tau_{|D|}\}$ satisfies the conditions of Theorem 4.55, and QCSP($\{\tau_{|D|}\}$) is **PSPACE**-complete. Similar arguments can be used to show that QCSP($\{\tau_s\}$) is **PSPACE**-complete, for any s in the range $3 \leq s \leq |D|$.

On the tractability side, we have the following result. We call a semilattice operation *bounded* if the corresponding partial order is bounded (that is, it is a lattice order). Recall that the dual discriminator operation is defined by the rule

$$d(x, y, z) = \begin{cases} y & \text{if } y = z, \\ x & \text{otherwise.} \end{cases}$$

Note that the dual discriminator is a special type of near-unanimity operation.

4.57 Theorem *For any constraint language* Γ *over a finite set:*

(1) *if* $\mathsf{Pol}(\Gamma)$ *contains a Mal'tsev operation, or a near-unanimity operation, or a bounded semilattice operation, then* QCSP(Γ) *is tractable;*

(2) *if* $\mathsf{Pol}(\Gamma)$ *contains the dual discriminator operation, then* QCSP(Γ) *is in* **NL**.

Recall that the graph of a permutation π is the binary relation $\{(x, y) \mid y = \pi(x)\}$ (or the binary predicate $\pi(x) = y$), For the special case when Γ contains the set Δ of all graphs of permutations, there is a trichotomy result which says that such problems are either tractable, or **NP**-complete, or **PSPACE**-complete. (We remark that the complexity of the standard CSP(Γ) for such sets Γ was completely classified in [25].)

To state this trichotomy result we need to define two additional surjective operations:

- The k-ary *near projection* operation,

$$l_k(x_1, \ldots, x_k) = \begin{cases} x_1 & \text{if } x_1, \ldots, x_k \text{ are all different,} \\ x_k & \text{otherwise.} \end{cases}$$

- The ternary *switching* operation,

$$s(x, y, z) = \begin{cases} x & \text{if } y = z, \\ y & \text{if } x = z, \\ z & \text{otherwise.} \end{cases}$$

4.58 Theorem *Let $\Delta \subseteq \Gamma \subseteq R_D$, and $|D| \geq 3$.*

- *If* $\mathsf{s\text{-}Pol}(\Gamma)$ *contains the dual discriminator d, or the switching operation s, or (when $|D| \in \{3, 4\}$) an affine operation, then* $\mathrm{QCSP}(\Gamma)$ *is in* **PTIME***;*
- *otherwise, if* $\mathsf{s\text{-}Pol}(\Gamma)$ *contains $l_{|D|}$, then* $\mathrm{QCSP}(\Gamma)$ *is* **NP***-complete;*
- *otherwise* $\mathrm{QCSP}(\Gamma)$ *is* **PSPACE***-complete.*

5 The infinite-valued CSP

There are many computational problems which can be represented as constraint satisfaction problems, but require an infinite set of values. In order to avoid representation problems for infinite objects, we will consider CSPs with infinite sets of values in the following form: fix an infinite relational structure $\mathcal{B}$ of finite signature; the input then is a finite structure $\mathcal{A}$ of the same signature, and the question is whether there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

Here are two well-known examples of problems with an infinite set of possible values.

5.1 Example An instance of the *Acyclic Digraph* problem is a directed graph G, and the question is whether G is acyclic, that is, contains no directed cycles. It is easy to see that this problem is equivalent to $\mathrm{Hom}(\mathcal{B})$ where $\mathcal{B} = (\mathbb{N}; <)$, since a directed graph is acyclic if and only if its vertices can be numbered in such a way that every arc leads from a vertex with smaller number to a vertex with a greater one. This problem is tractable.

5.2 Example An instance of the *Betweenness* problem is a pair (A, T) where A is a finite set and $T \subseteq A^3$; the question is whether there is a function $f : A \to \{1, \ldots, |A|\}$ such that, for every triple $(a, b, c) \in T$, we have either $f(a) < f(b) < f(c)$ or $f(a) > f(b) > f(c)$. This problem is equivalent to $\mathrm{Hom}(\mathcal{B})$ with $\mathcal{B} = (\mathbb{N}, R)$ where

$$R = \{(x, y, z) \in \mathbb{N}^3 \mid x < y < z \text{ or } x > y > z\}.$$

This problem is **NP**-complete [33].

It can be shown that neither of the above two problems can be represented as $\mathrm{CSP}(\Gamma)$ for any constraint language Γ over a finite set D.

5.1 Applicability of polymorphisms

In order to investigate the applicability of the algebraic approach, described in previous sections, to the infinite-valued CSP, the first question to be asked is whether the complexity is determined by the polymorphisms of the constraint relations; that is, whether $\langle \Gamma \rangle = \mathsf{Inv}(\mathsf{Pol}(\Gamma))$ when Γ is a finite constraint language over an infinite domain. It is not hard to see that the inclusion $\langle \Gamma \rangle \subseteq \mathsf{Inv}(\mathsf{Pol}(\Gamma))$ always holds. However, this inclusion can be strict, as the next example shows.

5.3 Example Consider $\Gamma = \{R_1, R_2, R_3\}$ on $\mathbb{N}$, where $R_1 = \{(a, b, c, d) \mid a = b \text{ or } c = d\}$, $R_2 = \{(1)\}$, and $R_3 = \{(a, a+1) \mid a \in \mathbb{N}\}$. It is not difficult to show that every polymorphism of Γ is a projection, and hence $\mathsf{Inv}(\mathsf{Pol}(\Gamma))$ is the set of all relations on $\mathbb{N}$. However, one can check that, for example, the unary relation $\{2, 4\}$ does not belong to $\langle \Gamma \rangle$.

However, for some countable structures $\mathcal{B}$, the required equality does hold, as the next result indicates.

A countable structure $\mathcal{B}$ (of finite signature) is called *homogeneous* if every isomorphism between any pair of substructures is induced by an automorphism of $\mathcal{B}$. A countable structure is called *ω-categorical* if it is determined (up to isomorphism) by its first-order theory. It is known that every countable homogeneous structure is ω-categorical, and that a countable structure is ω-categorical if and only if its automorphism group, when acting on the set of all n-tuples (for any n) of elements from the structure, has only finitely many orbits (see, e.g., [42]).

5.4 Theorem [4] *If $\mathcal{B}_\Gamma$ is a countable ω-categorical structure then $\langle\Gamma\rangle = \mathsf{Inv}(\mathsf{Pol}(\Gamma))$.*

Many examples of countable homogeneous structures, as well as remarks on the complexity of the corresponding constraint satisfaction problems, can be found in [4].

5.2 The interval-valued CSP

One form of infinite-valued CSP which has been widely studied in artificial intelliegence is the case where the values taken by the variables are intervals on the real line. This setting is used to model temporal behaviour of systems, where the intervals represent time intervals during which events occur. The most popular such formalism is Allen's interval algebra (AIA for short), introduced in [1], which concerns binary qualitative relations between intervals. This algebra contains 13 basic relations (see Table 1), corresponding to the 13 distinct ways in which two given intervals can be related. The complete set of relations in AIA consists of the $2^{13} = 8192$ possible unions of the basic relations.

Basic relation		Example	Endpoints
I precedes J	p	`III`	$I^+ < J^-$
J preceded by I	p^{-1}	`       JJJ`	
I meets J	m	`IIII`	$I^+ = J^-$
J met by I	m^{-1}	`    JJJJ`	
I overlaps J	o	`IIII`	$I^- < J^- < I^+$,
J overl. by I	o^{-1}	`  JJJJ`	$I^+ < J^+$
I during J	d	`  III`	$I^- > J^-$,
J includes I	d^{-1}	`JJJJJJJ`	$I^+ < J^+$
I starts J	s	`III`	$I^- = J^-$,
J started by I	s^{-1}	`JJJJJJJ`	$I^+ < J^+$
I finishes J	f	`    III`	$I^+ = J^+$,
J finished by I	f^{-1}	`JJJJJJJ`	$I^- > J^-$
I equals J	$\equiv$	`IIII` `JJJJ`	$I^- = J^-$, $I^+ = J^+$

Table 1: The 13 basic relations in Allen's interval algebra.

Let Γ be a constraint language over the set of intervals on the real line, whose elements are members of Allen's interval algebra, and let $\mathcal{B}_\Gamma$ be the corresponding relational structure. It is not hard to see that every instance of CSP(Γ) can also be (more graphically) viewed as a directed graph whose vertices represent the variables and whose arcs are each labelled with a relation from Γ. The question would then be whether one can assign intervals to the vertices so that all constraints on the arcs are satisfied.

Some well-known combinatorial problems can be represented as CSP(Γ) for a suitable subset Γ of AIA, as the next example indicates.

5.5 Example An undirected graph is called an interval graph if it possible to assign (open) intervals to its nodes so that two intervals intersect if and and only if the corresponding nodes are adjacent. An instance of the *Interval Graph Sandwich* problem [35] consists of two (undirected) graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ such that $E_1 \subseteq E_2$. The question is whether there is E such that $E_1 \subseteq E \subseteq E_2$ and $G = (V, E)$ is an interval graph. This problem is known to be **NP**-complete [35].

This problem can be represented as CSP(Γ) where Γ consists of two relations: "disjoint" (given by $\mathsf{p} \cup \mathsf{p}^{-1} \cup \mathsf{m} \cup \mathsf{m}^{-1}$) and its complement, "intersect" (the union of the other nine basic relations). Indeed, let V be the set of variables; then, to any edge $e \in E_1$ assign the constraint "intersect", to any edge $e \notin E_2$ assign the constraint "disjoint", and leave all other pairs of variables unrelated. Solutions of this CSP precisely correspond to interval graph sandwiches.

Note that the case when $G_1 = G_2$ is known as *Interval Graph Recognition* problem, which is tractable, but this problem is not of the form CSP(Γ) because we cannot leave variables unrelated.

Choosing other pairs of complementary relations, one can obtain other graph sandwich problems, such as the *Overlap* (or *Circle*) *Graph Sandwich* problem [35, 55]

The general CSP problem for AIA is **NP**-complete, as follows from the above example. The problem of classifying subsets of AIA with respect to the complexity of the corresponding CSP has attracted much attention in artificial intelligence (see, for example, [76]).

Allen's interval algebra has three operations on relations: composition, intersection, and inversion. Note that these three operations can each be represented by using conjunction and existential quantification, so, for any subset Γ of AIA, the subalgebra Γ' of AIA generated by Γ has the property that $\Gamma' \subseteq \langle \Gamma \rangle$. It follows from Corollary 4.6 that CSP(Γ) and CSP(Γ') are polynomial-time equivalent. Hence it is sufficient to classify all *subalgebras* of AIA.

Using computations in subalgebras of AIA, manipulations with primitive positive formulas (called *derivations* in [55]) and a number of new **NP**-completeness results, a complete classification of the complexity of all subsets of AIA was accomplished in [55], where the following result was obtained.

5.6 Theorem *Let Γ be a subset of Allen's interval algebra. If Γ is contained in one of the eighteen subalgebras listed in Table 2, then* CSP(Γ) *is tractable; otherwise it is* **NP***-complete.*

In Table 2, for the sake of brevity, relations between intervals are written as collections of basic relations. So, for instance, we write (pmod) instead of $\mathsf{p} \cup \mathsf{m} \cup \mathsf{o} \cup \mathsf{d}$. We also use the symbol $\pm$, which should be interpreted as follows: a condition involving $\pm$ means the conjunction of two conditions, one corresponding to $+$ and one corresponding to $-$. For example, the condition $(\mathsf{o})^{\pm 1} \subseteq r \Leftrightarrow (\mathsf{d})^{\pm 1} \subseteq r$ means that both $(\mathsf{o}) \subseteq r \Leftrightarrow (\mathsf{d}) \subseteq r$ and $(\mathsf{o}^{-1}) \subseteq r \Leftrightarrow (\mathsf{d}^{-1}) \subseteq r$ hold.

$$
\begin{aligned}
&\mathcal{S}_{\mathsf{p}} = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{p})^{\pm 1} \subseteq r\}\\
&\mathcal{S}_{\mathsf{d}} = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{d}^{-1})^{\pm 1} \subseteq r\}\\
&\mathcal{S}_{\mathsf{o}} = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r\}\\
&\mathcal{A}_1 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s}^{-1})^{\pm 1} \subseteq r\}\\
&\mathcal{A}_2 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r\}\\
&\mathcal{A}_3 = \{r \mid r \cap (\mathsf{pmodf})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r\}\\
&\mathcal{A}_4 = \{r \mid r \cap (\mathsf{pmodf}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{E}_{\mathsf{p}} = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{p})^{\pm 1} \subseteq r\}\\
&\mathcal{E}_{\mathsf{d}} = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{d})^{\pm 1} \subseteq r\}\\
&\mathcal{E}_{\mathsf{o}} = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r\}\\
&\mathcal{B}_1 = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r\}\\
&\mathcal{B}_2 = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f})^{\pm 1} \subseteq r\}\\
&\mathcal{B}_3 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{s}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r\}\\
&\mathcal{B}_4 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{s})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r\}
\end{aligned}
$$

$$
\mathcal{E}^* = \left\{ r \;\middle|\; \begin{array}{l} 1)\ r \cap (\mathsf{pmod})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r, \text{ and} \\ 2)\ r \cap (\mathsf{ff}^{-1}) \neq \emptyset \Rightarrow (\equiv) \subseteq r \end{array} \right\}
$$

$$
\mathcal{S}^* = \left\{ r \;\middle|\; \begin{array}{l} 1)\ r \cap (\mathsf{pmod}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r, \text{ and} \\ 2)\ r \cap (\mathsf{ss}^{-1}) \neq \emptyset \Rightarrow (\equiv) \subseteq r \end{array} \right\}
$$

$$
\mathcal{H} = \left\{ r \;\middle|\; \begin{array}{l} 1)\ r \cap (\mathsf{os})^{\pm 1} \neq \emptyset \ \&\ r \cap (\mathsf{o}^{-1}\mathsf{f})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{d})^{\pm 1} \subseteq r, \text{ and} \\ 2)\ r \cap (\mathsf{ds})^{\pm 1} \neq \emptyset \ \&\ r \cap (\mathsf{d}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r, \text{ and} \\ 3)\ r \cap (\mathsf{pm})^{\pm 1} \neq \emptyset \ \&\ r \not\subseteq (\mathsf{pm})^{\pm 1} \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r \end{array} \right\}
$$

$$
\mathcal{A}_{\equiv} = \{r \mid r \neq \emptyset \Rightarrow (\equiv) \subseteq r\}
$$

Table 2: The 18 maximal tractable subalgebras of Allen's algebra.

It follows from Theorem 5.6 that CSP($\{r\}$), where r is a single relation in AIA, is **NP**-complete if and only if r either satisfies $r \cap r^{-1} = (\mathsf{mm}^{-1})$ or is a relation with $r \cap r^{-1} = \emptyset$ and such that neither r nor r^{-1} is contained in one of $(\mathsf{pmod}^{-1}\mathsf{sf}^{-1})$, $(\mathsf{pmod}^{-1}\mathsf{s}^{-1}\mathsf{f}^{-1})$, (pmodsf) and (pmodsf^{-1}).

It was noted in [4] that AIA (without its operations) is in fact a homogeneous relational structure. Since we may assume, without loss of generality, that all intervals under consideration have rational endpoints, we obtain a countable homogeneous structure of finite signature. Therefore, by Theorem 5.4, the complexity classification problem for subsets of AIA can be tackled using polymorphisms. Such an approach may provide a route to simplifying the involved classification proof given in [55].

References

[1] J. F. Allen, Maintaining knowledge about temporal intervals, *Comm. ACM* **26** (1983), 832–843.

[2] J. F. Allen, *Natural Language Understanding*, Benjamin Cummings, 1994.

[3] J. Berman, E. Kiss, P. Pröhle, and Á. Szendrei, The set of types of a finitely generated variety, *Discrete Math.* **112** (1–3) (1993), 1–20.

[4] M. Bodirsky and J. Nešetřil, Constraint satisfaction problems with countable homogeneous structures, in: *Computer Science Logic (Vienna, 2003)*, Lecture Notes in Comput. Sci. **2803**, Springer, Berlin, 2003, 44–57.

[5] H. Bodlaender, On the complexity of some coloring games, *Internat. J. Found. Comput. Sci.*, **2** (2) (1991), 133–148.

[6] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer, Equivalence and isomorphism for Boolean constraint satisfaction, in: *Computer Science Logic (Edinburgh, 2002)*, Lecture Notes in Comput. Sci. **2471**, Springer, Berlin, 2002, 412–426.

[7] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer, The complexity of Boolean constraint isomorphism, Technical Report cs.CC/0306134, ACM Computing Research Repository, 2003.

[8] F. Börner, A. Bulatov, P. Jeavons, and A. Krokhin, Quantified constraints: Algorithms and complexity, in: *Computer Science Logic (Vienna, 2003)*, Lecture Notes in Comput. Sci. **2803**, Springer, Berlin, 2003, 58–70.

[9] A. Bulatov, Combinatorial problems raised from 2-semilattices, submitted for publication.

[10] A. Bulatov, Three-element Mal'tsev algebras, submitted for publication.

[11] A. Bulatov, A dichotomy theorem for constraints on a three-element set, in: *Foundations of Computer Science (Vancouver, BC, 2002)*, IEEE Comput. Soc., 2002, 649–658.

[12] A. Bulatov, Mal'tsev constraints are tractable, Technical Report PRG-RR-02-05, Computing Laboratory, University of Oxford, UK, 2002.

[13] A. Bulatov, Tractable conservative constraint satisfaction problems, in: *Logic in Computer Science (Ottawa, ON, 2003)*, IEEE Comput. Soc., 2003, 321–330.

[14] A. Bulatov and V. Dalmau, Towards a dichotomy theorem for the counting constraint satisfaction problem, in: *Foundations of Computer Science (Boston, MA, 2003)*, IEEE Comput. Soc., 2003, 562–571.

[15] A. Bulatov and P. Jeavons, Algebraic structures in combinatorial problems, Technical Report MATH-AL-4-2001, Technische Universität Dresden, Germany, 2001.

[16] A. Bulatov, P. Jeavons, and M. Volkov, Finite semigroups imposing tractable constraints, in: *Semigroups, Algorithms, Automata and Languages* (Gracinda M.S.Gomes, Jean-Eric Pin, Pedro V.Silva, eds), World Scientific, Singapore, 2002, 313–329.

[17] A. Bulatov, A. Krokhin, and P. Jeavons, Classifying complexity of constraints using finite algebras, submitted for publication.

[18] A. Bulatov, A. Krokhin, and P. Jeavons, Constraint satisfaction problems and finite algebras, in: *Automata, Languages and Programming (Geneva, 2000)*, Lecture Notes in Comput. Sci. **1853**, Springer, Berlin, 2000, 272–282.

[19] H. Chen, Quantified constraint satisfaction problems: Closure properties, complexity, and algorithms, manuscript, 2003.

[20] N. Creignou and M. Hermann, Complexity of generalized satisfiability counting problems, *Inform. and Comput.* **125** (1) (1996), 1–12.

[21] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM Monographs on Discrete Mathematics and Applications **7**, SIAM, Philadelphia, 2001.

[22] P. Crescenzi and G. Rossi, On the Hamming distance of constraint satisfaction problems, *Theoret. Comput. Sci.* **288** (1) (2002), 85–100.

[23] V. Dalmau, Some dichotomy theorems on constant-free quantified Boolean formulas, Technical Report TR LSI-97-43-R, Department LSI, Universitat Politecnica de Catalunya, 1997.

[24] V. Dalmau, Constraint satisfaction problems in non-deterministic logarithmic space, in: *Automata, Languages and Programming (Malaga, 2002)*, Lecture Notes in Comput. Sci. **2380**, Springer, Berlin, 2002, 414–425.

[25] V. Dalmau, A new tractable class of constraint satisfaction problems, *Ann. Math. Artif. Intell.*, to appear.

[26] V. Dalmau, Ph. G. Kolaitis, and M. Y. Vardi, Constraint satisfaction, bounded treewidth, and finite-variable logics, in: *Principles and Practice of Constraint Programming (Ithaca, NY, 2002)*, Lecture Notes in Comput. Sci. **2470**, Springer, Berlin, 2002, 310–326.

[27] V. Dalmau and J. Pearson, Set functions and width 1 problems, in: *Principles and Practice of Constraint Programming (Alexandria, VA, 1999)*, Lecture Notes in Comput. Sci. **1713**, Springer, Berlin, 1999, 159–173.

[28] N. W. Dunkin, J. E. Bater, P. G. Jeavons, and D. A. Cohen, Towards high order constraint represenations for the frequency assignment problem, Technical Report CSD-TR-98-05, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey, UK, 1998.

[29] M. Dyer and C. Greenhill, The complexity of counting graph homomorphisms, *Random Structures Algorithms* **17** (2000).

[30] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*, 2th ed., Perspectives in Mathematical Logic, Springer, Berlin, 1999.

[31] T. Feder and M. Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory, *SIAM J. Comput.* **28** (1998), 57–104.

[32] E. C. Freuder and R. Wallace, Partial constraint satisfaction, *Artificial Intelligence* **58** (1992), 21–70.

[33] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[34] M. Goldmann and A. Russell, The complexity of solving equations over finite groups, *Inform. and Comput.* **178** (1) (2002), 253–262.

[35] M.C. Golumbic, H. Kaplan, and R. Shamir, Graph sandwich problems, *J. Algorithms* **19** (3) (1995), 449–473.

[36] G. Gottlob, L. Leone, and F. Scarcello, Hypertree decomposition and tractable queries, *J. Comput. System Sci.* **64** (3) (2002), 579–627.

[37] M. Grohe, The complexity of homomorphism and constraint satisfaction problems seen from the other side, in: *Foundations of Computer Science (Boston, MA, 2003)*, IEEE Comput. Soc., 2003, 552–561.

[38] M. Gyssens, P. G. Jeavons, and D. A. Cohen, Decomposing constraint satisfaction problems using database techniques, *Artificial Intelligence* **66** (1) (1994) 57–89.

[39] P. Hell, Algorithmic aspects of graph homomorphisms, in: *Surveys in Combinatorics 2003* (C. Wensley, ed.), LMS Lecture Note Series **307**, Cambridge University Press, 2003, 239–276.

[40] P. Hell and J. Nešetřil, On the complexity of H-coloring, *J. Combin. Theory Ser. B* **48** (1990), 92–110.

[41] D. Hobby and R. N. McKenzie, *The Structure of Finite Algebras*, Contemporary Mathematics **76**, Amer. Math. Soc., Providence, R.I., 1988.

[42] W. Hodges, *A Shorter Model Theory*, Cambridge University Press, 1997.

[43] P. G. Jeavons, On the algebraic structure of combinatorial problems, *Theoret. Comput. Sci.* **200** (1998), 185–204.

[44] P. G. Jeavons, D. A. Cohen, and M. C. Cooper, Constraints, consistency and closure, *Artificial Intelligence* **101** (1–2) (1998), 251–265.

[45] P. G. Jeavons, D. A. Cohen, and M. Gyssens, Closure properties of constraints, *J. ACM* **44** (1997), 527–548.

[46] P. G. Jeavons, D. A. Cohen, and M. Gyssens, How to determine the expressive power of constraints, *Constraints* **4** (1999), 113–131.

[47] L. Juban, Dichotomy theorem for generalized unique satisfiability problem, in: *Fundamentals of Computation Theory (Iasi, 1999)*, Lecture Notes in Comput. Sci. **1684**, Springer, Berlin, 1999, 327–337.

[48] H. Kautz and B. Selman, Planning as satisfiability, in: *Tenth European Conference on Artificial Intelligence (Vienna, 1992)*, John Wiley and Sons, Chichester, 1992, 359–363.

[49] D. Kavvadias and M. Sireni, The inverse satisfiability problem, *SIAM J. Comput.* **28** (1) (1998) 152–163.

[50] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson, The approximability of constraint satisfaction problems, *SIAM J. Comput.* **30** (6) (2001), 1863–1920.

[51] L. Kirousis and Ph. Kolaitis, A dichotomy in the complexity of propositional circumscription, in: *Logic in Computer Science (Boston, MA, 2001)*, IEEE Comput. Soc., 2001, 71–80.

[52] L. Kirousis and Ph. Kolaitis, On the complexity of model checking and inference in minimal models, in: *Logic Programming and Nonmonotonic Reasoning (Vienna, 2001)*, Lecture Notes in Comput. Sci. **2173**, Springer, Berlin, 2001, 42–53.

[53] L. Kirousis and Ph. Kolaitis, The complexity of minimal satisfiability problems, *Inform. and Comput.* **187** (2003), 20–39.

[54] Ph. G. Kolaitis and M. Y. Vardi, Conjunctive-query containment and constraint satisfaction, *J. Comput. System Sci.* **61** (2000), 302–332.

[55] A. Krokhin, P. Jeavons, and P. Jonsson, Reasoning about temporal relations: The maximal tractable subalgebras of Allen's interval algebra, *J. ACM* **50** (5) (2003), 591–640.

[56] A. Krokhin and P. Jonsson, Recognizing frozen variables in constraint satisfaction problems, Technical Report TR03-062, Electronic Colloquium on Computational Complexity, 2003.

[57] R. E. Ladner, On the structure of polynomial time reducibility, *J. ACM* **22** (1975), 155–171.

[58] B. Larose and L. Zádori, Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras, manuscript, 2003.

[59] T. Łuczak and J. Nešetřil, A probabilistic approach to the dichotomy problem, Technical report 2003-640, KAM-DIMATIA Series, 2003.

[60] A. K. Mackworth, Constraint satisfaction, in: *Encyclopedia of Artificial Intelligence, vol. 1* (S. C. Shapiro, ed.), Wiley Interscience, 1992, 285–293.

[61] S. S. Marchenkov, Homogeneous algebras, *Problemy Kibernet.* **39** (1982), 85–106, (in Russian).

[62] K. Marriott and P. J. Stuckey, *Programming with Constraints: an Introduction*, MIT Press, 1998.

[63] R. N. McKenzie, G. F. McNulty, and W. F. Taylor, *Algebras, Lattices and Varieties*, vol. I, Wadsworth and Brooks, CA, 1987.

[64] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, *Inform. Sci.* **7** (1974), 95–132.

[65] C. Moore, P. Tesson, and D. Therien, Satisfiability of systems of equations over finite monoids. in: *Mathematical Foundations of Computer Science (Marianske Lazne, 2001)*, Lecture Notes in Comput. Sci. **2136**, Springer, Berlin, 2001, 537–547.

[66] B. A. Nadel, Constraint satisfaction in Prolog: complexity and theory-based heuristics, *Inform. Sci.* **83** (3–4) (1995), 113–131.

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

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

[69] N. Pippenger, *Theories of Computability*, Cambridge University Press, Cambridge, 1997.

[70] R. Pöschel, Galois connections for operations and relations, Technical Report MATH-AL-8-2001, Technische Universität Dresden, Germany, 2001.

[71] R. Pöschel and L. A. Kalužnin, *Funktionen- und Relationenalgebren*, DVW, Berlin, 1979.

[72] E. L. Post, The two-valued iterative systems of mathematical logic, in: *Annals of Mathematical Studies* **5**, Princeton University Press, 1941.

[73] J. S. Provan and M. O. Ball, The complexity of counting cuts and of computing the probability that a graph is connected, *SIAM J. Comput.* **12** (4) (1983), 777–788.

[74] S. Reith and H. Vollmer, Optimal satisfiability for propositional calculi and constraint satisfaction problems, *Inform. and Comput.* **186** (1) (2003), 1–19.

[75] T. J. Schaefer, The complexity of satisfiability problems, in: *Tenth ACM Symposium on Theory of Computing (San Diego, CA, 1978)*, ACM Press, New York, 1978, 216–226.

[76] E. Schwalb and L. Vila, Temporal constraints: a survey, *Constraints* **3** (2–3) (1998), 129–149.

[77] A. Szendrei, Simple surjective algebras having no proper subalgebras, *J. Austral. Math. Soc. Ser. A* **48** (1990), 434–454.

[78] W. Taylor, Varieties obeying homotopy laws, *Canad. J. Math.* **29** (1977) 498–527.

[79] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.

[80] L. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* **8** (1979), 189–201.

[81] L. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8** (3) (1979), 410–421.