Functions of multiple-valued logic and the complexity of constraint satisfaction: A short survey

Andrei Krokhin
Department of Computer Science
University of Warwick
Coventry CV4 7AL, UK
andrei.krokhin@dcs.warwick.ac.uk

Andrei Bulatov, Peter Jeavons
Computing Laboratory
University of Oxford
Oxford OX1 3QD, UK
{*andrei.bulatov,peter.jeavons*}*@comlab.ox.ac.uk*

Abstract

Many computational problems arising in artificial intelligence, computer science and elsewhere can be represented as constraint satisfaction and optimization problems. In this short survey we discuss an approach that is related to the algebraic component of multiple-valued logic and that has proved to be very succesful in studying the complexity of constraint satisfaction.

1. Constraint Satisfaction Problems

The constraint satisfaction problem (CSP) is a powerful general framework in which a variety of combinatorial problems can be expressed [12, 31, 32, 44]. 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 [26] and scheduling [45], frequency assignment problems [16] and image processing [34], programming language analysis [35] and natural language understanding [1]. In database theory, it has been shown that the key problem of conjunctive-query evaluation can be viewed as a constraint satisfaction problem [19, 30]. Furthermore, some central problems in combinatorial optimization can be represented as constraint problems [12, 28]. 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 [12, 17]. 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., [32]).

The standard toy example of a problem modelled as a constraint satisfaction problem is the 8-queen problem: place eight queens on a chess board so that no queen can capture any other one. 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.

Now let us give a formal definition of the CSP.

Definition 1 *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* $\langle s_i, \rho_i \rangle$ *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.*

In this paper we consider only CSPs with a finite domain of values, so we will always assume that $D = E_k = \{0, 1, \ldots, , k-1\}$, $k \geq 2$. Now we give some examples of natural problems and their representations as CSPs.

Example 1 An instance of the standard propositional SATISFIABILITY problem [18, 36] is specified by giving a formula in propositional logic, that is, a conjunction of clauses, 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 $(V, E_2, \mathcal{C})$ of CSP, where V is the set of all variables appearing in the clauses ϕ_i, and $\mathcal{C}$ is the set of constraints $\{\langle s_1, \rho_1\rangle, \ldots, \langle s_n, \rho_n\rangle\}$, where each constraint $\langle s_k, \rho_k\rangle$ is constructed as follows: s_k is the list of 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 SATISFIABILITY can be expressed as an instance of CSP. □

Example 2 An instance of GRAPH UNREACHABILITY consists of a graph $G = (V, E)$ and a pair of vertices, $v, w \in V$. The question is whether there is no path in G from v to w. This can be expressed as the CSP instance $(V, E_2, \mathcal{C})$ where $\mathcal{C}$ is the following set of constraints:

$$\{\langle e, \{(0,0), (1,1)\}\rangle | e \in E\} \cup \{\langle (v), \{(0)\}\rangle, \langle (w), \{(1)\}\rangle\}.$$

□

Many other examples of well-known problems expressed as CSPs can be found further in this paper and also in [23].

Example 1 suggests that any instance of CSP can be represented in a logical form. Indeed, using the standard correspondence between relations and predicates, one can re-write an instance of CSP as a first-order formula $\rho_1(s_1) \wedge \ldots \wedge \rho_q(s_q)$ where ρ_i $(1 \leq i \leq q)$ are predicates on E_k and $\rho_i(s_i)$ means ρ_i applied to the tuple s_i of variables. The question then would be whether this formula is satisfiable [42]. In this paper we will be working with this logical form of CSP. This form is commonly used in database theory because it corresponds so closely to conjunctive query evaluation [30].

Another important reformulation of CSP is the HOMOMORPHISM problem: whether there exists a homomorphism between two relational structures (see [17, 19, 30]).

2. Related Constraint Problems

As with many other computational problems, it is not only the decision version of CSP (that is, whether or not a solution exists) 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 instance have?

  A standard natural problem associated with every computational decision problem [12].

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

  Problems of this form have been fundamental examples of **PSPACE**-complete problems [12, 14, 42]. Any instance of ordinary CSP can be viewed as an instance of this problem with all quantifiers existential.

- **Minimal Solution**
  Given an 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 propositional circumscription, a framework used in artificial intelligence to formalize common-sense reasoning [29].

- **Equivalence**
  Given two 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 [2].

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

  This problem is related to efficient knowledge representation issues in artificial intelligence [27].

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

  A standard natural problem associated with every computational decision problem [12].

- **MAX CSP**
  Maximize the number of satisfied constraints in an instance.

  A number of optimization problems, e.g., maximum cut, can be expressed as MAX CSP problems [12, 28].

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

  Many optimization problems including maximum clique are of this form; in the Boolean case this problem is known as MAX ONES [12, 28].

- **Maximum Hamming Distance**
  Find two solutions 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 [13].

3. Parameterization of the CSP

The main object of our interest is the complexity of constraint satisfaction problems. We refer the reader to [36] 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, 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., [37]).

With any CSP instance one can associate two natural parameters: a hypergraph, showing which variables constrain which others, and a set of relations reflecting the way in which the values are constrained. The hypergraph is defined on the set of variables used in the instance, each hyperedge consisting of all variables appearing in one constraint scope. The set of relations consists of the constraint relations used in the instance. Therefore the general CSP can be restricted by fixing the set of hypergraphs or the set of relations that are allowed to be used in instances.

The case when the set of hypergraphs is fixed has been studied in connection with databases [19, 30]. In this paper we concentrate on the case when the set of relations allowed in instances is fixed, but there is no restriction on the form of the associated hypergraphs. Let $R_k^{(n)}$ denote the set of all n-ary relations (or predicates) on E_k, and let $R_k = \bigcup_{n=1}^{\infty} R_k^{(n)}$.

Definition 2 *A* constraint language *over E_k is a subset Γ of R_k.*

The constraint satisfaction problem over Γ*, denoted* CSP(Γ)*, is the subclass of* CSP *defined by the following property: any constraint relation in any instance must belong to Γ.*

Of course, such a parameterization can be considered for all the related constraint problems discussed above.

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

Example 3 The NOT-ALL-EQUAL SATISFIABILITY problem [18, 42] is a restricted version of the standard SATISFIABILITY problem 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 $\{0, 1\}$:

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

□

Example 4 An instance of GRAPH k-COLORABILITY consists of a graph G. The question is whether the vertices of G can be labelled with k colours so that adjacent vertices are assigned different colours.

This problem corresponds to the problem CSP($\{\neq_{E_k}\}$) where $\neq_{E_k}$ is the disequality relation on E_k given by

$$\neq_{E_k} \;=\; \{(a,b) \in E_k^2 \mid a \neq b\}.$$

This problem is tractable if $k = 2$ and it is **NP**-complete for any $k \geq 3$ (see [18]).

□

Example 5 An instance of LINEAR EQUATIONS is a system of linear equation over a field. 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 tractable. □

Following a seminal work by Schaefer in 1978 [42], many researchers have studied the following problem:

Problem 1 *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 is E_2), such problems are sometimes called "generalized satisfiability problems". Schaefer obtained a complete classification for the ordinary CSP over E_2 [42], which is described in Section 6. Over the last decade, classifications for many related Boolean constraint problems, including all of the problems mentioned in Section 2, have been completed [2, 12, 13, 14, 27, 29]. Some of these classifications are also described in Section 6.

Classifying the complexity in the non-Boolean case has proved to be a very difficult task. However, the approach described in the next two sections has made it possible to obtain strong results in this direction; we discuss these results in Section 7.

4. Expressive Power of Constraint Languages

In any constraint satisfaction problem instance some of the required relationships between variables are given explicitly in the constraints, whilst others generally arise implicitly from interactions of 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.

Example 6 Let Γ be the set containing a single binary relation, χ, over the set E_3, 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

$$\mathcal{P} = (\{v_1, v_2, v_3, v_4\}, E_3, \{C_1, C_2, C_3, C_4, C_5\}),$$

where

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

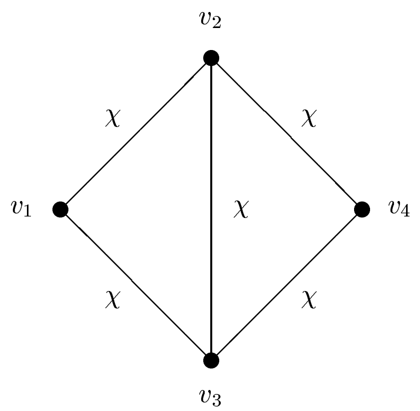

Figure 1. The CSP instance $\mathcal{P}$

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.

Definition 3 *A relation ρ can be* expressed *in a constraint language Γ over E_k if there exists a problem instance (V, E_k, C) in* CSP(Γ)*, and a list, s, of variables, such that the solutions to (V, E_k, 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.

This set can be characterised in a number of different ways [25]. 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 [20]. 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.

Definition 4 *For any set of relations Γ over E_k, 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 E_k (denoted $=_{E_k}$),*
2. *conjunction, and*
3. *existential quantification.*

Example 7 Example 6 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 6 that

$$\chi'(x,y) \equiv \exists a \exists b \chi(x,a) \wedge \chi(x,b) \wedge \chi(a,b) \wedge \chi(a,y) \wedge \chi(b,y).$$

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

5. 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 CSP(Γ).

Proposition 1 *For any constraint language Γ and any relation ρ belonging to the expressive power of Γ,* CSP($\Gamma \cup \{\rho\}$) *is reducible in polynomial time to* CSP(Γ).

This result can be established simply by noting that, given any problem instance in CSP($\Gamma \cup \{\rho\}$), we can obtain an equivalent instance in CSP(Γ) 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.

Corollary 1 *For any constraint language Γ, and any finite constraint language Γ_0, if Γ_0 is contained in the expressive power of Γ, then* CSP(Γ_0) *is reducible to* CSP(Γ) *in polynomial time.*

Corollary 1 implies that for any finite constraint language Γ, the complexity of CSP(Γ) 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 CSP(Γ)?

A very successful approach to this question has been developed in [10, 23, 24], using techniques from universal algebra and multiple-valued logic [33, 39]. To describe this approach, we need to consider arbitrary k-valued *operations* (or *functions*). We will use $O_k^{(n)}$ to denote the set of all n-ary operations on the set E_k (that is, the set of mappings $f\colon E_k^n \to E_k$), and O_k to denote the set $\bigcup_{n=1}^{\infty} O_k^{(n)}$.

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

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

An essentially unary operation with g being 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 E_k can be extended in a standard way to an operation on tuples over E_k, as follows. For any operation $f \in O_k^{(n)}$, and any collection of tuples $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n \in E_k^m$, where $\vec{a}_i = (\vec{a}_{i1}, \ldots, \vec{a}_{im})$ $(i = 1 \ldots n)$, define $f(\vec{a}_1, \ldots, \vec{a}_n)$ to be the tuple $(\, f(\vec{a}_{11}, \ldots, \vec{a}_{n1}), \ldots, f(\vec{a}_{1m}, \ldots, \vec{a}_{nm})\,)$.

Definition 5 *For any relation $\varrho \in R_k^{(m)}$, and any operation $f \in O_k^{(n)}$, 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 $C \subseteq O_k$ will be denoted $\mathsf{Inv}(C)$. The set of all operations that are polymorphisms of every relation from some set $\Gamma \subseteq R_k$ will be denoted $\mathsf{Pol}(\Gamma)$. The operators Inv and Pol form a Galois correspondence between R_k and O_k (see Proposition 1.1.14 of [39]). A basic introduction to this correspondence can be found in [38], and a comprehensive study in [39].

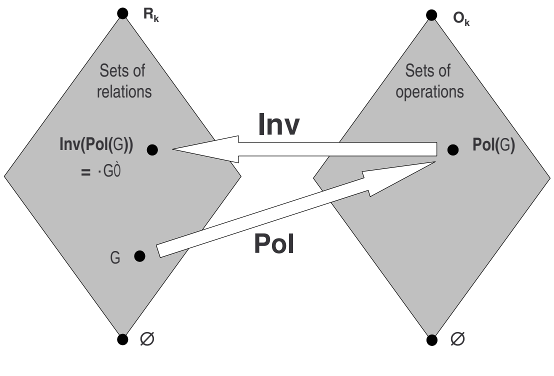

Figure 2. The operators Inv and Pol

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* [39]; they have received much attention in multiple-valued logic (see, e.g., [41]). Moreover, the following useful characterisation of sets of the form $\mathsf{Inv}(\mathsf{Pol}(\Gamma))$ is given in [39].

Theorem 1 *For every set $\Gamma \subseteq R_k$, $\mathsf{Inv}(\mathsf{Pol}(\Gamma)) = \langle\Gamma\rangle$.*

This result was combined with Corollary 1 to obtain the following result in [23].

Theorem 2 *For any constraint language $\Gamma \subseteq R_k$, and any finite constraint language $\Gamma_0 \subseteq R_k$, if $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Gamma_0)$, then* CSP$(\Gamma_0)$ *is reducible to* CSP(Γ) *in polynomial time.*

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.

Now we are faced with a new question: how can we obtain sufficient information about $\mathsf{Pol}(\Gamma)$ to determine the complexity of CSP(Γ)? Happily, it can be shown that this question can itself be formulated as a particular form of constraint satisfaction problem.

Definition 6 *Let Γ be a constraint language over E_k.*

For any natural number $m > 0$, the indicator problem *for Γ of order m is defined to be the constraint satisfaction problem instance $\mathcal{IP}(\Gamma, m) = (E_k^m, E_k, \{C_1, C_2, \ldots, C_q\})$, where $q = \sum_{\rho \in \Gamma} |\rho|^m$, and the constraints $C_1, C_2, \ldots, C_q$ are defined as follows.*

For each $\rho \in \Gamma$, and for each sequence $t_1, t_2, \ldots, t_m$ of tuples from ρ, there is a constraint $C_i = \langle s_i, \rho\rangle$ with $s_i = (v_1, v_2, \ldots, v_n)$, where n is the arity of ρ and $v_j = (t_1[j], t_2[j], \ldots, t_m[j])$ for $j = 1$ to n.

Note that for any set of relations Γ over a set E_k, the indicator problem $\mathcal{IP}(\Gamma, m)$ has k^m variables, and each variable corresponds to an m-tuple over E_k. It is straightforward to check that the solutions to $\mathcal{IP}(\Gamma, m)$ are precisely the mappings from E_k^m to E_k under which every element of Γ is invariant, that is, the polymorphisms of Γ.

Several concrete examples of indicator problems and their solutions can be found in [22].

Example 8 Consider the constraint language $\Gamma = \{N\}$, where N is the binary "not-all-equal" relation over E_2 defined in Example 3.

The indicator problem for Γ of order 3, $\mathcal{IP}(\Gamma, 3)$, has 8 variables and 216 constraints, and has exactly 6 solutions. Each of these solutions is an essentially unary operation.

Hence $\mathsf{Pol}(\Gamma)$ contains only six distinct ternary operations, which are all essentially unary. Using general results from clone theory [43], it can be shown that this implies that in this case $\mathsf{Pol}(\Gamma)$ contains essentially unary operations only. $\square$

Example 9 Consider the constraint language $\Gamma = \{\chi, \xi\}$, where χ is the binary relation over E_3 defined in Example 6 and ξ is the ternary relation over E_3 containing the single tuple $(0, 1, 2)$.

The indicator problem for Γ of order 3, $\mathcal{IP}(\Gamma, 3)$, has 27 variables and 217 constraints, but has only 3 solutions. Each of these solutions is a projection operation.

Hence $\mathsf{Pol}(\Gamma)$ contains only three distinct ternary operations, which are the three ternary projection operations. Using general results from clone theory [43], it can be shown that this implies that in this case $\mathsf{Pol}(\Gamma)$ contains only projections. ☐

6. 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 [42]. A computational problem is called *tractable* if there is a polynomial-time algorithm deciding every instance of the problem.

Theorem 3 *For any set of relations $\Gamma \subseteq R_2$,* CSP(Γ) *is tractable when 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 field with two elements.*

In all other cases CSP(Γ) *is* **NP**-*complete.*

This result establishes a dichotomy for versions of this problem parameterised 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.

Using the algebraic approach developed in the previous sections, together with the knowledge of possible clones on a two-element set obtained in [40], Schaefer's result can be reformulated in the following much more concise form.

Theorem 4 *For any set of relations $\Gamma \subseteq R_2$,* CSP(Γ) *is tractable when* $\mathsf{Pol}(\Gamma)$ *contains any essentially non-unary operation or a constant operation. Otherwise it is* **NP**-*complete.*

Example 10 Recall the relation N over E_2 defined in Example 3. It was shown in Example 8 that $\mathsf{Pol}(\{N\})$ contains essentially unary operations only, and hence, by Theorem 4, 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 the 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.

Theorem 5 *Let $\Gamma \subseteq R_2$ be a Boolean constraint language. Then the following is true about constraint problems parameterized by Γ.*

- *The* **Counting Problem** *is tractable if* $\mathsf{Pol}(\Gamma)$ *contains the unique affine operation on E_2, $x - y + z$. Otherwise it is* **#P**-*complete [11, 12].*
- *The* **Quantified Problem** *is tractable if* $\mathsf{Pol}(\Gamma)$ *contains an essentially non-unary operation. Otherwise it is* **PSPACE**-*complete [12, 14].*
- *The* **Equivalence** *problem is tractable if* $\mathsf{Pol}(\Gamma)$ *contains an essentially non-unary operation or a constant operation. Otherwise it is* **coNP**-*complete [2].*
- *The* **Inverse Satisfiability** *problem is tractable if* $\mathsf{Pol}(\Gamma)$ *contains an essentially non-unary operation. Otherwise it is* **coNP**-*complete [27].*
- *The* **Maximum Hamming Distance** *problem is tractable if* $\mathsf{Pol}(\Gamma)$ *contains either a constant, or the affine operation and the negation operation on E_2[13].*

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

Example 11 Recall the relation N over E_2 defined in Example 3. It was shown in Example 8 that $\mathsf{Pol}(\{N\})$ contains essentially unary operations only. Hence, by Theorem 5, 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 involving only conjunctions of the relation 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.

□

7. Complexity of Non-Boolean Problems

Obtaining a complete complexity classifications for a constraint problem over an arbitrary set of values containing more than 2 elements, remains a very challenging open problem, even for the standard decision-problem version of the CSP set out in Definition 1. One reason for this is that in the Boolean case the number of relational clones is countable and they are all fully characterized [39, 40], but for $k \geq 3$ the number of relational clones is continuum (see, e.g., [39]), and there is strong evidence that a description similar to the $k = 2$ case is impossible to obtain (see, e.g., [4]).

However, using the algebraic approach described above a large number of tractable and **NP**-complete cases have now been identified for constraint problems over arbitrary finite sets of values. To describe these cases we need to define a number of special operations.

Definition 7

- *A binary operation f is said to be a* semilattice *operation if it satisfies the following identities:*
  - *(a)* $f(x, x) = x$; *(idempotency)*
  - *(b)* $f(x, y) = f(y, x)$; *(commutativity)*
  - *(c)* $f(f(x, y), z) = f(x, f(y, z))$. *(associativity)*
- *An operation f is said to be a* near-unanimity *operation if it satisfies the following identities*

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

- *A ternary operation f is said to be a* Mal'tsev *operation if it satisfies the following identities*

$$f(x, y, y) = f(y, y, x) = x.$$

- *An n-ary operation f is said to be* conservative *if $f(x_1, \ldots, x_n) \in \{x_1, \ldots, x_n\}$ for all $x_1, \ldots, x_n$.*

Theorem 6 *Let $\Gamma \subseteq R_k$ be a constraint language.*

If $\mathsf{Pol}(\Gamma)$ *contains one of the following operations then* CSP(Γ) *is tractable: a constant operation [24]; a semilattice operation [24]; a near-unanimity operation [24]; a Mal'tsev operation [6]; or a binary commutative conservative operation [8].*

If $\mathsf{Pol}(\Gamma)$ *contains essentially unary surjective operations only, then* CSP(Γ) *is* **NP***-complete [24].*

Example 12 Recall the binary disequality relation $\neq_{E_k}$ defined in Example 4. It is straightforward to check that $\mathsf{Pol}(\{\neq_{E_2}\})$ contains the unique ternary near-unanimity operation on E_2. Hence, by Theorem 6, CSP($\{\neq_{E_2}\}$), which corresponds to the GRAPH 2-COLORABILITY problem, is tractable.

However, it is shown in [39] that for any $k \geq 3$, $\mathsf{Pol}(\{\neq_{E_k}\})$ contains essentially unary surjective operations only. Hence, by Theorem 6, CSP($\{\neq_{E_k}\}$), which corresponds to the GRAPH k-COLORABILITY problem, is **NP**-complete for any $k \geq 3$. □

Example 13 Let k be a prime number, and consider E_k as a finite field. For any set of relations Γ over E_k, where each $\rho \in \Gamma$ is definable by a linear equation, $\mathsf{Pol}(\Gamma)$ will contain the Mal'tsev operation f given by

$$f(x, y, z) = x - y + z.$$

Hence, by Theorem 6, CSP(Γ), which corresponds to the LINEAR EQUATIONS problem, is tractable. □

Example 14 Recall the relations χ and ξ over E_3 defined in Examples 6 and 9. It is straightforward to check that $\mathsf{Pol}(\{\chi\})$ contains the constant operation with value 0, hence, by Theorem 6, CSP($\{\chi\}$) is tractable. Moreover, $\mathsf{Pol}(\{\xi\})$ contains the semilattice operation $\max : E_3^2 \to E_3$, which returns the maximum of its two arguments. Hence, by Theorem 6, CSP($\{\xi\}$) is tractable.

However, it was shown in Example 9 that $\mathsf{Pol}(\{\chi, \xi\})$ contains only projections. Hence, by Theorem 6, CSP($\{\chi, \xi\}$) is **NP**-complete. □

We remark that the technique discussed in this paper is based on clone theory, but it has been extended in [6, 7, 9, 10, 15] to involve more powerful machinery from universal algebra [21, 43]. Using this approach, the following result was proved in [5].

Theorem 7 *Let Γ be an arbitary constraint language over E_3. Then* CSP(Γ) *is either tractable or* **NP***-complete.*

In fact, [5] contains a precise description of all tractable and **NP**-complete constraint languages over E_3, and an algorithm for distinguishing them; we refer the reader to that paper for details.

Finally, we consider what is known in the non-Boolean case about two of the related constraint problems described in Section 2: the Counting Problem and the Quantified Problem. For both of these problems it has been shown that the complexity of the parameterized version depends entirely on the polymorphisms of the constraint language.

We denote the problem of counting the number of solutions to an instance of CSP(Γ) by #CSP(Γ). The following result was obtained in [7].

Theorem 8 *For any constraint language $\Gamma \subseteq R_k$, and any finite constraint language $\Gamma_0 \subseteq R_k$, if $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Gamma_0)$, then #CSP($\Gamma_0$) is reducible to #CSP($\Gamma$) in polynomial time.*

Using this result it has been shown that Mal'tsev operations play a key role for the complexity of the Counting Problem. The following (partial) classification result for the parameterized Counting Problem was obtained in [7].

Theorem 9 *For any constraint language $\Gamma \subseteq R_k$, if $\mathsf{Pol}(\Gamma)$ contains no Mal'tsev operations, then #CSP(Γ) is* **#P**-*complete.*

Conversely, if $k \leq 3$ and $\mathsf{Pol}(\Gamma)$ contains a Mal'tsev operation, then #CSP(Γ) is tractable.

Example 15 Recall the binary disequality relation $\neq_{E_k}$ defined in Example 4. It is straightforward to check that $\mathsf{Pol}(\{\neq_{E_2}\})$ contains the Mal'tsev operation on E_2 which returns the sum of its 3 arguments modulo 2. Hence, by Theorem 9, it is tractable to count the number of solutions to CSP($\{\neq_{E_2}\}$), which corresponds to the GRAPH 2-COLORABILITY problem.

However, it is shown in [39] that for any $k \geq 3$, $\mathsf{Pol}(\{\neq_{E_k}\})$ does not contain any Mal'tsev operations. Hence, by Theorem 6, for any $k \geq 3$ it is **#P**-complete to count the number of solutions to CSP($\{\neq_{E_k}\}$), which corresponds to the GRAPH k-COLORABILITY problem. ☐

We denote the problem of deciding the truth of a quantified Boolean formula involving only conjunctions of relations from Γ by QCSP(Γ). For this problem, it has recently been shown that the complexity is determined by the *surjective* polymorphisms. We denote the set of all surjective polymorphisms of a constraint language Γ by $\mathsf{s\text{-}Pol}(\Gamma)$. The following result was obtained in [3].

Theorem 10 *For any constraint language $\Gamma \subseteq R_k$, and any finite constraint language $\Gamma_0 \subseteq R_k$, if $\mathsf{s\text{-}Pol}(\Gamma) \subseteq \mathsf{s\text{-}Pol}(\Gamma_0)$, then QCSP($\Gamma_0$) is reducible to QCSP($\Gamma$) in polynomial time.*

Using this result, the following (partial) classification result was obtained in [3].

Theorem 11 *For any constraint language $\Gamma \subseteq R_k$:*

- *If $\mathsf{s\text{-}Pol}(\Gamma)$ contains the affine operation $x - y + z$ of an Abelian group with base set E_k, then QCSP(Γ) is tractable.*
- *if $\mathsf{s\text{-}Pol}(\Gamma)$ contains the ternary near-unanimity operation d on E_k given by*

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

  then QCSP(Γ) is in **NL**.
- *If $k \geq 3$, and $\mathsf{s\text{-}Pol}(\Gamma)$ contains only essentially unary operations, then QCSP(Γ) is* **PSPACE**-*complete.*

Example 16 Let τ_n be the n-ary "not-all-distinct" relation over E_k consisting of all tuples $(a_1, \ldots, a_n)$ such that $|\{a_1, \ldots, a_n\}| < n$. Note that $\tau_n \supseteq \{(a, \ldots, a) \mid a \in E_k\}$, so every instance of the standard CSP problem CSP($\{\tau_n\}$) is trivially satisfiable by assigning the same value to all variables.

However, by Lemma 2.2.4 of [39], the set $\mathsf{Pol}(\{\tau_k\})$ consists of all non-surjective operations on E_k, together with all operations of the form $f(x_1, \ldots, x_n) = g(x_i)$ for some $1 \leq i \leq n$ and some permutation g on E_k. This implies that $\mathsf{s\text{-}Pol}(\{\tau_k\})$ contains essentially unary operations only. Hence, by Theorem 11, QCSP($\{\tau_k\}$) is **PSPACE**-complete. Similar arguments can be used to show that QCSP($\{\tau_i\}$) is **PSPACE**-complete, for any i in the range $3 \leq i \leq k$. ☐

References

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

[2] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for Boolean constraint satisfaction. In *Proceedings 16th International Workshop on Computer Science Logic, CSL'02*, volume 2471 of *Lecture Notes in Computer Science*, pages 412–426. Springer-Verlag, 2002.

[3] F. Börner, A. Bulatov, P. Jeavons, and A. Krokhin. Quantified constraints and surjective polymorphisms. Technical Report PRG-RR-02-11, Computing Laboratory, University of Oxford, Oxford, UK, 2002.

[4] A. Bulatov. Finite sublattices in a lattice of clones. *Algebra i Logika*, 33(5):514–549, 1994. (in Russian).

[5] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings 43rd IEEE Symposium on Foundations of Computer Science, FOCS'02*, pages 649–658, 2002.

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

[7] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. Submitted for publication.

[8] A. Bulatov and P. Jeavons. Tractable constraints closed under a binary operation. Technical Report PRG-TR-12-00, Computing Laboratory, University of Oxford, Oxford, UK, 2000.

[9] A. Bulatov and P. Jeavons. Algebraic approach to multi-sorted constraints. Technical Report PRG-RR-01-18, Computing Laboratory, University of Oxford, Oxford, UK, 2001.

[10] A. Bulatov, A. Krokhin, and P. Jeavons. Constraint satisfaction problems and finite algebras. In *Proceedings 27th International Colloquium on Automata, Languages and Programming, ICALP'00*, volume 1853 of *Lecture Notes in Computer Science*, pages 272–282. Springer-Verlag, 2000.

[11] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12, 1996.

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

[13] P. Crescenzi and G. Rossi. On the Hamming distance of constraint satisfaction problems. *Theoretical Computer Science*, 288(1):85–100, 2002.

[14] 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.

[15] V. Dalmau. A new tractable class of constraint satisfaction problems. In *Proceedings 6th International Symposium on Artificial Intelligence and Mathematics*, 2000.

[16] N. Dunkin, J. Bater, P. Jeavons, and D. 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.

[17] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal of Computing*, 28:57–104, 1998.

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

[19] G. Gottlob, L. Leone, and F. Scarcello. Hypertree decomposition and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

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

[21] D. Hobby and R. McKenzie. *The Structure of Finite Algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, Providence, R.I., 1988.

[22] P. Jeavons. Constructing constraints. In *Proceedings 4th International Conference on Constraint Programming—CP'98 (Pisa, October 1998)*, volume 1520 of *Lecture Notes in Computer Science*, pages 2–16. Springer-Verlag, 1998.

[23] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

[24] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[25] P. Jeavons, D. Cohen, and M. Gyssens. How to determine the expressive power of constraints. *Constraints*, 4:113–131, 1999.

[26] H. Kautz and B. Selman. Planning as satisfiability. In *Proceedings 10th European Conference on Artificial Intelligence, ECAI'92*, pages 359–363, 1992.

[27] D. Kavvadias and M. Sireni. The inverse satisfiability problem. *SIAM Journal on Computing*, 28(1):152–163, 1998.

[28] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001.

[29] L. Kirousis and P. Kolaitis. The complexity of minimal satisfiability problems. In *Proceedings 18th International Symposium on Theoretical Aspects of Computer Science, STACS'01*, volume 2010 of *Lecture Notes in Computer Science*, pages 407–418. Springer-Verlag, 2001.

[30] P. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.

[31] A. Mackworth. Constraint satisfaction. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, volume 1, pages 285–293. Wiley Interscience, 1992.

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

[33] R. McKenzie, G. McNulty, and W. Taylor. *Algebras, Lattices and Varieties*, volume I. Wadsworth and Brooks, California, 1987.

[34] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[35] B. Nadel. Constraint satisfaction in Prolog: complexity and theory-based heuristics. *Information Sciences*, 83(3-4):113–131, 1995.

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

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

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

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

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

[41] I. Rosenberg. Completeness, closed classes and relations in multiple-valued logics. In *Proceedings 4th International Symposium on Multiple-Valued Logic, ISMVL'74*, pages 1–26, 1974.

[42] T. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th ACM Symposium on Theory of Computing, STOC'78*, pages 216–226, 1978.

[43] A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathematiques Superieures*. University of Montreal, 1986.

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

[45] P. van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.