Symmetries and Complexity

Andrei A. Bulatov
School of Computing Science, Simon Fraser University, Burnaby, Canada

Abstract

The Constraint Satisfaction Problem (CSP) and a number of problems related to it have seen major advances during the past three decades. In many cases the leading driving force that made these advances possible has been the so-called algebraic approach that uses symmetries of constraint problems and tools from algebra to determine the complexity of problems and design solution algorithms. In this presentation we give a high level overview of the main ideas behind the algebraic approach illustrated by examples ranging from the regular CSP, to counting problems, to optimization and promise problems, to graph isomorphism.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Design and analysis of algorithms

Keywords and phrases constraint problems, algebraic approach, dichotomy theorems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.2

Category Invited Talk

Funding *Andrei A. Bulatov*: This research was supported by an NSERC Discovery grant.

As Jeavons et al. [39] discovered in the mid 90s, symmetries or lack thereof of combinatorial structures in many cases determine the complexity of the corresponding computational problems. This was the starting point of the so-called algebraic approach that has been initially developed for certain kinds of Constraint Satisfaction Problems (CSPs), and over the course of 20 years has been instrumental in resolving a number of long standing open problems, most important of which is the CSP Dichotomy Conjecture by Feder and Vardi [31]. These techniques also spread out and found applications in multiple areas somewhat related to the CSP. The types of research problems the algebraic approach has been most useful include complexity classifications and design of algorithms. It clearly does not apply to every single kind of CSP-related problems, but whenever the algebraic approach is possible, it has led to a significant progress in the area.

In this presentation we take a bird's-eye view on the main ideas behind the algebraic approach. We do not go into deep technical details, although we give links that can be used by an interested reader, but give a collection of simple examples showing how algebraic techniques can be used in various types of computational problems. In the first part of the presentation, Sections 1,2 we introduce several common types of constraint problems and lay out the basics of the algebraic approach. Then in Sections 3,4 we show how these ideas apply to the CSP Dichotomy Conjecture, and also briefly outline how the algebraic approach works for Counting, Promise, and Valued CSPs. We also mention a somewhat unexpected use of the method in Graph Isomorphism. Although there has been developed a rich and beautiful theory of CSPs on infinite domains [6, 7], we only focus here on finite domains. Also, we leave out many areas were the approach has been successfully used: Quantified CSPs [45], homomorphism lower bounds [41], robust approximation [3], proof complexity [1], global cardinality constraints [20, 21], Subalgebra [12] and Ideal Membership Problems [46, 22], solvability of equations [42], learnability [27], property testing [26], and many others.


© Andrei A. Bulatov;
licensed under Creative Commons License CC-BY 4.0

1 Constraint Problems

1.1 The problem

There are multiple ways to define the CSP. For our purpose here we give two equivalent definitions [39, 43]. We use the terms *predicate* and *relation* interchangeably.

The first one, we refer to it as the *logic* definition, goes as follows. Fix a set A (always finite in this paper), a *domain*. The input of the CSP is a collection $\mathcal{I}$ of constraints, $R(x_1, \ldots, x_k)$, expressed as predicates over A, where the variables x_i are from some finite set X of variables. The goal is to decide whether a *solution* exists, that is, a mapping $\varphi : X \to A$ that satisfies all the constraints. Alternatively, the input can be thought of as the conjunction of all the constraints, and we want to know whether this formula is satisfiable.

The second definition of the CSP, which we will refer to as the *homomorphic* definition, is given in terms of relational structures. Recall that a relational *signature* is a collection of *relational symbols*, that is, names of relations we are going to use, each symbol R is assigned a natural number k_R, the *arity* of R. For instance, the relational signature of a graph or digraph is $\{E\}$, only one relational symbol, whose arity is 2. For a signature σ a *relational structure* $\mathcal{A}$ with signature σ is a set A along with a predicate $R^{\mathcal{A}} \subseteq A^{k_R}$ for each $R \in \sigma$, called an *interpretation* of R. The set A is called the *base set* of $\mathcal{A}$. Structures with signature σ are often referred to as *σ-structures* and structures with the same signature are called *similar*. Let $\mathcal{A}, \mathcal{B}$ be two σ-structures with base sets A, B, respectively. A *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a mapping $\varphi : A \to B$ such that $R^{\mathcal{B}}(\varphi(a_1), \ldots, \varphi(a_{k_R})) = 1$ for every $R \in \sigma$ and any $a_1, \ldots, a_{k_R} \in A$ with $R^{\mathcal{A}}(a_1, \ldots, a_{k_R}) = 1$. If there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$, we write $\mathcal{A} \to \mathcal{B}$.

In the homomorphic version of the CSP the input is a pair of similar relational structures $\mathcal{A}, \mathcal{B}$ (always finite in this paper). The question is to decide whether there exists a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. A translation between the two versions of the CSP is straightforward, it will be illustrated in Example 2 below.

▶ **Example 1.** In the 3-SAT problem the question is to decide the satisfiability of a 3-CNF. 3-SAT is readily a CSP in the logic form, it is a conjunction of clauses, each of which represents a ternary predicate on $\{0, 1\}$.

▶ **Example 2.** In the 3-Coloring problem we need to decide the existence of a proper 3-coloring of a given graph G. It can be stated as a CSP by treating the vertices of G as variables that need to be assigned one of the three colors, and edges of G as constraints requiring that if $uv \in E(G)$ then the values of u, v satisfy the predicate $\neq_3 (u, v)$, which is the disequality relation on the set of colors.

Note that 3-Coloring can also be naturally represented in the homomorphic form: Any proper 3-coloring of G is a homomorphism from G to K_3. Using this approach one can generalize 3-Coloring to H-Coloring, where H is a fixed graph. The goal in this problem is to decide the existence of a homomorphism from a given graph G to H.

This transition between the two forms of the CSP can be extended to more general case: The structure $\mathcal{A}$ in the homomorphic form encodes the interaction between constraints, while the structure $\mathcal{B}$ encodes the constraints themselves.

▶ **Example 3.** A system of linear (as well as any other type of equations) naturally provides a conjunction of constraints, in which every constraint is represented by an equation.

▶ **Example 4.** The Clique problem, in which we are given a number k and a graph G, asks whether or not G contains a clique of size at least k. This problem can be reformulated as the question about the existence of a homomorphism from K_k to G, that is, a CSP in the homomorphic form.

▶ **Example 5.** The Perfect Matching problem asks whether a graph G has a perfect matching It is representable as a CSP in the logic form, although in a slightly more sophisticated way. The edges of G represent the variables of our CSP instance $\mathcal{I}(G)$ that take values 1 or 0 (in a matching or not), and the vertices correspond to constraints. For each $v \in V(G)$ of degree d we introduce a constraint of the form $R_{1\text{-in-}d}(x_1, \dots, x_d)$, where $x_1, \dots, x_d$ are the edges incident to v, which is true if and only if exactly one of $x_1, \dots, x_d$ equals 1 and the rest equal 0. As is easily seen, any solution of $\mathcal{I}(G)$ corresponds to a perfect matching in G and vice versa.

In order to capture specific computational problems the general CSP is often restricted in a certain way. It can be easily done for the homomorphic version of the CSP [43]. Let $\mathfrak{A}, \mathfrak{B}$ be classes of similar structures with signature σ. Then $\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ denotes the class of CSP instances $\mathcal{A}, \mathcal{B}$, in which $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{B} \in \mathfrak{B}$. If $\mathfrak{A}$ or $\mathfrak{B}$ is the class of all σ-structures, we use $\mathrm{CSP}(-, \mathfrak{B})$ and $\mathrm{CSP}(\mathfrak{A}, -)$, respectively. If one of the classes contains only one structure, we write $\mathrm{CSP}(-, \mathcal{B}), \mathrm{CSP}(\mathcal{A}, -)$ instead of $\mathrm{CSP}(-, \{\mathcal{B}\}), \mathrm{CSP}(\{\mathcal{A}\}, -)$. The general CSP is NP-complete as the examples above show, and assuming the Exponential Time Hypothesis it cannot be solved faster than $|\mathcal{B}|^{O(|\mathcal{A}|)}$ [33]. However, restricted problems may have much lower complexity, and this is the drive to understand the complexity of restricted problems that has been guiding the study of the CSP.

▶ **Example 6.** All the problems from Examples 1–5 can be viewed as $\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ for appropriate classes $\mathfrak{A}, \mathfrak{B}$.

- 3-SAT is the problem $\mathrm{CSP}(-, \mathcal{B}_{3\text{-SAT}})$ where $\mathcal{B}_{3-SAT}$ is the relational structure with base set $\{0, 1\}$ and 8 predicates that are defined by the 8 possible 3-clauses.
- 3-Coloring is the problem $\mathrm{CSP}(-, K_3)$. More generally, the H-Coloring problem for a graph or digraph H can be represented as $\mathrm{CSP}(-, H)$.
- Representing Linear Equations requires a relational structure with an infinite signature: a predicate symbol for each possible linear equation. It is therefore a common practice to first observe that every system of linear equations is equivalent to one in which every equation contains at most 3 variables; it may require introducing new variables. Then in the case of a finite field $\mathbb{F}$ such a problem is the same as $\mathrm{CSP}(-, \mathcal{B}_{3\text{-Lin}})$, where $\mathcal{B}_{3\text{-Lin}}$ is the relational structure with the base set $\mathbb{F}$ whose predicates are given by all the possible linear equations over $\mathbb{F}$ containing at most 3 variables.
- The Clique problem is $\mathrm{CSP}(\mathfrak{K}, -)$, where $\mathfrak{K}$ is the class of all cliques.
- The Perfect Matching problem is a bit more difficult to represent. Let σ be an (infinite) signature that contains one symbol $R_{1\text{-in-}d}$ for every natural d. Then Perfect Matching is equivalent to $\mathrm{CSP}(\mathfrak{A}_2, \mathcal{B}_{1\text{-in}})$, where $\mathfrak{A}_2$ is the class of σ-structures $\mathcal{A}$ in which every element of the base set appears in exactly two tuples from the predicates of $\mathcal{A}$. Then $\mathcal{B}_{1\text{-in}}$ is the σ-structure with the base set $\{0, 1\}$ and whose predicates are interpreted as in Example 5. Note that the Perfect Matching problem can be expressed more naturally as a *holant* problem [24].

In this paper we focus on the problems of the form $\mathrm{CSP}(-, \mathcal{B})$, which are often referred to as *nonuniform* CSPs. We will shorten the notation to $\mathrm{CSP}(\mathcal{B})$. In this case it is often convenient to replace the structure $\mathcal{B}$ with a *constraint language*, the set of relations given by the predicates of $\mathcal{B}$. For any collection Γ of relations over a set A such a problem is denoted by $\mathrm{CSP}(\Gamma)$. The base set of $\mathcal{B}$ or the set on which Γ is defined on will be called the *domain* of $\mathrm{CSP}(\mathcal{B})$ or $\mathrm{CSP}(\Gamma)$.

1.2 Friends of the CSP

The CSP is not limited to just the decision problem from the previous section. It can also be modified to include other types of problems, some of which we will consider next. Most of the variations below were considered in [28] in the case of a 2-element domain.

Quantified CSP

The CSP in the logic form can also be thought of as checking the validity of an existentially quantified sentence $\exists x_1, \ldots, x_n(R_1 \wedge \cdots \wedge R_k)$. The problem that allows for an arbitrary quantifier prefix is known as the Quantified CSP or QCSP for short. Examples of QCSPs are the Quantified Satisfiability problem (QSAT) as well as a number of standard problems in PSPACE. For a structure $\mathcal{B}$ or a constraint language Γ, $\mathrm{QCSP}(\mathcal{B}), \mathrm{QCSP}(\Gamma)$ denote the Quantified CSPs restricted in the same way as the regular CSP [28, 45].

Counting CSPs

In the Counting CSP, or #CSP, the goal is to find the number of solutions of a CSP instance. In general, such problems belong to the class #P and many of them are #P-complete. Counting CSPs restricted by specifying a relational structure or a constraint language are denoted by $\#\mathrm{CSP}(\mathcal{B}), \#\mathrm{CSP}(\Gamma)$.

Optimization problems

There are several ways to convert a CSP into an optimization problem, see [28] for some examples. The most natural one is, given a CSP instance that may have no solution, find an assignment of variables that maximizes the number of satisfied constraints. This optimization problem is called the Max-CSP. Clearly, even when $\mathrm{CSP}(\Gamma)$ can be solved efficiently, the optimization problem may be hard. Linear Equations provides a well known example of such problem. More examples will be mentioned in Section 4.2 and can be found in [28]. The book [28] also presents the range of possible complexities of Max-CSPs on the domain $\{0, 1\}$.

Another way to optimize a CSP instance is to look for a solution that assigns a specific value to a maximal number of variables. For instance, Max-Ones is the version of SAT that seeks for an assignment in which as many propositional variables as possible are assigned 1.

Valued CSPs

The Valued CSP, or the VCSPs for short, is a generalization of optimization constraint problems such as Max-CSP and Max-Ones. In order to introduce them we need to replace predicates in the regular CSP with more general functions. Let A be a domain and $\mathbb{K}$ an ordered semiring, e.g. the ring of natural, integer, rational, or real numbers. Instead of predicates we now consider functions $R : A^k \to \mathbb{K}$ for some k. An instance of the Valued CSP consists of a set X of variables and a collection $\mathcal{C}$ of *function applications* of the form $R(x_1, \ldots, x_k)$, where each R is as above. For a mapping $\varphi : X \to A$ define its *weight* to be $w(\varphi) = \sum_{R(x_1,\ldots,x_k)\in\mathcal{C}} R(\varphi(x_1), \ldots, \varphi(x_k))$. The goal now is to find a mapping φ that yields the maximal (or minimal) weight possible. Similar to regular CSPs, VCSPs can also be parametrized by a constraint language, which in this case is a set of functions that are allowed in VCSP instances.

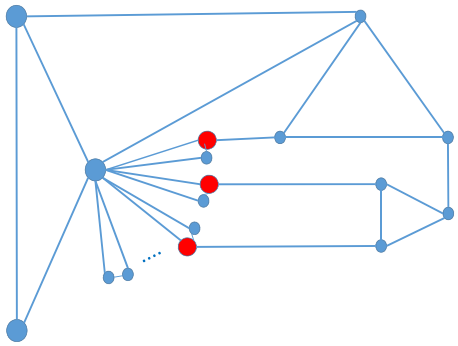

Figure 1 The gadget used to reduce 3-SAT to 3-Coloring.

▶ **Example 7** (Max-CSP as a VCSP)**.** Let Γ be a constraint language, that is, a set of predicates on some set A. We view the predicates from Γ as functions from Cartesian powers of A to $\mathbb{N}$. They take values $0, 1$. This way every instance of CSP(Γ) is treated as an instance of VCSP(Γ). As is easily seen, a mapping that maximizes the number of satisfied constraints in a CSP instance also has the maximal weight in the corresponding instance of the VCSP.

▶ **Example 8** (Max-Ones as a VCSP)**.** Transforming the Max-Ones problem into a VCSP is less straightforward, because we need to make sure that optimization is only happening over solutions of a Max-Ones instance. This is achieved by adding the *infinity* elements $-\infty, \infty$ to $\mathbb{N}$. In terms of order and arithmetic operations they relate to other elements of $\mathbb{N}$ in the natural way. Let Γ be a constraint language on $\{0,1\}$, and let $\Gamma' = \{R' \mid R \in \Gamma\} \cup \{O\}$, where $O : \{0,1\} \to \mathbb{N}$ with $O(0) = 0, O(1) = 1$, and $R'(a_1, \dots, a_k) = 1$ when $R(a_1, \dots, a_k) = 1$, $a_1, \dots, a_k \in \{0,1\}$, and $R'(a_1, \dots, a_k) = -\infty$ otherwise. Take an instance $\mathcal{I}$ of Max-Ones(Γ), replace every predicate $R(x_1, \dots, x_k)$ with a function application $R'(x_1, \dots, x_k)$, and for each variable $x \in X$ of $\mathcal{I}$ add the function application $O(x)$. Denote the resulting instance by $\mathcal{I}'$, it is an instance of VCSP(Γ'). The weight of a mapping $\varphi : X \to \{0,1\}$ in $\mathcal{I}'$ is not negative infinite if and only if φ is a solution of $\mathcal{I}$. Moreover, if w is a solution of $\mathcal{I}$ we have $w(\varphi) = m + \ell$, where m is the number of constraints in $\mathcal{I}$ that does not depend on φ, and ℓ is the number of variables that are assigned 1 by φ. Thus φ maximizes the number of ones if and only if it maximizes the weight in $\mathcal{I}'$.

2 Reductions and symmetries

In this section we look at the formalism that includes primitive-positive definitions and interpretations, and that captures one of the oldest tools in complexity theory, gadget reductions. We then introduce higher level symmetries of problems, polymorphisms, that set boundaries of what can be done using gadget reductions. Finally, we give some examples showing that polymorphisms can also be useful when designing solution algorithms. A more detailed and technical exposition can be found in [4].

2.1 Gadget reductions and primitive-positive definitions

A usual gadget reduction known from a basic course in complexity looks somewhat like what is shown in Fig. 1. They may be complicated and often difficult to come up with. Can we make the process of constructing such gadgets more orderly?

Let Γ be a set of relations (predicates) over a set A. A predicate R over A is said to be *primitive-positive (pp-) definable* in Γ if $R(\mathbf{x}) = \exists \mathbf{y}\ \Phi(\mathbf{x}, \mathbf{y})$, where Φ is a conjunction that involves predicates from Γ and equality relations. The formula above is then called a *pp-definition* of R in Γ. A constraint language Δ is pp-definable in Γ if so is every relation from Δ. In a similar way pp-definability can be introduced for relational structures.

▶ **Example 9.** Let $K_3 = (\{0, 1, 2\}, E)$ be a 3-element complete graph. Its edge relation is the binary disequality relation $\neq_3$ on $\{0, 1, 2\}$. Then the pp-formula

$$\begin{aligned} Q(x, y, z) \quad = \quad & \exists t, u, v, w (E(t, x) \wedge E(t, y) \wedge E(t, z) \wedge E(u, v) \wedge E(v, w) \\ & \wedge E(w, u) \wedge E(u, x) \wedge E(v, y) \wedge E(w, z)) \end{aligned}$$

defines the predicate Q that is true on all triples containing exactly 2 different elements from $\{0, 1, 2\}$.

A link between pp-definitions and reducibility of nonuniform CSPs was first observed in [39].

▶ **Theorem 10** ([39]). *Let Γ and Δ be constraint languages and Δ finite. If Δ is pp-definable in Γ then* CSP(Δ) *is polynomial time reducible*[1] *to* CSP(Γ).

The gadget in Fig. 1 can be represented by a pp-formula, in which every variable corresponds to a vertex in the graph, the large red vertices are the free variables, and the edges are the constraints $\neq_3$. In general it seems plausible that pp-definitions and pp-interpretations discussed later capture what we think of "gadgets" and "gadget reductions".

2.2 Polymorphisms

While pp-definitions is a convenient and uniform way of representing gadgets, it is polymorphisms that are at the core of the algebraic approach.

Primitive positive definability can be concisely characterized using polymorphisms. An operation $f : A^k \to A$ is said to be a *polymorphism* of a relation $R \subseteq A^n$ if for any $\mathbf{a}_1, \ldots, \mathbf{a}_k \in R$, $\mathbf{a}_i = (a_{i1}, \ldots, a_{in})$, the tuple $f(\mathbf{a}_1, \ldots, \mathbf{a}_k)$ also belongs to R, where $f(\mathbf{a}_1, \ldots, \mathbf{a}_k)$ stands for $(f(a_{11}, \ldots, a_{k1}), \ldots, f(a_{1n}, \ldots, a_{kn}))$. The parameter k above is called the *arity* of f. Relation R is said to be *invariant* under f. Operation f is a polymorphism of a constraint language Γ if it is a polymorphism of every relation from Γ. Similarly, operation f is a polymorphism of a relational structure $\mathcal{B}$ if it is a polymorphism of every relation of $\mathcal{B}$. The set of all polymorphisms of language Γ or relational structure $\mathcal{B}$ is denoted by $\mathsf{Pol}(\Gamma)$, $\mathsf{Pol}(\mathcal{B})$, respectively.

▶ **Example 11.** Let R be an affine relation, that is, R is the solution space of a system of linear equations over a field $\mathbb{F}$. Then the operation $f(x, y, z) = x - y + z$, where $+, -$ are operations of $\mathbb{F}$, is a polymorphism of R. Indeed, let $A \cdot \mathbf{x} = \mathbf{b}$ be the system defining R, and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in R$. Then

$$A \cdot f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = A \cdot (\mathbf{x} - \mathbf{y} + \mathbf{z}) = A \cdot \mathbf{x} - A \cdot \mathbf{y} + A \cdot \mathbf{z} = \mathbf{b}.$$

In fact, the converse can also be shown: if R is invariant under f, where f is defined in a certain finite field $\mathbb{F}$, then R is the solution space of some system of linear equations over $\mathbb{F}$.

[1] In fact, due to the result of [47] this reduction can be made log-space.

Polymorphisms can be viewed as a generalization of homomorphisms of relational structures. We use $[k]$ to denote the set $\{1,\dots,k\}$. For a structure $\mathcal{B}$ with signature σ and $k \geq 1$, let $\mathcal{B}^k$ denote the following relational structure. The base set of $\mathcal{B}^k$ is B^k, where B is the base set of $\mathcal{B}$. For every symbol $R \in \sigma$, say, ℓ-ary, the predicate $R^{\mathcal{B}^k}$ is given by $(\mathbf{a}_1,\dots,\mathbf{a}_\ell) \in R^{\mathcal{B}^k}$ if and only if $(a_{1i},\dots,a_{\ell i}) \in R^{\mathcal{B}}$ for each $i \in [k]$, where $\mathbf{a}_j = (a_{j1},\dots,a_{jk})$. Then a mapping $f : B^k \to B$ is a polymorphism of $\mathcal{B}$ if and only if f is a homomorphism of $\mathcal{B}^k$ to $\mathcal{B}$.

A link between polymorphisms and pp-definability of relations is given by *Galois connection*.

▶ **Theorem 12** (Galois connection, [8, 34]). *Let Γ be a constraint language on A, and let $R \subseteq A^n$ be a non-empty relation. Then R is preserved by all polymorphisms of Γ if and only if R is pp-definable in Γ.*

Every relation on a set A has projections as polymorphisms. A *projection* is an operation f, say, k-ary, such that for some $i \in [k]$, $f(x_1,\dots,x_k) = x_i$. Theorem 12 means, in particular, that if a constraint language Γ on A does not have polymorphisms other than the projections, every relation is pp-definable in Γ.

The following statement is what makes the algebraic approach possible.

▶ **Corollary 13.** *Let Γ and Δ be constraint languages and Δ finite. If $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$ then $\mathrm{CSP}(\Delta)$ is polynomial time reducible to $\mathrm{CSP}(\Gamma)$.*

For our next example we need another more technical result.

▶ **Proposition 14** ([19]). *Let Γ be a constraint language on a set A. Then either*

1. *there is a non-injective unary polymorphism f of Γ and $\mathrm{CSP}(\Gamma)$ is polynomial time interreducible with $\mathrm{CSP}(f(\Gamma))$, where $f(\Gamma) = \{f(R) \mid R \in \Gamma\}$, $f(R) = \{f(\mathbf{a}) \mid \mathbf{a} \in R\}$; or*
2. *all unary polymorphisms of Γ are injective and $\mathrm{CSP}(\Gamma)$ is polynomial time interreducible with $\mathrm{CSP}(\Gamma^*)$, where $\Gamma^* = \Gamma \cup \{R_a \mid a \in A\}$, $R_a = \{(a)\}$ is a* constant *relation, i.e. a unary relation that contains only one tuple.*

Constraint languages that contain all the constant relations are called *idempotent*.

▶ **Example 15.** We revisit the reducibility of 3-SAT to 3-Coloring. Let $\Gamma = \{\neq_3\}$ be the constraint language consisting of just the disequality relation $\neq_3$ on the 3-element set $\{0,1,2\}$. As we noted in Example 2, $\mathrm{CSP}(\Gamma)$ is equivalent to 3-Coloring. Let also Γ_{3-SAT} denote the constraint language consisting of the 8 ternary relations represented by 3-clauses. Then $\mathrm{CSP}(\Gamma_{3-SAT})$ is equivalent to 3-SAT. We show that Γ_{3-SAT} is pp-definable in Γ^*. By Theorem 10 and Proposition 14 it implies that 3-SAT is reducible to 3-Coloring.

First, we prove that every polymorphism of $\neq_3$ has only one *essential variable*, that is, a variable such that the value of the function can be changed by changing the value of this variable only. Suppose there is $f \in \mathsf{Pol}(\neq_3)$ that has at least two essential variables. To save on notation we assume that $f(x,y)$ is binary. Assume first that for some $a,a',b,b' \in \{0,1,2\}$ we have $f(a,b) \neq f(a',b), f(a,b) \neq f(a,b')$. Without loss of generality, let $a = b = 0, a' = b' = 1$. Then since $(0,1),(1,0) \in \neq_3$ and f is a polymorphism of $\neq_3$, we also have $f(0,1) \neq f(1,0)$. For a similar reason $f(2,2)$ is not equal to any of $f(0,0), f(0,1), f(1,0)$, which is impossible. Next, as y is an essential variable in $f(x,y)$ there are $a,b,b' \in \{0,1,2\}$ such that $f(a,b) \neq f(a,b')$. Again, let us assume $a = b = 0, b' = 1$. If $f(0,0) \neq f(1,0)$, or $f(0,0) \neq f(2,0)$, or $f(0,1) \neq f(1,1)$, or $f(0,1) \neq f(2,1)$, we have the previous case. Otherwise $f(0,2) \neq f(1,0) = f(0,0), f(0,2) \neq f(1,1) = f(0,1)$, because $(0,1),(2,0),(2,1) \in \neq_3$ and f is a polymorphism of $\neq_3$. By the same argument, either we have the previous case, or $f(0,2) = f(1,2) = f(2,2)$ implying that x is not essential variable in $f(x,y)$, a contradiction.

Second, if $f \in \mathsf{Pol}(\neq_3)$ has only one essential variable, we have $f(x_1, \dots, x_k) = g(x_i)$ for some operation g. Since f is a polymorphism of each constant relation R_a we get $g(a) = f(a, \dots, a) = a$, that is, f is a projection. Thus, the only polymorphisms of Γ^* are projections. By Theorem 12 every relation on $\{0, 1, 2\}$ is pp-definable in Γ^*. This includes the relations from Γ_{3-SAT}. By Theorem 10 $\mathrm{CSP}(\Gamma_{3-SAT})$ is polynomial time reducible to $\mathrm{CSP}(\Gamma)$.

Example 15 gives a reduction and a hardness proof only in one very special case. However, we shall see soon that together with pp-interpretability introduced next this technique gives a hardness proof for all NP-complete CSPs of the form $\mathrm{CSP}(\Gamma)$.

Let Γ, Δ be constraint languages over sets A, B, respectively. We say that Γ *pp-interprets* Δ if there exists a natural number ℓ, a set $F \subseteq A^\ell$, and an onto mapping $\pi : F \to B$ such that Γ pp-defines the following relations

1. the relation F,
2. the π-preimage of the equality relation on B, and
3. the π-preimage of every relation in Δ,

where by the π-preimage of a k-ary relation R on B we mean the ℓk-ary relation $\pi^{-1}(R)$ on A defined by

$$\pi^{-1}(R)(x_{11}, \dots, x_{1k}, x_{21}, \dots, x_{2k}, \dots, x_{\ell 1}, \dots, x_{\ell k}) \qquad \text{is true}$$

if and only if

$$R(\pi(x_{11}, \dots, x_{\ell 1}), \dots, \pi(x_{1k}, \dots, x_{\ell k})) \qquad \text{is true.}$$

If $\ell = 1$ in this definition, we say that Γ *1-pp-interprets* Δ.

▶ **Theorem 16** ([19, 4]). *Let Γ and Δ be constraint languages and Δ finite. If Δ is pp-interpretable in Γ then $\mathrm{CSP}(\Delta)$ is polynomial time reducible to $\mathrm{CSP}(\Gamma)$.*

▶ **Example 17.** We demonstrate how to use pp-interpretability in a small special case of the H-Coloring problem, see Example 2. Consider graph H on the left hand side of Fig. 2, we show that H-Coloring for this graph is NP-complete. In order to do that we will 1-pp-interpret $\neq_3$ in the constraint language $\Gamma = \{E = E(H)\}$. In the definition of pp-interpretation above we set $F = A = \{0, 0', 1, 2\}$, $B = \{0, 1, 2\}$, $\Delta = \{\neq_3\}$, $\ell = 1$, and $\pi : A \to B$ as shown in Fig. 2. The only thing we need to check is that $\pi^{-1}(B)$, $\pi^{-1}(=)$, and $\pi^{-1}(\neq_3)$ are pp-definable in H^2. The first and the last requirements are straightforward, because $\pi^{-1}(B) = A$ and $\pi^{-1}(\neq_3) = E$. For the preimage of the equality relation, the relation $\pi^{-1}(=)$ is the equivalence relation on A with classes $\{0, 0'\}, \{1\}, \{2\}$. This relation is defined by the pp-formula

$$\exists u, w (E(u, w) \wedge E(x, u) \wedge E(x, w) \wedge E(y, v) \wedge E(y, w)).$$

[2] Note that although π is a homomorphism from H to K_3, this does not yet give rise to a reduction from H-Coloring to 3-Coloring. Indeed, any graph is a homomorphic image of a collection of disconnected edges. Homomorphism π is a very special kind of homomorphism that gets along pp-definitions well.

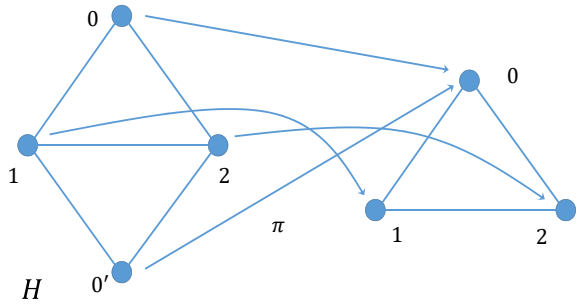

Figure 2 Pp-interpretations of 3-Coloring.

2.3 Polymorphisms and algorithms

So far we have only seen applications of polymorphisms and pp-interpretations for discovering polynomial time reductions between CSPs. In this section we mention two examples when polymorphisms also help developing efficient solution algorithms.

▶ **Example 18.** As we saw in Example 11, a relation can be represented as the set of solutions of a system of linear equations over a finite field $\mathbb{F}$ if and only if it is invariant under the affine operation $f(x, y, z) = x - y + z$, where $+, -$ are the operations of the same field $\mathbb{F}$. This means that if a constraint language Γ is invariant with respect to f then any instance of CSP(Γ) can be thought of as a system of linear equations over $\mathbb{F}$ and so can be solved by Gaussian elimination. In particular, it is possible not only to decide the existence of a solution in polynomial time, but also to construct a concise representation of the set of all solutions – a basis of the solution space.

Affine operations are a special kind of more general Maltsev operations. A ternary operation f is said to be *Maltsev* if it satisfies the equations $f(x, y, y) = f(y, y, x) = x$. The affine operation clearly satisfies them. It was shown in [16] that if a constraint language Γ has a Maltsev operation as a polymorphism, there is a polynomial time algorithm that constructs a concise representation of the set of solutions of any instance of CSP(Γ). Note that when applied to systems of linear equations this algorithm provides an alternative to Gaussian elimination. This result was further generalized in [38].

▶ **Example 19.** For this example it will be convenient to use infix notation for binary operations, that is, to write $x \cdot y$ rather than $f(x, y)$. A binary operation $\cdot$ on a set A is said to be *semilattice* if it is idempotent ($x \cdot x = x$), commutative ($x \cdot y = y \cdot x$), and associative ($x \cdot (y \cdot z) = (x \cdot y) \cdot z$), where the equations hold for all $x, y, z \in A$. A semilattice operation defines a partial order on A: $a \leq b$ if and only if $b = a \cdot b$.

Let $R \subseteq A^k$ be a relation invariant with respect to $\cdot$. Let also $R_i \subseteq A$ denote its *ith projection*, the set $R_i = \{a_i \mid (a_1, \dots, a_k) \in R\}$. As is easily seen, R_i is invariant under $\cdot$, that is, $a \cdot b \in R_i$ for any $a, b \in R_i$. Every set R_i has a greatest element in terms of the order $\leq$, it is the product of all the elements of R_i. Let us denote this element by m_i. Now, if $\mathbf{a}_1, \dots, \mathbf{a}_r$ is a list of all tuples from R_i, we have $\mathbf{a}_1 \cdot \mathbf{a}_2 \cdot \dots \cdot \mathbf{a}_r = (m_1, \dots, m_k)$. In other words, the tuple, whose entries are the greatest elements of the corresponding projections, belongs to R.

The following algorithm known as the *arc-consistency* algorithm [40] uses the discussed above properties of relations invariant under $\cdot$ to solve CSP(Γ) where Γ is any constraint language invariant under $\cdot$. Let $\mathcal{I}$ be an instance of CSP(Γ) with the set of variables X. The algorithm works in rounds until the instance cannot be further improved. Suppose that two constraints $R^1(x, \mathbf{y})$ and $R^2(x, \mathbf{z})$ from $\mathcal{I}$ share a variable. Here $\mathbf{y}, \mathbf{z}$ stand for some variables involved in R^1, R^2, and clearly x does not have to be in the same position in R^1, R^2. Let $R_1 = R^1_1 \cap R^2_1$. If $R_1 \neq R^1_1$, we remove from R^1 all tuples $(a, \mathbf{b})$ such that $a \notin R_1$. Then we repeat the procedure for R^2. After the process converges, every variable $x \in X$ has a domain $A_x \subseteq A$ such that the projection of every constraint containing x on the corresponding coordinate position equals A_x. Finally, that $\cdot$ is a polymorphism of Γ implies that the mapping $\varphi : X \to A$, $\varphi(x) = m_x$, where m_x is the greatest element of A_x is a solution of $\mathcal{I}$.

3 Dichotomies

3.1 The CSP dichotomy

The first attempt for a broad classification of problems of the form CSP(Γ) was made in [48]. When translated into the language of polymorphisms, the main result is

▶ **Theorem 20** ([48]). *Let Γ be a constraint language over $\{0, 1\}$. Then* CSP(Γ) *is solvable in polynomial time if and only if Γ has one of the following polymorphisms: a constant operation, the affine operation $x - y + z \pmod 2$, one of the semilattice operations $x \vee y$ or $x \wedge y$, or the* majority *operation $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$. Otherwise it is NP-complete.*

The hardness part of the theorem follows by an argument similar to that in Example 15. The tractability part follows from Proposition 14 and Examples 18 and 19. The case of the majority operation can be solved as 2-SAT or by a slightly stronger consistency algorithm, see [40].

An important feature of Theorem 20 is that it leaves no room for CSPs of intermediate complexity: every CSP(Γ) is either solvable in polynomial time or is NP-complete. Results of this kind are also known as *complexity dichotomies*, and believed to be true for the majority of "natural" problems, even though problems of intermediate complexity provably exist provided P≠NP, [44].

A dichotomy for CSPs over arbitrary finite sets was conjectured in [31, 32]. This conjecture has been referred to as the CSP Dichotomy Conjecture. The conjecture was refined by providing a specific criterion of polynomial time solvability in [19], and settled in [14, 15, 51].

We now state the Dichotomy Theorem from [14, 15, 51]. Let Γ, Δ be constraint languages on sets A, B respectively and such that Γ 1-pp-interprets Δ. Suppose $F \subseteq A$ and $\pi : F \to B$ are the parameters of the pp-interpretation, and θ the equivalence relation on F that is the π-preimage of the equality relation on B, or the kernel of π. For $a \in F$ let $a/_\theta$ denote the θ-class containing a. Take a polymorphism $f(x_1, \dots, x_k)$ of Γ. By $f_{F,\pi}$ we denote the k-ary operation on B given by $f_{F,\pi}(x_1, \dots, x_k) = \pi(f(\pi^{-1}(x_1), \dots, \pi^{-1}(x_k)))$. Since θ is invariant under f, the operation $f_{F,\pi}(x_1, \dots, x_k)$ is properly defined, that is, its value does not depend on the choices of preimages $\pi^{-1}(x_i)$. It is straightforward to verify that $f_{F,\pi}$ is a polymorphism of Δ. Note that in order to define $f_{F,\pi}$ we do not need to specify constraint language Δ, we only need F and π.

Also, by Proposition 14 it suffices to consider idempotent constraint languages.

▶ **Theorem 21** ([14, 15, 51]). *Let Γ be an idempotent constraint language on a finite set A. Then* $\mathrm{CSP}(\Gamma)$ *is NP-complete if and only if there is a pp-interpretation with parameters F, π such that $f_{F,\pi}$ is a projection for any $f \in \mathsf{Pol}(\Gamma)$.*

The criterion in Theorem 21 can be elevated to a higher level of abstraction that involves considering sets of polymorphisms as a new algebraic structure and using the properties of mappings between these structures, *clone homomorphisms*. We touch upon such approach in Section 4.1.

3.2 Counting CSPs and polymorphisms

In this section we give some examples that hint at how a dichotomy result can be obtained for the counting version of the CSP of the form $\#\mathrm{CSP}(\Gamma)$.

We first remark that the algebraic approach works for counting CSPs as well as for decision CSPs. Recall that Γ^* for a constraint language Γ denotes Γ with added constant relations.

▶ **Theorem 22** ([17]). *Let Γ be a constraint language on a finite set A.*

1. $\#\mathrm{CSP}(\Gamma)$ *and* $\#\mathrm{CSP}(\Gamma^*)$ *are polynomial time interreducible.*
2. *For any constraint language Δ on A such that* $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$ *the problem* $\#\mathrm{CSP}(\Delta)$ *is polynomial time reducible to* $\mathrm{CSP}(\Gamma)$.
3. *For any constraint language Δ, if Γ pp-interprets Δ then* $\#\mathrm{CSP}(\Delta)$ *is polynomial time reducible to* $\#\mathrm{CSP}(\Gamma)$.

Next we use the algebraic approach to prove a dichotomy theorem for $\#H$-Coloring, the counting version of H-Coloring, first proved in [29].

▶ **Example 23.** It was proved in [29] that $\#H$-Coloring is #P-complete, unless every connected component of H is either an isolated vertex, or a complete graph with all loops present, or a complete bipartite graph. That $\#H$-Coloring can be solved in polynomial time for graphs of this kind is straightforward. We show how to prove the hardness part through the algebraic approach in 3 easy steps.

First, using some algebraic machinery ([36] or Theorem 9.13 from [37]) it can be shown that if a constraint language Γ does not have a Maltsev polymorphism, Γ pp-interprets a binary reflexive but not symmetric relation R. Second, using the standard interpolation technique (see e.g. [50]) [17] proves that $\#\mathrm{CSP}(R)$ is #P-complete. This shows that $\#\mathrm{CSP}(\Gamma)$, not only $\#H$-Coloring, is #P-complete for any Γ without a Maltsev polymorphism.

Finally, if a connected graph H is not a single vertex, a complete graph with all loops present, or a complete bipartite graph, it contains the N-graph shown in Fig. 3 as an induced subgraph (some of a, b, c, d may be equal). It remains to observe that no Maltsev operation can be a polymorphism of such graph. Indeed, if f is a Maltsev operation on H then

$$f\left(\binom{a}{c}, \binom{a}{d}, \binom{b}{d}\right) = \binom{b}{c} \notin E(H).$$

It turns out that a generalized version of avoiding N-graphs is key in the case of general counting CSPs, $\#\mathrm{CSP}(\Gamma)$. A constraint language is said to be *singular* if for any pp-interpretable equivalence relations θ, η a certain condition holds on the number of elements in θ- and η-classes. For more details, see [18, 13, 30].

▶ **Theorem 24** ([13, 30]). *For a constraint language Γ on a finite set,* $\#\mathrm{CSP}(\Gamma)$ *is solvable in polynomial time if and only if Γ has a Maltsev polymorphism and is singular. Otherwise it is #P-complete.*

Theorem 24 has been generalized to the weighted version of the CSP in [25].

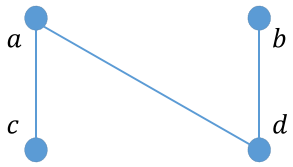

Figure 3 The N-graph.

4 Beyond CSP

In this section we consider several examples in which some sort of algebraic approach is used, although it requires significant modifications. In the first two examples, the Promise CSP and the optimization version, the Valued CSP, while preserving the main motifs of the algebraic approach, that is, rich families of polymorphisms give rise to easy problems, and poor ones give rise to hard problems, new types of "polymorphism-like" objects need to be introduced. In the third case, Graph Isomorphism, the algebraic approach does not (probably) provide a general technique, but is useful in some special cases.

4.1 Promise CSP

Unlike the regular CSP that is often parametrized by a single relational structure, $\mathrm{CSP}(\mathcal{B})$, a Promise CSP or PCSP for short is parametrized by a pair of similar relational structures, $\mathrm{PCSP}(\mathcal{A},\mathcal{B})$, such that $\mathcal{A}\to\mathcal{B}$. An instance of $\mathrm{PCSP}(\mathcal{A},\mathcal{B})$ is a relational structure $\mathcal{C}$ similar to $\mathcal{A},\mathcal{B}$, and the question is whether $\mathcal{C}\to\mathcal{A}$ or $\mathcal{C}\not\to\mathcal{B}$; no specific answer is required if $\mathcal{C}\not\to\mathcal{A}$ but $\mathcal{C}\to\mathcal{B}$. It is also often formulated in a slightly different way: the input is a structure $\mathcal{C}$ such that $\mathcal{C}\to\mathcal{A}$, and the goal is to find a homomorphism from $\mathcal{C}$ to $\mathcal{B}$, which explains the "promise" in the name of the problem, [10, 23].

▶ **Example 25.** The Approximate Graph Coloring problem is parametrized by two numbers k,c, $k\le c$, and the goal is to find a c-coloring of a given k-colorable graph. As is easily seen, this problem is equivalent to $\mathrm{PCSP}(K_k,K_c)$.

Polymorphisms of a single relational structure $\mathcal{A}$ were defined as homomorphisms from $\mathcal{A}^k$ to $\mathcal{A}$. In the case of PCSPs such mappings do not make much sense. Instead, a polymorphism of a pair $\mathcal{A},\mathcal{B}$ is a homomorphism from $\mathcal{A}^k$ to $\mathcal{B}$. Let $\mathsf{Pol}(\mathcal{A},\mathcal{B})$ denote the set of all such homomorphisms for all natural k. Note that as there is a homomorphism $\varphi:\mathcal{A}\to\mathcal{B}$, there also exist some homomorphisms in $\mathsf{Pol}(\mathcal{A},\mathcal{B})$: Fix $i\in[k]$ the mapping $\varphi_i:\mathcal{A}^k\to\mathcal{B}$ given by $\varphi_i(a_1,\dots,a_k)=\varphi(a_i)$ is a homomorphism. Homomorphisms φ_i are analogous to projections in the realm of polymorphisms of a single structure.

We describe the algebraic approach to the PCSP at a higher level of abstraction than that for the CSP. An operation $f:A^n\to B$ is a *minor* of operation $g:A^m\to B$ if there is a mapping $\pi:[m]\to[n]$ such that $f(x_1,\dots,x_n)=g(x_{\pi(1)},\dots,x_{\pi(m)})$. In other words, f is obtained from g by permuting and identifying variables. For example, $f(x_1,x_2,x_3)=g(x_2,x_3,x_2,x_1,x_1,x_3)$ is a minor of $g(x_1,x_2,x_3,x_4,x_5,x_6)$. Let $\mathcal{A}_1,\mathcal{B}_1$, and $\mathcal{A}_2,\mathcal{B}_2$, $\mathcal{A}_1\to\mathcal{B}_1$, $\mathcal{A}_2\to\mathcal{B}_2$, be two pairs of relational structures such that $\mathcal{A}_1,\mathcal{B}_1$ are similar and so are $\mathcal{A}_2,\mathcal{B}_2$. A mapping $\Psi:\mathsf{Pol}(\mathcal{A}_1,\mathcal{B}_1)\to\mathsf{Pol}(\mathcal{A}_2,\mathcal{B}_2)$ (i.e., polymorphisms are mapped to polymorphisms) is called a *minion homomorphism* if the following two conditions hold.

1. Ψ preserves arity, that is, for an operation $f : A_1^k \to B_1$, it holds $\Psi f : A_2^k \to B_2$.
2. Ψ preserves minors, that is, for any m-ary $g \in \mathsf{Pol}(\mathcal{A}_1, \mathcal{B}_1)$ and any $\pi : [m] \to [n]$, if $f(x_1, \dots, x_n) = g(x_{\pi(1)}, \dots, x_{\pi(m)})$ is a minor of g then $\Psi f(x_1, \dots, x_n) = \Psi g(x_{\pi(1)}, \dots, x_{\pi(m)})$.

The core of the algebraic approach to the PCSP is the following

▶ **Theorem 26** ([23]). *Let* $\mathrm{PCSP}(\mathcal{A}_1, \mathcal{B}_1), \mathrm{PCSP}(\mathcal{A}_2, \mathcal{B}_2)$ *be two Promise CSPs such that there is a minion homomorphism from* $\mathsf{Pol}(\mathcal{A}_1, \mathcal{B}_1)$ *to* $\mathsf{Pol}(\mathcal{A}_2, \mathcal{B}_2)$*. Then* $\mathrm{PCSP}(\mathcal{A}_2, \mathcal{B}_2)$ *is polynomial time reducible to* $\mathrm{PCSP}(\mathcal{A}_1, \mathcal{B}_1)$*.*

Note that by setting $\mathcal{A}_1 = \mathcal{B}_1$ and $\mathcal{A}_2 = \mathcal{B}_2$ Theorem 26 specializes to a statement about sets of polymorphisms of a single relational structure. Thus, Theorem 26 is a generalized and more abstract version of the combination of Corollary 13, Proposition 14, and Theorem 16.

▶ **Example 27** ([23]). In this example we use Theorem 26 to prove that $\mathrm{PCSP}(K_3, K_4)$ is NP-complete. In other words we outline a proof that given a 3-colorable graph it is in general NP-hard to find its 4-coloring. In order to do this we present a minion homomorphism from $\mathsf{Pol}(K_3, K_4)$ to $\mathsf{Pol}(\mathcal{A}, \mathcal{A})$, where $\mathcal{A}$ is any 2-element relational structure that only has projections as polymorphisms. Note that $\mathsf{Pol}(\mathcal{A}, \mathcal{A}) = \mathsf{Pol}(\mathcal{A})$, and by Corollary 13 what specific structure we choose is completely irrelevant. For instance, the structure with the base set $\{0, 1\}$ and the only predicate, which is $R_{1\text{-in-}3}$, see Example 6, fits this purpose. Since $\mathrm{CSP}(\mathcal{A}) = \mathrm{PCSP}(\mathcal{A}, \mathcal{A})$ is NP-complete in this case, it implies the result.

The structure of polymorphisms from $\mathsf{Pol}(K_3, K_4)$ is described in [23, 9]. For each (say, k-ary) $f \in \mathsf{Pol}(K_3, K_4)$, there exist $t \in K_4$, $i \in [k]$, and a mapping $\varphi : K_3 \to K_4$ such that $f(a_1, \dots, a_k) \in \{t, \varphi(a_i)\}$ for all $a_1, \dots, a_n \in K_3$. In other words, if the value of f is not t, it depends only on x_i. A mapping $\Psi : \mathsf{Pol}(K_3, K_4) \to \mathsf{Pol}(\mathcal{A}, \mathcal{A})$ is defined as follows: For every k-ary $f \in \mathsf{Pol}(K_3, K_4)$, Ψf is the k-ary projection $p(x_1, \dots, x_k) = x_i$, where i is the parameter associated with f. It is straightforward to verify that Ψ is a minion homomorphism.

4.2 Valued CSP, optimization

The VCSP and optimization problems admit some sort of algebraic approach, however the concept of a polymorphism is substantially different. In this section we briefly outline how the approach works when we want to find the exact optimum of a VCSP [49], and also for approximation algorithms [11]. The functions we consider here are real-valued.

To introduce the notion of polymorphisms needed for VCSPs, we need to take into account all the operations of certain arity on a set, rather than a single operation. Let $\mathcal{O}_A^{(k)}$ denote the set of all k-ary operations on a set A. Let $R : A^m \to \mathbb{R}$ be a function on A. A k-ary *fractional polymorphism* of R is a probability distribution μ on $\mathcal{O}_A^{(k)}$ that for any $\mathbf{a}_1, \dots, \mathbf{a}_k \in A^m$, $\mathbf{a}_i = (a_{i1}, \dots, a_{im})$ satisfies (in the case of minimization problems) the following condition

$$\mathbb{E}_{f \sim \mu}[R(f(\mathbf{a}_1, \dots, \mathbf{a}_k))] \le \mathsf{avg}(R(\mathbf{a}_1), \dots, R(\mathbf{a}_k)),$$

where $f(\mathbf{a}_1, \dots, \mathbf{a}_k) = (f(a_{11}, \dots, a_{k1}), \dots, f(a_{1m}, \dots, a_{km}))$. In the case of maximization problems the inequality should be reversed. Distribution μ is a fractional polymorphism of a valued constraint language Γ if it is a fractional polymorphism of every function from Γ.

▶ **Example 28.** Let $A = \{0, 1\}$, and let μ be the distribution on $\mathcal{O}_A^{(2)}$ that assigns probability $1/2$ to $\vee$ (disjunction) and $\wedge$ (conjunction), which are binary operations on $\{0, 1\}$, and probability 0 to all other operations. For a binary function R the inequality above is transformed into

$$\frac{1}{2}(R(x_1 \vee x_2, y_1 \vee y_2) + R(x_1 \wedge x_2, y_1 \wedge y_2) \le \frac{1}{2}(R(x_1, y_1) + R(x_2, y_2)),$$

which is the well known condition of submodularity.

A binary fractional polymorphism μ (i.e., a distribution on $\mathcal{O}_A^{(2)}$) is said to be *symmetric* if it assigns nonzero probabilities only to operations $f(x,y)$ satisfying $f(x,y) = f(y,x)$.

▶ **Theorem 29** ([49]). *For a valued constraint language* Γ *with real values the problem* VCSP(Γ) *is solvable in polynomial time if and only if* Γ *has a binary symmetric fractional polymorphism.*

Observe that the condition of submodularity is an example of a symmetric fractional polymorphism.

Many VCSPs that cannot be solved exactly can be approximated within a constant factor. Assuming the Unique Games Conjecture (UGC) [11] determines the best approximation factor, so-called *approximation threshold*, for arbitrary VCSPs. The main tool in this result is a variation of fractional polymorphisms. For a set A we again consider probability distributions over $\mathcal{O}_A^{(k)}$. Let $\alpha \in [0,1]$. Distribution μ is said to be an *α-approximation polymorphism* of $R : A^m \to \mathbb{R}$ if for any $\mathbf{a}_1, \dots, \mathbf{a}_k \in A^m$ it satisfies the following condition

$$\alpha \cdot \mathbb{E}_{f\sim\mu}[R(f(\mathbf{a}_1, \dots, \mathbf{a}_k))] \geq \mathsf{avg}(R(\mathbf{a}_1), \dots, R(\mathbf{a}_k)).$$

For a definition of pseudorandom approximation polymorphisms the reader is referred to [11].

▶ **Theorem 30** ([11]). *Let* Γ *be a valued constraint language, and let* α_Γ *be the greatest constant such that there is a pseudorandom* α_Γ*-approximation polymorphism of* Γ*. Then (assuming the UGC)* α_Γ *is the approximation threshold for* VCSP(Γ).

4.3 Graph Isomorphism: bounded color classes

A somewhat surprising application of the algebraic approach in the Graph Isomorphism problem was observed in [5]. In the Graph Isomorphism problem [35] the question is to decide whether two given graphs are isomorphic. The equivalent formulation we use here asks to find a generating set for the *automorphism group* $\mathsf{Aut}(G)$ of G. Often considering the structure of G, e.g. the degrees of its vertices, it is possible to identify some restrictions on the *orbits* of $\mathsf{Aut}(G)$, that is, which vertices can be mapped to each other by automorphisms. For instance, a vertex can only be mapped to a vertex of the same degree. Usually such restrictions are a result of some sort of *color refinement* process, but this is not important here. Suppose that we can identify a partition of $V(G)$ into classes $V_1, \dots, V_k$ such that any automorphism φ maps V_i to V_i. Assume also that the sizes of the V_is are bounded by $\ell \in \mathbb{N}$. Let us further assume for simplicity that $|V_i| = \ell$ for all $i \in [k]$. Then any $\varphi \in \mathsf{Aut}(G)$ is a union of permutations φ_i of V_i, $i \in [k]$, and the question is which combinations of such local permutations give rise to an automorphism of G.

We describe how this problem can be reduced to CSP(Γ) where Γ is a constraint language with a Maltsev polymorphism, see Example 18. Let $V_i = \{v_i^1, \dots, v_i^\ell\}$ be some enumeration of V_i, and let the symmetric group S_ℓ of permutations of $[\ell]$ acts on V_i as follows: for $\pi \in S_\ell$, $\pi(v_i^j) = v_i^{\pi(j)}$. The domain for our CSP is S_ℓ, and the variables are $V_1, \dots, V_k$. For every pair V_i, V_j we introduce a constraint $R_{ij}(V_i, V_j)$ as follows. Suppose $v_i^s v_j^r$ is an edge of G. Then if φ_i, φ_j are a part of an automorphism of G, $\varphi_i(v_i^s)\varphi_j(v_j^r) \in E(G)$. Permutations φ_i, φ_j correspond to some $\pi_i, \pi_j \in S_\ell$. Thus, we define R_{ij} to be $\{(\pi_i, \pi_j) \mid$ for any $v_i^s v_j^r \in E(G)$, $v_i^{\pi_i(s)} v_j^{\pi_j(r)} \in E(G)\}$. Let $\Gamma = \{R_{ij} \mid i, j \in [k]\}$. Observe that R_{ij} is invariant under composition, i.e. if $(\pi_i, \pi_j), (\tau_i, \tau_j) \in R_{ij}$ then $(\pi_i \circ \tau_i, \pi_j \circ \tau_j) \in R_{ij}$. In particular, it is invariant under the operation $x \circ y^{-1} \circ z$ of S_ℓ, which is a Maltsev operation on S_ℓ.

The algorithm from [16] finds a concise representation of the set of solutions of the CSP above, which then can be used to construct a generating set for $\mathsf{Aut}(G)$. The running time of this algorithm is polynomial in k and $|S_\ell| = \ell!$.

5 Conclusion

As we have seen, algebraic methods in CSPs have found their applications in numerous areas of computer science. In some of those areas such as decision, counting, and valued CSPs comprehensive and broad results have been obtained. In some other areas active research is ongoing. In particular the algebraic structure of the Promise CSP and CSPs over infinite domains have received much attention recently, although major questions remain open. There have been attempts to introduce the algebraic approach to the study of the holant problem [2] and certain proof systems suitable for approximation (such as Sum-of-Squares) [22], but these studies are in their infancy. Finally, from our perspective one of the most interesting areas where the algebraic approach is yet to be developed is approximate counting. Apart from the CSP itself it also has strong connections to other fields such as statistical physics, and it would be very interesting to see what kind of algebraic structure is possible here.

References

1 Albert Atserias and Joanna Ochremiak. Proof complexity meets algebra. *ACM Trans. Comput. Log.*, 20(1):1:1–1:46, 2019.

2 Miriam Backens and Leslie Ann Goldberg. Holant clones and the approximability of conservative holant problems. *ACM Trans. Algorithms*, 16(2):23:1–23:55, 2020.

3 Libor Barto and Marcin Kozik. Robustly solvable constraint satisfaction problems. *SIAM J. Comput.*, 45(4):1646–1669, 2016.

4 Libor Barto, Andrei A. Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei A. Krokhin and Stanislav Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

5 Christoph Berkholz and Martin Grohe. Limitations of algebraic approaches to graph isomorphism testing. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 155–166. Springer, 2015.

6 Manuel Bodirsky, Martin Hils, and Barnaby Martin. On the scope of the universal-algebraic approach to constraint satisfaction. *Log. Methods Comput. Sci.*, 8(3), 2009.

7 Manuel Bodirsky and Marcello Mamino. Constraint satisfaction problems over numeric domains. In Andrei A. Krokhin and Stanislav Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 79–111. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

8 V.G. Bodnarchuk, L.A. Kaluzhnin, V.N. Kotov, and B.A. Romov. Galois theory for Post algebras. I. *Kibernetika*, 3:1–10, 1969.

9 Joshua Brakensiek and Venkatesan Guruswami. New hardness results for graph and hypergraph colorings. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 14:1–14:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

10 Joshua Brakensiek and Venkatesan Guruswami. Promise constraint satisfaction: Structure theory and a symmetric Boolean dichotomy. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1782–1801. SIAM, 2018.

11 Jonah Brown-Cohen and Prasad Raghavendra. Combinatorial optimization algorithms via polymorphisms. *CoRR*, abs/1501.01598, 2015. `arXiv:1501.01598`.

12 Andrei Bulatov, Marcin Kozik, Peter Mayr, and Markus Steindl. The subpower membership problem for semigroups. *Int. J. Algebra Comput.*, 26(7):1435–1451, 2016.

13 Andrei A. Bulatov. The complexity of the Counting Constraint Satisfaction Problem. *J. ACM*, 60(5):34:1–34:41, 2013.

14 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330, 2017.

15 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs simplified. *CoRR*, abs/2007.09099, 2020. `arXiv:2007.09099`.

16 Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

17 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the Counting Constraint Satisfaction Problem. *Inf. Comput.*, 205(5):651–678, 2007.

18 Andrei A. Bulatov and Martin Grohe. The complexity of partition functions. *Theor. Comput. Sci.*, 348(2-3):148–186, 2005.

19 Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.

20 Andrei A. Bulatov and Dániel Marx. Constraint Satisfaction Problems and global cardinality constraints. *Commun. ACM*, 53(9):99–106, 2010.

21 Andrei A. Bulatov and Dániel Marx. Constraint Satisfaction parameterized by solution size. *SIAM J. Comput.*, 43(2):573–616, 2014.

22 Andrei A. Bulatov and Akbar Rafiey. On the complexity of CSP-based ideal membership problems. *CoRR*, abs/2011.03700, 2020. `arXiv:2011.03700`.

23 Jakub Bulín, Andrei A. Krokhin, and Jakub Oprsal. Algebraic approach to promise constraint satisfaction. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 602–613. ACM, 2019.

24 Jin-Yi Cai and Xi Chen. *Complexity dichotomies for counting problems. Volume 1: Boolean domain.* Cambridge University Press, 2017.

25 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. *J. ACM*, 64(3):19:1–19:39, 2017.

26 Hubie Chen, Matt Valeriote, and Yuichi Yoshida. Constant-query testability of assignments to Constraint Satisfaction Problems. *SIAM J. Comput.*, 48(3):1022–1045, 2019.

27 Hubie Chen and Matthew Valeriote. Learnability of solutions to conjunctive queries. *J. Mach. Learn. Res.*, 20:67:1–67:28, 2019.

28 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 2001.

29 Martin E. Dyer and Catherine S. Greenhill. The complexity of counting graph homomorphisms. *Random Struct. Algorithms*, 17(3-4):260–289, 2000.

30 Martin E. Dyer and David Richerby. An effective dichotomy for the Counting Constraint Satisfaction Problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013.

31 Tomas Feder and Moshe Y. Vardi. Monotone Monadic SNP and constraint satisfaction. In *Proceedings of 25th ACM Symposium on the Theory of Computing (STOC)*, pages 612–622, 1993.

32 Tomas Feder and Moshe Y. 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.

33 Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, and Ivan Mihajlin. Lower bounds for the graph homomorphism problem. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 481–493. Springer, 2015.

34 D. Geiger. Closed systems of function and predicates. *Pacific Journal of Mathematics*, pages 95–100, 1968.

35 Martin Grohe and Pascal Schweitzer. The graph isomorphism problem. *Commun. ACM*, 63(11):128–134, 2020.

36 J. Hagemann and A. Mitschke. On n-permutable congruences. *Algebra Universalis*, pages 8–12, 1973.

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

38 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

39 Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

40 Peter G. Jeavons, David A. Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

41 Peter Jonsson, Victor Lagerkvist, and Biman Roy. Fine-grained time complexity of Constraint Satisfaction Problems. *ACM Trans. Comput. Theory*, 13(1):2:1–2:32, 2021.

42 Ondrej Klíma, Pascal Tesson, and Denis Thérien. Dichotomies in the complexity of solving systems of equations over finite semigroups. *Theory Comput. Syst.*, 40(3):263–297, 2007.

43 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. In Alberto O. Mendelzon and Jan Paredaens, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, pages 205–213. ACM Press, 1998.

44 Richard Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

45 Barnaby Martin. Quantified constraints in twenty seventeen. In Andrei A. Krokhin and Stanislav Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 327–346. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

46 Monaldo Mastrolilli. The complexity of the ideal membership problem for constrained problems over the Boolean domain. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 456–475. SIAM, 2019.

47 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, 2008.

48 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226, 1978.

49 Johan Thapper and Stanislav Zivný. The complexity of finite-valued CSPs. *J. ACM*, 63(4):37:1–37:33, 2016.

50 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

51 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020.