Algebraic Theory of Promise Constraint Satisfaction Problems, First Steps

Libor Barto(✉)

Department of Algebra, Faculty of Mathematics and Physics, Charles University, Sokolovská 83, 18675 Praha 8, Czechia
libor.barto@gmail.com
http://www.karlin.mff.cuni.cz/~barto

Abstract. What makes a computational problem easy (e.g., in P, that is, solvable in polynomial time) or hard (e.g., NP-hard)? This fundamental question now has a satisfactory answer for a quite broad class of computational problems, so called fixed-template constraint satisfaction problems (CSPs) – it has turned out that their complexity is captured by a certain specific form of symmetry. This paper explains an extension of this theory to a much broader class of computational problems, the promise CSPs, which includes relaxed versions of CSPs such as the problem of finding a 137-coloring of a 3-colorable graph.

Keywords: Computational complexity · Promise constraint satisfaction · Polymorphism

1 Introduction

In Computational Complexity we often try to place a given computational problem into some familiar complexity class, such as P, NP-complete, etc. In other words, we try to determine the image of a computational problem under the following mapping Φ.

$$\Phi : \text{computational problems} \to \text{complexity classes}$$
$$\text{problem} \mapsto \text{its complexity class}$$

When we try to achieve this goal for a whole class of computational problems, say $\mathscr{S}$, it is a natural idea to look for some intermediate collection $\mathfrak{I}$ of "invariants" and a decomposition of Φ through $\mathfrak{I}$:

$$\mathscr{S} \xrightarrow{\Psi} \mathfrak{I} \to \text{complexity classes}$$

Members of $\mathfrak{I}$ are then objects that exactly capture the computational complexity of problems in $\mathscr{S}$. The larger $\mathscr{S}$ is and the more objects Ψ glues together, the better such a decomposition is.

Libor Barto has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 771005).

© Springer Nature Switzerland AG 2019
L. A. Gąsieniec et al. (Eds.): FCT 2019, LNCS 11651, pp. 3–17, 2019.
https://doi.org/10.1007/978-3-030-25027-0_1

This idea proved greatly useful for an interesting class of problems, so called fixed-template constraint satisfaction problems (CSPs), and eventually led to a full complexity classification result [17,30]. In a decomposition, suggested in [20] and proved in [24], Ψ assigns to a CSP a certain algebraic object that describes, informally, the high dimensional symmetries of the CSP. This basic insight of the so called *algebraic approach to CSPs* was later twice improved [7,16], giving us a chain

$$\text{CSPs} \xrightarrow{\Psi} \mathfrak{I}_1 \to \mathfrak{I}_2 \to \mathfrak{I}_3 \to \text{complexity classes}.$$

The basics of the algebraic theory can be adapted and applied in various generalizations and variants of the fixed-template CSPs, see surveys in [26]. One particularly exciting direction is a recently proposed significant generalization of CSPs, so called promise CPSs (PCSPs) [3,14]. This framework is substantially richer, both on the algorithmic and the hardness side, and a full complexity classification is wide open even in very restricted subclasses. On the other hand, the algebraic basics can be generalized from CSP to PCSP and, moreover, one of the results in [18] not only gives such a generalization but also provides an additional insight and simplifies the algebraic theory of CSPs.

The aim of this paper is to explain this result (here Theorem 6) and the development in CSPs leading to it (Theorems 1, 2 and 3). The most recent material comes from the conference papers [18] and [4], which will be merged and expanded in [5]. Very little preliminary knowledge is assumed but an interested reader may find an in depth introduction to the fixed-template CSP and its variants in [26].

2 CSP

Fur the purposes of this paper, we define a *finite relational structure* as a tuple $\mathbb{A} = (A; R_1, \dots, R_n)$, where A is a finite set, called the *domain* of $\mathbb{A}$, and each R_i is a relation on A of some arity, that is, $R_i \subseteq A^{\mathrm{ar}(R_i)}$ where $\mathrm{ar}(R_i)$ is a natural number.

A *primitive positive formula (pp-formula)* over $\mathbb{A}$ is a formula that uses only existential quantification, conjunction, relations in $\mathbb{A}$, and the equality relation. We will work only with formulas in a prenex normal form.

Definition 1. *Fix a finite relational structure $\mathbb{A}$. The CSP over $\mathbb{A}$, written* $\mathrm{CSP}(\mathbb{A})$*, is the problem of deciding whether a given pp-sentence over $\mathbb{A}$ is true.*

In this context, $\mathbb{A}$ is called the template *for* $\mathrm{CSP}(\mathbb{A})$*.*

For example, if $\mathbb{A} = (A; R, S)$ and both R and S are binary, then an input to $\mathrm{CSP}(\mathbb{A})$ is, e.g.,

$$(\exists x_1 \exists x_2 \dots \exists x_5)\ R(x_1, x_3) \wedge S(x_5, x_2) \wedge R(x_3, x_3).$$

This sentence is true if there exists a *satisfying assignment*, that is, a mapping $f : \{x_1, \dots, x_5\} \to A$ such that $(f(x_1), f(x_3)) \in R$, $(f(x_5), f(x_2)) \in S$, and

$(f(x_3), f(x_3)) \in R$. Each conjunct thus can be thought of as a constraint limiting f and the goal is to decide whether there is an assignment satisfying each constraint.

Clearly, CSP($\mathbb{A}$) is always in NP.

The CSP over $\mathbb{A}$ can be also defined as a search problem where the goal is to find a satisfying assignment when it exists. It has turned out that the search problem is no harder then the decision problem presented in Definition 1 [16].

2.1 Examples

Typical problems covered by the fixed-template CSP framework are satisfiability problems, (hyper)graph coloring problems, and equation solvability problems. Let us look at several examples. We use here the notation

$$E_k = \{0, 1, \dots, k-1\}.$$

Example 1. Let $3\mathbb{SAT} = (E_2; R_{000}, R_{001}, \dots, R_{111})$, where

$$R_{abc} = E_2^3 \setminus \{(a, b, c)\} \quad \text{for all } a, b, c \in \{0, 1\}.$$

An input to CSP($3\mathbb{SAT}$) is, e.g.,

$$(\exists x_1 \exists x_2 \dots)\ S_{001}(x_1, x_4, x_2) \wedge S_{110}(x_2, x_5, x_5) \wedge S_{000}(x_2, x_1, x_3) \wedge \dots.$$

Observe that this sentence is true if and only if the propositional formula

$$(x_1 \vee x_4 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_5 \vee x_5) \wedge (x_2 \vee x_1 \vee x_3) \wedge \dots$$

is satisfiable. Therefore CSP($3\mathbb{SAT}$) is essentially the same as the 3SAT problem, a well known NP-complete problem.

On the other hand, recall that the 2SAT problem, which is the CSP over $2\mathbb{SAT} = (E_2; R_{00}, R_{01}, R_{10}, R_{11})$, where $R_{ab} = E_2^2 \setminus \{(a, b)\}$, is in P.

Example 2. Let $\mathbb{K}_3 = (E_3; N_3)$, where N_3 is the binary inequality relation, i.e.,

$$N_3 = \{(a, b) \in E_3^2 : a \neq b\}.$$

An input to CSP($\mathbb{K}_3$) is, e.g.,

$$(\exists x_1 \dots \exists x_5)\ N_3(x_1, x_2) \wedge N_3(x_1, x_3) \wedge N_3(x_1, x_4) \wedge N_3(x_2, x_3) \wedge N_3(x_2, x_4).$$

Here an input can be drawn as a graph – its vertices are the variables and vertices x, y are declared adjacent iff the input contains a conjunct $N_3(x, y)$ or $N_3(y, x)$. For example, the graph associated to the input above is the five vertex graph obtained by merging two triangles along an edge. Clearly, an input sentence is true if and only if the vertices of the associated graph can be colored by colors 0, 1, and 2 so that adjacent vertices receive different colors. Therefore CSP($\mathbb{K}_3$) is essentially the same as the 3-coloring problem for graphs, another well known NP-complete problem.

More generally, CSP($\mathbb{K}_k$) $= (E_k, N_k)$, where N_k is the inequality relation on E_k, is NP-complete for $k \geq 3$ and in P for $k = 2$.

Example 3. Let $3\mathbb{NAE}_k = (E_k; 3NAE_k)$, where $3NAE_k$ is the ternary not-all-equal relation, i.e.,

$$3NAE_k = E_k^3 \setminus \{(a,a,a) : a \in E_k\}.$$

Taking the viewpoint of Example 1, the CSP over $3\mathbb{NAE}_2$ is the positive not-all-equal 3SAT, where one is given a 3SAT formula without negations and the aim is to decide whether there is an assignment such that, in every clause, not all variables get the same value. This problem is NP-complete [29].

From the graph theoretical viewpoint, CSP($3\mathbb{NAE}_k$) is the problem of deciding whether a given 3-uniform hypergraph[1] admits a coloring by k colors so that no hyperedge is monochromatic.

Example 4. Let $1\mathbb{IN}3 = (E_2; 1IN3)$, where

$$1IN3 = \{(1,0,0),(0,1,0),(1,0,0)\}.$$

The CSP over $1\mathbb{IN}3$ is the positive one-in-three SAT problem or, in other words, the problem of deciding whether a given 3-uniform hypergraph admits a coloring by colors 0 and 1 so that exactly one vertex in each hyperedge receives the color 1. This problem is, again, NP-complete [29].

Example 5. Let $3\mathbb{LIN}_5 = (E_5; L_{0000}, L_{0001}, \dots, L_{4444})$, where

$$L_{abcd} = \{(x,y,z) \in E_5^3 : ax + by + cz = d \pmod 5\}.$$

An input, such as

$$(\exists x_1 \exists x_2 \dots)\ L_{1234}(x_3, x_4, x_2) \wedge L_{4321}(x_5, x_1, x_3) \wedge \dots$$

can be written as a system of linear equations over the 5-element field $\mathbb{Z}_5$, such as

$$1x_3 + 2x_4 + 3x_2 = 4,\ 4x_5 + 3x_1 + 2x_3 = 1,\ \dots,$$

therefore CSP($3\mathbb{LIN}_5$) is essentially the same problem as deciding whether a system of linear equations over $\mathbb{Z}_5$ (with each equation containing 3 variables) has a solution. This problem is in P.

2.2 1st Step: Polymorphisms

The crucial concept for the algebraic approach to the CSP is a polymorphism, which is a homomorphism from a cartesian power of a structure to the structure:

Definition 2. *Let $\mathbb{A} = (A; R_1, \dots, R_n)$ be a relational structure. A k-ary (total) function $f : A^k \to A$ is a* polymorphism *of $\mathbb{A}$ if it is compatible with every relation R_i, that is, for all tuples $\mathbf{r}_1, \dots, \mathbf{r}_k \in R_i$, the tuple $f(\mathbf{r}_1, \dots, \mathbf{r}_k)$ (where f is applied component-wise) is in R_i.*

By Pol($\mathbb{A}$) *we denote the set of all polymorphisms of $\mathbb{A}$.*

[1] Here we should rather say a hypergraph whose hyperedges have size at most 3 because of conjuncts of the form $3NAE_k(x,x,y)$ or $3NAE_k(x,x,x)$. Let us ignore this minor technical imprecision.

The compatibility condition is often stated as follows: for any $(\mathrm{ar}(R_i) \times k)$-matrix whose column vectors are in R_i, the vector obtained by applying f to its rows is in R_i as well.

Note that the unary polymorphisms of $\mathbb{A}$ are exactly the endomorphisms of $\mathbb{A}$. One often thinks of endomorphisms (or just automorphisms) as symmetries of the structure. In this sense, polymorphisms can be thought of as higher dimensional symmetries.

For any domain A and any $i \leq k$, the k-ary projection to the i-th coordinate, that is, the function $\pi_i^k : A^k \to A$ defined by

$$\pi_i^k(x_1, \ldots, x_n) = x_i,$$

is a polymorphism of every structure with domain A. These are the *trivial* polymorphisms. The following examples show some nontrivial polymorphisms.

Example 6. Consider the template $2\mathbb{SAT}$ from Example 1. It is easy to verify that the ternary majority function $\mathrm{maj} : E_2^3 \to E_2$ given by

$$\mathrm{maj}(x, x, y) = \mathrm{maj}(x, y, x) = \mathrm{maj}(y, x, x) = x \quad \text{for all } x, y \in E_2$$

is a polymorphism of $2\mathbb{SAT}$.

In fact, whenever a relation $R \subseteq E_2^m$ is compatible with maj, it can be pp-defined (that is, defined by a pp-formula) from relations in $2\mathbb{SAT}$ (see e.g. [25]). Now for any template $\mathbb{A} = (E_2; R_1, \ldots, R_n)$ with polymorphism maj, an input of $\mathrm{CSP}(\mathbb{A})$ can be easily rewritten to an equivalent input of $\mathrm{CSP}(2\mathbb{SAT})$ and therefore $\mathrm{CSP}(\mathbb{A})$ is in P.

Example 7. Consider the template $3\mathbb{LIN}_5$ from Example 5. Each relation in this structure is an affine subspace of $\mathbb{Z}_5^3$. Every affine subspace is closed under affine combinations, therefore, for every k and every $t_1, \ldots, t_k \in E_5$ such that $t_1 + \cdots + t_k = 1 \pmod 5$, the k-ary function $f_{t_1,\ldots,t_k} : E_5^k \to E_5$ defined by

$$f_{t_1,\ldots,t_k}(x_1, \ldots, x_k) = t_1 x_1 + \ldots, t_k x_k \pmod 5$$

is a polymorphism of $3\mathbb{LIN}_5$.

Conversely, every subset of A^m closed under affine combinations is an affine subspace of $\mathbb{Z}_5^m$. It follows that if every $f_{t_1,\ldots,t_k}$ is a polymorphism of $\mathbb{A} = (E_5; R_1, \ldots, R_n)$, then inputs to $\mathrm{CSP}(\mathbb{A})$ can be rewritten to systems of linear equations over $\mathbb{Z}_5$ and thus $\mathrm{CSP}(\mathbb{A})$ is in P.

The above examples also illustrate that polymorphisms influence the computational complexity. The first step of the algebraic approach was to realize that this is by no means a coincidence.

Theorem 1 ([24]). *The complexity of* $\mathrm{CSP}(\mathbb{A})$ *depends only on* $\mathrm{Pol}(\mathbb{A})$.

More precisely, if $\mathbb{A}$ *and* $\mathbb{B}$ *are finite relational structures and* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$, *then* $\mathrm{CSP}(\mathbb{B})$ *is (log-space) reducible to* $\mathrm{CSP}(\mathbb{A})$. *In particular, if* $\mathrm{Pol}(\mathbb{A}) = \mathrm{Pol}(\mathbb{B})$, *then* $\mathrm{CSP}(\mathbb{A})$ *and* $\mathrm{CSP}(\mathbb{B})$ *have the same complexity.*

Proof (sketch). If $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$, then relations in $\mathbb{B}$ can be pp-defined from relations in $\mathbb{A}$ by a classical result in Universal Algebra [10,11,21]. This gives a reduction from $\mathrm{CSP}(\mathbb{B})$ to $\mathrm{CSP}(\mathbb{A})$.

Theorem 1 can be used as a tool for proving NP-hardness: when $\mathbb{A}$ has only trivial polymorphism (and has domain of size at least two), any CSP on the same domain can be reduced to $\mathrm{CSP}(\mathbb{A})$ and therefore $\mathrm{CSP}(\mathbb{A})$ is NP-complete. This NP-hardness criterion is not perfect, e.g., $\mathrm{CSP}(3\mathbb{NAE}_2)$ has a nontrivial endomorphism $x \mapsto 1 - x$.

2.3 2nd Step: Strong Maltsev Conditions

Theorem 1 shows that the set of polymorphisms determines the complexity of a CSP. What information do we really need to know about the polymorphisms to determine the complexity? It has turned out that it is sufficient to know which functional equations they solve. In the following definition we use a standard universal algebraic term for a functional equation, a strong Maltsev condition.

Definition 3. *A* strong Maltsev condition *over a set of function symbols Σ is a finite set of equations of the form $t = s$, where t and s are terms built from variables and symbols in Σ.*

Let $\mathcal{M}$ be a set of functions on a common domain. A strong Maltsev condition S is satisfied *in $\mathcal{M}$ if the function symbols of Σ can be interpreted in $\mathcal{M}$ so that each equation in S is satisfied for every choice of variables.*

Example 8. A strong Maltsev condition over $\Sigma = \{f, g, h\}$ (where f and g are binary symbols and h is ternary) is, e.g.,

$$f(g(f(x,y),y),z) = g(x,h(y,y,z))$$
$$f(x,y) = g(g(x,y),x).$$

This condition is satisfied in the set of all projections (on any domain) since, by interpreting f and g as π^2_1 and h as π^3_1, both equations are satisfied for every x, y, z in the domain – they are equal to x.

The strong Maltsev condition in the above example is not interesting for us since it is satisfied in every $\mathrm{Pol}(\mathbb{A})$. Such conditions are called trivial:

Definition 4. *A strong Maltsev condition is called* trivial *if it is satisfied in the set of all projections on a two-element set (equivalently, it is satisfied in* $\mathrm{Pol}(\mathbb{A})$ *for every $\mathbb{A}$).*

Two nontrivial Maltsev condition are shown in the following example.

Example 9. The strong Maltsev condition (over a single ternary symbol m)

$$m(x,x,y) = x$$
$$m(x,y,x) = x$$
$$m(y,x,x) = x$$

is nontrivial since each of the possible interpretations π_1^3, π_2^3, π_3^3 of m falsifies one of the equations. This condition is satisfied in $\mathrm{Pol}(2\mathbb{SAT})$ by interpreting m as the majority function, see Example 6.

The strong Maltsev condition

$$p(x,x,y) = y$$
$$p(y,x,x) = y$$

is also nontrivial. It is satisfied in $\mathrm{Pol}(3\mathbb{LIN}_5)$ by interpreting p as $x + 4y + z$ (mod 5).

In fact, if $\mathrm{Pol}(\mathbb{A})$ satisfies one of the strong Maltsev conditions in this example, then $\mathrm{CSP}(\mathbb{A})$ is in P (see e.g. [6]).

The following theorem is (a restatement of) the second crucial step of the algebraic approach.

Theorem 2 ([16], see also [9]). *The complexity of* $\mathrm{CSP}(\mathbb{A})$ *depends only on strong Maltsev conditions satisfied by* $\mathrm{Pol}(\mathbb{A})$.

More precisely, if $\mathbb{A}$ *and* $\mathbb{B}$ *are finite relational structures and each strong Maltsev condition satisfied in* $\mathrm{Pol}(\mathbb{A})$ *is satisfied in* $\mathrm{Pol}(\mathbb{B})$*, then* $\mathrm{CSP}(\mathbb{B})$ *is (log-space) reducible to* $\mathrm{CSP}(\mathbb{A})$*. In particular, if* $\mathrm{Pol}(\mathbb{A})$ *and* $\mathrm{Pol}(\mathbb{B})$ *satisfy the same strong Maltsev conditions, then* $\mathrm{CSP}(\mathbb{A})$ *and* $\mathrm{CSP}(\mathbb{B})$ *have the same complexity.*

Proof (sketch). The proof can be done in a similar way as for Theorem 1. Instead of pp-definitions one uses more general constructions called pp-interpretations and, on the algebraic side, the Birkhoff HSP theorem [8].

Theorem 2 gives us an improved tool for proving NP-hardness: if $\mathrm{Pol}(\mathbb{A})$ satisfies only trivial strong Maltsev conditions, then $\mathrm{CSP}(\mathbb{A})$ is NP-hard. This criterion is better, e.g., it can be applied to $\mathrm{CSP}(3\mathbb{NAE}_2)$, but still not perfect, e.g., it cannot be applied to the CSP over the disjoint union of two copies of $\mathbb{K}_3$.

2.4 3rd Step: Minor Conditions

Strong Maltsev conditions that appear naturally in the CSP theory or in Universal Algebra are often of an especially simple form, they involve no nesting of function symbols. The third step in the basics of the algebraic theory was to realize that this is also not a coincidence.

Definition 5. *A strong Maltsev condition is called a* minor condition *if each side of every equation contains exactly one occurrence of a function symbol.*

In other words, each equation in a strong Maltsev condition is of the form "symbol(variables) = symbol(variables)".

Example 10. The condition in Example 8 is not a minor condition since, e.g., the left-hand side of the first equation involves three occurrences of function symbols.

The conditions in Example 9 are not minor conditions either since the right-hand sides do not contain any function symbol. However, these conditions have close friends which are minor conditions. For instance, the friend of the second system is the minor condition

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

Note that this system is also satisfied in $\mathrm{Pol}(3\mathbb{LIN}_5)$ by the same interpretation as in Example 9, that is, $x + 4y + z \pmod 5$.

The following theorem is a strengthening of Theorem 2. We give only the informal part of the statement, the precise formulation is analogous to Theorem 2.

Theorem 3 ([7]). *The complexity of* $\mathrm{CSP}(\mathbb{A})$ *(for finite* $\mathbb{A}$*) depends only on minor conditions satisfied by* $\mathrm{Pol}(\mathbb{A})$*.*

Proof (sketch). The proof again follows the same pattern by further generalizing pp-interpretations (to so called pp-constructions) and the Birkhoff HSP theorem.

2.5 Classification

Just like Theorems 1 and 2 give hardness criteria for CSPs, we get an improved sufficient condition for NP-hardness as a corollary of Theorem 3.

Corollary 1. *Let* $\mathbb{A}$ *be a finite relational structure which satisfies only trivial minor conditions. Then* $\mathrm{CSP}(\mathbb{A})$ *is NP-complete.*

Bulatov, Jeavons, and Krokhin have conjectured [16] that satisfying only trivial minor conditions is actually the only reason for hardness[2]. Intensive efforts to prove this conjecture, called the *tractability conjecture* or the *algebraic dichotomy conjecture*, have recently culminated in two independent proofs by Bulatov and Zhuk:

Theorem 4 ([17,30]). *If a finite relational structure* $\mathbb{A}$ *satisfies a nontrivial minor condition, then* $\mathrm{CSP}(\mathbb{A})$ *is in P.*

Thus we now have a complete classification result: every finite structure either satisfies a nontrivial minor condition and then its CSP is in P, or it does not and its CSP is NP-complete. The proofs of Bulatov and Zhuk are very complicated and it should be stressed out that the basic steps presented in this paper form only a tiny (but important) part of the theory.

In fact, the third step did not impact on the resolution of the tractability conjecture for CSP over finite domains at all. However, it turned out to be significant for some generalizations of the CSP, including the generalization that we discuss in the next section, the Promise CSP.

[2] Their conjecture is equivalent but was, of course, originally stated in a different language – the significance of minor conditions in CSPs was identified much later.

3 PCSP

Many fixed-template CSPs, such as finding a 3-coloring of a graph or finding a satisfying assignment to a 3SAT formula, are hard computational problems. There are two ways how to relax the requirement on the assignment in order to get a potentially simpler problem. The first one is to require a specified fraction of the constraints to be satisfied. For example, given a satisfiable 3SAT input, is it easier to find an assignment satisfying at least 90% of clauses? A celebrated result of Håstad [22], which strengthens the famous PCP Theorem [1,2], proves that the answer is negative – it is still an NP-complete problem. (Actually, any fraction greater than 7/8 gives rise to an NP-complete problem while the fraction 7/8 is achievable in polynomial time.)

The second type of relaxation, the one we consider in this paper, is to require that a specified weaker version of every constraint is satisfied. For example, we want to find a 100-coloring of a 3-colorable graph, or we want to find a valid $\mathrm{CSP}(3\mathbb{NAE}_2)$ assignment to a true input of $\mathrm{CSP}(1\mathbb{IN}3)$. This idea is formalized in the following definition.

Definition 6. *Let* $\mathbb{A} = (A; R_1^{\mathbb{A}}, R_2^{\mathbb{A}}, \dots, R_n^{\mathbb{A}})$ *and* $\mathbb{B} = (B; R_1^{\mathbb{B}}, R_2^{\mathbb{B}}, \dots, R_n^{\mathbb{B}})$ *be two similar finite relational structures (that is,* $R^{\mathbb{A}}$ *and* $R^{\mathbb{B}}$ *have the same arity for each* i*), and assume that there exists a homomorphism* $\mathbb{A} \to \mathbb{B}$*. Such a pair* $(\mathbb{A}, \mathbb{B})$ *is refered to as a* PCSP template*.*

The PCSP over $(\mathbb{A}, \mathbb{B})$*, denoted* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$*, is the problem to distinguish, given a pp-sentence* ϕ *over the relational symbols* $R_1, \dots, R_n$*, between the cases that* ϕ *is true in* $\mathbb{A}$ *(answer "Yes") and* ϕ *is not true in* $\mathbb{B}$ *(answer "No").*

For example, consider $\mathbb{A} = (A; R^{\mathbb{A}}, S^{\mathbb{A}})$ and $\mathbb{B} = (B; R^{\mathbb{B}}, S^{\mathbb{B}})$, where all the relations are binary. An input to $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is, e.g.,

$$(\exists x_1 \exists x_2 \dots \exists x_5)\ R(x_1, x_3) \wedge S(x_5, x_2) \wedge R(x_3, x_3).$$

The algorithm should answer "Yes" if the sentence is true in $\mathbb{A}$, i.e., the following sentence is true

$$(\exists x_1 \exists x_2 \dots \exists x_5)\ R^{\mathbb{A}}(x_1, x_3) \wedge S^{\mathbb{A}}(x_5, x_2) \wedge R^{\mathbb{A}}(x_3, x_3),$$

and the algorithm should answer "No" if the sentence is not true in $\mathbb{B}$. In case that neither of the cases takes place, we do not have any requirements on the algorithm. Alternatively, we can say that the algorithm is promised that the input satisfies either "Yes" or "No" and it is required to decide which of these two cases takes place.

Note that the assumption that $\mathbb{A} \to \mathbb{B}$ is necessary for the problem to make sense, otherwise, the "Yes" and "No" cases would not be disjoint. Also observe that $\mathrm{CSP}(\mathbb{A})$ is the same problem as $\mathrm{PCSP}(\mathbb{A}, \mathbb{A})$.

The search version of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is perhaps a bit more natural problem: the goal is to find a $\mathbb{B}$-satisfying assignment given an $\mathbb{A}$-satisfiable input. Unlike in the CSP, it is not known whether the search version can be harder than the decision version presented in Definition 6.

3.1 Examples

The examples below show that PCSPs are richer than CSP, both on the algorithmic and the hardness side.

Example 11. Recall the structure $\mathbb{K}_k$ from Example 2 consisting of the inequality relation on a k-element set. For $k \leq l$, the PCSP over $(\mathbb{K}_k, \mathbb{K}_l)$ is the problem to distinguish between k-colorable graphs and graphs that are not even l-colorable (or, in the search version, the problem to find an l-coloring of a k-colorable graph).

Unlike for the case $k = l$, the complexity of this problem for $3 \leq k < l$ is a notorious open question. It is conjectured that $\mathrm{PCSP}(\mathbb{K}_k, \mathbb{K}_l)$ is NP-hard for every $k < l$, but this conjecture was confirmed only in special cases: for $l \leq 2k-2$ [12] (e.g., 4-coloring a 3-colorable graph) and for a large enough k and $l \leq 2^{\Omega(k^{1/3})}$ [23]. The algebraic development discussed in the next subsection helped in improving the former result to $l \leq 2k-1$ [18] (e.g., 5-coloring a 3-colorable graph).

Example 12. Recall the structure $3\mathbb{NAE}_k$ from Example 3 consisting of the ternary not-all-equal relation on a k-element set. For $k \leq l$, the PCSP over $(3\mathbb{NAE}_k, 3\mathbb{NAE}_l)$ is essentially the problem to distinguish between k-colorable 3-uniform hypergraphs and 3-uniform hypergraphs that are not even l-colorable.

This problem is NP-hard for every $2 \leq k \leq l$ [19], the proof uses strong tools, the PCP theorem and Lovász's theorem on the chromatic number of Kneser's graphs [27].

Example 13. Recall from Example 4 that $1\mathbb{IN}3$ denotes the structure on the domain E_2 with the ternary "one-in-three" relation $1IN3$. The PCSP over $(1\mathbb{IN}3, 3\mathbb{NAE}_2)$ is the problem to distinguish between 3-uniform hypergraphs, which admit a coloring by colors 0 and 1 so that exactly one vertex in each hyperedge receives the color 1, and 3-uniform hypergraphs that are not even 2-colorable.

This problem, even its search version, admits elegant polynomial time algorithms [14,15] – one is based on solving linear equations over the integers, another one on linear programming. For this specific template, the algorithm can be further simplified as follows.

We are given a 3-uniform hypergraph, which admits a coloring by colors 0 and 1 so that $(x, y, z) \in 1IN3$ for every hyperedge $\{x, y, z\}$, and we want to find a 2-coloring. We create a system of linear equations *over the rationals* as follows: for each hyperedge $\{x, y, z\}$ we introduce the equation $x + y + z = 1$. By the assumption on the input hypergraph, this system is solvable in $\{0, 1\} \subseteq \mathbb{Q}$ (in fact, $\{0, 1\}$-solutions are the same as valid $1IN3$-assignments). Solving equations in $\{0, 1\}$ is hard, but it is possible to solve the system in $\mathbb{Q} \setminus \{1/3\}$ in polynomial time by a simple adjustment of Gaussian elimination. Now we assign 1 to a vertex x if $x > 1/3$ in our rational solution, and 0 otherwise. It is simple to see that we get a valid 2-coloring.

Interestingly, to solve PCSP($1\mathbb{IN}3, 3\mathbb{NAE}_2$), the presented algorithm uses a CSP over an *infinite* structure, namely $(\mathbb{Q} \setminus \{1/3\}; R)$, where $R = \{(x, y, z) \in (\mathbb{Q} \setminus \{1/3\})^3 : x + y + z = 1\}$. In fact, the infinity is necessary for this PCSP, see [4] for a formal statement and a proof.

3.2 4th Step: Minor Conditions!

After the introduction of the PCSP framework, it has quickly turned out that both the notion of a polymorphism and Theorem 1 have straightforward generalizations.

Definition 7. *Let* $\mathbb{A} = (A; R_1^{\mathbb{A}}, \dots)$ *and* $\mathbb{B} = (B; R_1^{\mathbb{B}}, \dots)$ *be two similar relational structures. A* k*-ary (total) function* $f : A^k \to B$ *is a* polymorphism *of* $(\mathbb{A}, \mathbb{B})$ *if it is compatible with every pair* $(R_i^{\mathbb{A}}, R_i^{\mathbb{B}})$*, that is, for all tuples* $\mathbf{r}_1, \dots, \mathbf{r}_k \in R_i^{\mathbb{A}}$*, the tuple* $f(\mathbf{r}_1, \dots, \mathbf{r}_k)$ *is in* $R_i^{\mathbb{B}}$*.*

By $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$ *we denote the set of all polymorphisms of* $(\mathbb{A}, \mathbb{B})$*.*

Example 14. For every k which is not disible by 3, the k-ary "1/3-threshold" function $f : E_2^k \to E_2$ defined by

$$f(x_1, \dots, x_k) = \begin{cases} 1 \text{ if} \sum x_i/k > 1/3 \\ 0 \text{ else} \end{cases}$$

is a polymorphism of the PCSP template $(1\mathbb{IN}3, 3\mathbb{NAE}_2)$ from Example 13. Any PCSP whose template (over the domains E_2 and E_2) admits all these polymorphisms is in P [14,15].

Theorem 5 ([13]). *The complexity of* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *depends only on* $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$*.*

Proof (sketch). Proof is similar to Theorem 1 using [28] instead of [10,11,21].

Note that, in general, composition of polymorphisms is not even well-defined. Therefore the second step, considering strong Maltsev conditions satisfied by polymorphisms, does not make sense for PCSPs. However, minor conditions make perfect sense and they do capture the complexity of PCSPs, as proved in [18]. Furthermore, the paper [18] also provides an alternative proof by directly relating a PCSP to a computational problem concerning minor conditions!

Theorem 6 ([18]). *Let* $(\mathbb{A}, \mathbb{B})$ *be a PCSP template and* $\mathcal{M} = \mathrm{Pol}(\mathbb{A}, \mathbb{B})$*. The following computational problems are equivalent for every sufficiently large* N*.*

- $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.
- *Distinguish, given a minor condition* $\mathbf{C}$ *whose function symbols have arity at most* N*, between the cases that* $\mathbf{C}$ *is trivial and* $\mathbf{C}$ *is not satisfied in* $\mathcal{M}$*.*

Proof (sketch). The reduction from $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ to the second problem works as follows. Given an input to the PCSP we introduce one $|A|$-ary function symbol g_a for each variable a and one $|R^{\mathbb{A}}|$-ary function symbol f_C for each conjunct

$R(\dots)$. The way to build a minor condition is quite natural, for example, the input

$$(\exists a \exists b \exists c \exists d)\ R(c,a,b) \wedge R(a,d,c) \wedge \dots$$

to $\mathrm{PCSP}(1\mathbb{IN}3, 3\mathbb{NAE}_2)$ is transformed to the minor condition

$$\begin{aligned} f_1(x_1,x_0,x_0) &= g_c(x_0,x_1) \\ f_1(x_0,x_1,x_0) &= g_a(x_0,x_1) \\ f_1(x_0,x_0,x_1) &= g_b(x_0,x_1) \end{aligned}$$

$$\begin{aligned} f_2(x_1,x_0,x_0) &= g_a(x_0,x_1) \\ f_2(x_0,x_1,x_0) &= g_d(x_0,x_1) \\ f_2(x_0,x_0,x_1) &= g_c(x_0,x_1) \\ &\dots \end{aligned}$$

It is easy to see that a sentence that is true in $\mathbb{A}$ is transformed to a trivial minor condition. On the other hand, if the minor condition is satisfied in $\mathcal{M}$, say by the functions denoted $f'_1, f'_2, g'_a, \dots$, then the mapping $a \mapsto g'_a(0,1)$, $b \mapsto g'_b(0,1)$, ... gives a $\mathbb{B}$-satisfying assignment of the sentence – this can be deduced from the fact that f's and g' are polymorphisms.

The reduction in the other direction is based on the idea that the question "Is this minor condition satisfied by polymorphisms of $\mathbb{A}$?" can be interpreted as an input to $\mathrm{CSP}(\mathbb{A})$. The main ingredient is to look at functions as tuples (their tables); then "f is a polymorphism" translates to a conjunction, and equations can be simulated by merging variables.

Theorem 6 implies Theorem 3 (and its generalization to PCSPs) since the computational problem in the second item clearly only depends on minor conditions satisfied in $\mathcal{M}$. The proof sketched above

- is simple and does not (explicitly) use any other results, such as the correspondence between polymorphisms and pp-definitions used in Theorem 1 or the Birkhoff HSP theorem used in Theorem 2, and
- is based on constructions which have already appeared, in some form, in several contexts; in particular, the second item is related to important problems in approximation, versions of the Label Cover problem (see [5,18]).

The theorem and its proof are simple, nevertheless, very useful. For example, the hardness of $\mathrm{PCSP}(\mathbb{K}_k, \mathbb{K}_{2k-1})$ mentioned in Example 11 was proved in [18] by showing that every minor condition satisfied in $\mathrm{Pol}(\mathbb{K}_k, \mathbb{K}_{2k-1})$ is satisfied in $\mathrm{Pol}(3\mathbb{NAE}_2, 3\mathbb{NAE}_l)$ (for some l) and then using the NP-hardness of $\mathrm{PCSP}(3\mathbb{NAE}_2, 3\mathbb{NAE}_l)$ proved in [19] (see Example 12).

4 Conclusion

The PCSP framework is much richer than the CSP framework; on the other hand, the basics of the algebraic theory generalize from CSP to PCSP, as shown

in Theorem 6. Strikingly, the computational problems in Theorem 6 are (promise and restricted versions) of two "similar" problems:

(i) Given a structure $\mathfrak{A}$ and a first-order sentence ϕ over the same signature, decide whether $\mathfrak{A}$ satisfies ϕ.
(ii) Given a structure $\mathfrak{A}$ and a first-order sentence ϕ in a different signature, decide whether symbols in ϕ can be interpreted in $\mathfrak{A}$ so that $\mathfrak{A}$ satisfies ϕ.

Indeed, CSP($\mathbb{A}$) is the problem (i) with $\mathfrak{A}$ a fixed relational structure and ϕ a pp-sentence (and PCSP is a promise version of this problem), whereas a promise version of the problem (ii) restricted to a fixed $\mathfrak{A}$ of purely functional signature and universally quantified conjunctive first-order sentences ϕ is the second item in Theorem 6. Variants of problem (i) appear in many contexts throughout Computer Science. What about problem (ii)?

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM **45**(3), 501–555 (1998). https://doi.org/10.1145/278298.278306
2. Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. J. ACM **45**(1), 70–122 (1998). https://doi.org/10.1145/273865.273901
3. Austrin, P., Guruswami, V., Håstad, J.: (2+ϵ)-sat is NP-hard. SIAM J. Comput. **46**(5), 1554–1573 (2017). https://doi.org/10.1137/15M1006507
4. Barto, L.: Promises make finite (constraint satisfaction) problems infinitary. In: LICS (2019, to appear)
5. Barto, L., Bulín, J., Krokhin, A.A., Opršal, J.: Algebraic approach to promise constraint satisfaction (2019, in preparation)
6. Barto, L., Krokhin, A., Willard, R.: Polymorphisms, and how to use them. In: Krokhin, A., Živný, S. (eds.) The Constraint Satisfaction Problem: Complexity and Approximability, Dagstuhl Follow-Ups, vol. 7, pp. 1–44. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2017). https://doi.org/10.4230/DFU.Vol7.15301.1
7. Barto, L., Opršal, J., Pinsker, M.: The wonderland of reflections. Isr. J. Math. **223**(1), 363–398 (2018). https://doi.org/10.1007/s11856-017-1621-9
8. Birkhoff, G.: On the structure of abstract algebras. Math. Proc. Camb. Philos. Soc. **31**(4), 433–454 (1935). https://doi.org/10.1017/S0305004100013463
9. Bodirsky, M.: Constraint satisfaction problems with infinite templates. In: Creignou, N., Kolaitis, P.G., Vollmer, H. (eds.) Complexity of Constraints. LNCS, vol. 5250, pp. 196–228. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-92800-3_8
10. Bodnarchuk, V.G., Kaluzhnin, L.A., Kotov, V.N., Romov, B.A.: Galois theory for post algebras. I. Cybernetics **5**(3), 243–252 (1969). https://doi.org/10.1007/BF01070906
11. Bodnarchuk, V.G., Kaluzhnin, L.A., Kotov, V.N., Romov, B.A.: Galois theory for Post algebras. II. Cybernetics **5**(5), 531–539 (1969)

12. Brakensiek, J., Guruswami, V.: New hardness results for graph and hypergraph colorings. In: Raz, R. (ed.) 31st Conference on Computational Complexity (CCC 2016). Leibniz International Proceedings in Informatics (LIPIcs), vol. 50, pp. 14:1–14:27. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016). https://doi.org/10.4230/LIPIcs.CCC.2016.14
13. Brakensiek, J., Guruswami, V.: Promise constraint satisfaction: Algebraic structure and a symmetric boolean dichotomy. ECCC Report No. 183 (2016)
14. Brakensiek, J., Guruswami, V.: Promise constraint satisfaction: Structure theory and a symmetric boolean dichotomy. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, pp. 1782–1801. Society for Industrial and Applied Mathematics, Philadelphia (2018). http://dl.acm.org/citation.cfm?id=3174304.3175422
15. Brakensiek, J., Guruswami, V.: An algorithmic blend of LPs and ring equations for promise CSPs. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, 6–9 January 2019, pp. 436–455 (2019). https://doi.org/10.1137/1.9781611975482.28
16. Bulatov, A., Jeavons, P., Krokhin, A.: Classifying the complexity of constraints using finite algebras. SIAM J. Comput. **34**(3), 720–742 (2005). https://doi.org/10.1137/S0097539700376676
17. Bulatov, A.A.: A dichotomy theorem for nonuniform CSPs. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 319–330, October 2017. https://doi.org/10.1109/FOCS.2017.37
18. Bulín, J., Krokhin, A., Opršal, J.: Algebraic approach to promise constraint satisfaction. In: Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC 2019). ACM, New York (2019). https://doi.org/10.1145/3313276.3316300
19. Dinur, I., Regev, O., Smyth, C.: The hardness of 3-uniform hypergraph coloring. Combinatorica **25**(5), 519–535 (2005). https://doi.org/10.1007/s00493-005-0032-4
20. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory. SIAM J. Comput. **28**(1), 57–104 (1998). https://doi.org/10.1137/S0097539794266766
21. Geiger, D.: Closed systems of functions and predicates. Pacific J. Math. **27**, 95–100 (1968). https://doi.org/10.2140/pjm.1968.27.95
22. Håstad, J.: Some optimal inapproximability results. J. ACM **48**(4), 798–859 (2001). https://doi.org/10.1145/502090.502098
23. Huang, S.: Improved hardness of approximating chromatic number. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) APPROX/RANDOM -2013. LNCS, vol. 8096, pp. 233–243. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40328-6_17
24. Jeavons, P.: On the algebraic structure of combinatorial problems. Theor. Comput. Sci. **200**(1–2), 185–204 (1998). https://doi.org/10.1016/S0304-3975(97)00230-2
25. Jeavons, P., Cohen, D., Gyssens, M.: Closure properties of constraints. J. ACM **44**(4), 527–548 (1997). https://doi.org/10.1145/263867.263489
26. Krokhin, A., Živný, S. (eds.): The Constraint Satisfaction Problem: Complexity and Approximability, Dagstuhl Follow-Ups, vol. 7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
27. Lovász, L.: Kneser's conjecture, chromatic number, and homotopy. J. Combin. Theory Ser. A **25**(3), 319–324 (1978). https://doi.org/10.1016/0097-3165(78)90022-5
28. Pippenger, N.: Galois theory for minors of finite functions. Discrete Math. **254**(1), 405–419 (2002). https://doi.org/10.1016/S0012-365X(01)00297-7

29. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, STOC 1978, pp. 216–226. ACM, New York (1978). https://doi.org/10.1145/800133.804350
30. Zhuk, D.: A proof of CSP dichotomy conjecture. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 331–342, October 2017. https://doi.org/10.1109/FOCS.2017.38