A graph of a relational structure and Constraint Satisfaction Problems

Andrei A. Bulatov
Computing Laboratory, University of Oxford, Oxford, UK
e-mail: Andrei.Bulatov@comlab.ox.ac.uk

Abstract

In the constraint satisfaction problem $\mathrm{CSP}(\mathcal{H})$ *corresponding to a finite relational structure* $\mathcal{H}$*, the aim is to decide, given a relational structure* $\mathcal{G}$*, whether there exists a homomorphism from* $\mathcal{G}$ *to* $\mathcal{H}$*. In [5], we proved that if* $\mathcal{H}$ *is a conservative structure, then it can be associated with a complete edge-3-colored graph whose vertex set is the universe of* $\mathcal{H}$*. The complexity and a solution algorithm for* $\mathrm{CSP}(\mathcal{H})$ *strongly depend on certain properties of the associated graph. In this paper we show how a similar edge-3-colored graph can be defined for an arbitrary finite relational structure* $\mathcal{H}$*. Then we study properties of the defined graph and find a solution algorithm for* $\mathrm{CSP}(\mathcal{H})$*, where* $\mathcal{G}(\mathcal{H})$ *satisfies some restrictions. The latter result substantially generalizes the results of [1, 8, 12, 25, 27] concerning max-closed constraints and constraints with a 2-semilattice, semigroup or conservative groupoid polymorphism. Finally, we complete the study of the complexity of maximal constraint languages started in [11].*

1. Introduction

The Constraint Satisfaction Problem (CSP) is a way of expressing simultaneous requirements on values of a specified set of variables. Other useful representations of the CSP are the problem of deciding whether there exists a homomorphism between two relational structures, and the problem of deciding if a given conjunctive formula has a model.

In its full generality, the CSP is NP-complete [30, 31] and, thus presumed to be algorithmically intractable. To cope with the intractability of the CSP, researchers have devoted considerable research efforts to both the design of heuristics algorithms for the CSP and the search for restrictions onto the general CSP, which lead to tractable problems. One way to express such restrictions is to limit the possible form of relational structures involved in the homomorphism form of the CSP. In this form, the CSP involves two relational structures, called the instance and the template. Restrictions on both sides have been widely studied. The ultimate goal in this research direction is to identify those classes of relational structures that give rise to a tractable (that is solvable in polynomial time) constraint satisfaction problem from those which do not. We refer to this problem as to the *classification problem*.

The classification problem has been completely solved for restrictions onto the 'left side' of the CSP, that is when the form of the instance is limited while the template is arbitrary [17, 22] (see also related works [20, 21]). The pioneering work in the study of 'right side' restrictions, i.e. constraint satisfaction problems with the template fixed and the instance arbitrary, is the paper [35], in which the classification problem was completely solved for Boolean CSP (that is with a 2-element template). The main result of [35] is known as *Schaefer's dichotomy theorem*. This subclass of the CSP is equivalent to the propositional GENERALIZED SATISFIABILITY problem. In [2], the classification problem was solved for CSPs with a 3-element template. In the case of a bigger template the classification problem is believed to be very hard and, in spite of substantial progress achieved during last decade [5, 10, 16, 19, 23, 25, 27, 28], it withstands all attacks.

Remarkably, all studied restricted CSP turn out to be either tractable or NP-complete. In [19], it was conjectured that such a dichotomy holds for the complexity of an arbitrary restricted CSP, and in [10], a possible criteria for tractable problems has been suggested. For a short survey of dichotomy results the reader is referred to [14].

One of the most successful approach in tackling the classification problem, the algebraic approach, exploits the idea that the complexity of a right-side restricted CSP is completely determined by the *polymorphisms* of the relational structures allowed, that is homomorphisms of Cartesian powers of a structure to the structure itself. The algebraic approach has been used in [2] and [5] to solve the classification problem for 3-element and conservative structures, and also in [1, 3, 15, 25, 27] to identify a number of new tractable CSPs.

The main goal of this paper is to further develop the algebraic approach. To explain our results we need to recall some details concerning the results of [5]. In that paper we have obtained a dichotomy theorem for *conservative* constraint satisfaction problems (called also *list* CSPs). Recall that a relational structure $\mathcal{H}$ is said to be *conservative* if every subset of its universe is a relation in $\mathcal{H}$. The main tool used in [5] was an edge-colored complete graph $\mathcal{G}(\mathcal{H})$ associated with a conservative relational structure $\mathcal{H}$. The color of an edge reflects the local structure of $\mathcal{H}$, or rather the local structure of its polymorphisms. There are three ways the polymorphisms may act locally; these ways correspond to the three types of Boolean operations occurring in the algebraic version of Schaefer's dichotomy theorem (see Section 2.2) and to the three possible colors of edges of $\mathcal{G}(\mathcal{H})$. Many properties of $\mathcal{H}$ and the corresponding constraint satisfaction problem including a solution algorithm are completely determined by the coloring of this graph. For example, if one of the colors (corresponding to affine polymorphisms) does not occur in the graph, the corresponding CSP has bounded width and, therefore, can be solved using standard propagation techniques.

In this paper, we generalize the definition of $\mathcal{G}(\mathcal{H})$ so that it can be associated to an arbitrary finite relational structure $\mathcal{H}$. This graph is not complete in the general case, but still connected, edge-3-colored and the colors correspond to the three Boolean operations. Then we establish several properties of $\mathcal{G}(\mathcal{H})$, which hopefully will make it possible to use this graph in the study of general relational structures in a similar way it has been used in the study of conservative structures. In particular, we prove that omitting the color corresponding to affine polymorphisms is a necessary condition for a constraint satisfaction problem to have bounded width, and show the converse in some important particular cases. Finally, making use of the obtained results we complete the study of the complexity of maximal constraint languages started in [11] by providing a complete classification theorem.

2. Definitions and preliminaries

2.1. Constraint Satisfaction Problem

The set of all n-tuples with components from a set D is denoted D^n. The ith component of a tuple $\mathbf{a}$ will be denoted by $\mathbf{a}[i]$. Any subset of D^n is called an n-ary *relation* on D; the set of all relations on D is denoted by $\mathbb{R}_D$; and a *constraint language* on D is an arbitrary set $\Gamma \subseteq \mathbb{R}_D$. The Constraint Satisfaction Problem has been introduced by Montanary (see [31]).

Definition 1 (Constraint Satisfaction Problem) The *constraint satisfaction problem (*CSP*)* over a constraint language Γ, denoted $\mathrm{CSP}(\Gamma)$, is defined to be the decision problem with instance $\mathcal{P} = (V, D, \mathcal{C})$, where

- V is a set of *variables*;
- D is a set of *values* (sometimes called a domain); and
- $\mathcal{C}$ is a set of *constraints*, $\{C_1, \ldots, C_q\}$,

in which the constraint $C_i \in \mathcal{C}$ is a pair $\langle s_i, R_i\rangle$ where s_i is a tuple of variables of length m_i, called the *constraint scope*, and R_i an m_i-ary relation on D, called the *constraint relation*.

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

Is was observed in [19] that the Constraint Satisfaction Problem can be equivalently reformulated in the form of the Homomorphism Problem. To do this we need more definitions.

A *vocabulary* is a set of relational symbols $(R_i)_{i\in I}$ each of which has a fixed arity. A *relational structure* over the vocabulary $(R_i)_{i\in I}$ is a tuple $\mathcal{H} = (H; (R_i^{\mathcal{H}})_{i\in I})$ such that H is a non-empty set, called the *universe* of $\mathcal{H}$, and each $R_i^{\mathcal{H}}$ is a relation on H having the same arity as the symbol R_i. Notice that in the standard definition of a relational structure, the vocabulary is always finite. However, we allow structures to have an arbitrary vocabulary, as such structures are often more relevant in the study of the CSP. If $J \subseteq I$ then the structure $\mathcal{H}' = (H; (R_i^{\mathcal{H}})_{i\in J})$ is said to be a *reduct* of $\mathcal{H}$. If J is finite then $\mathcal{H}'$ is said to be a *finite reduct*. Let $\mathcal{G}, \mathcal{H}$ be relational structures over the same vocabulary $R_1, \ldots, R_n$. A *homomorphism* from $\mathcal{G}$ to $\mathcal{H}$ is a mapping $\varphi\colon G \to H$ from the universe of $\mathcal{G}$ (the *instance*) to the universe H of $\mathcal{H}$ (the *template*) such that, for every relation $R^{\mathcal{G}}$ of $\mathcal{G}$ and every tuple $(\mathbf{a}[1], \ldots, \mathbf{a}[m]) \in R^{\mathcal{G}}$, we have $(\varphi(\mathbf{a}[1]), \ldots, \varphi(\mathbf{a}[m])) \in R^{\mathcal{H}}$.

Definition 2 *Let $\mathcal{H}$ be a relational structure over a vocabulary $(R_i)_{i\in I}$. In the* homomorphism problem associated with $\mathcal{H}$ ($\mathrm{CSP}(\mathcal{H})$)*, the question is, given a finite reduct $\mathcal{H}'$ of $\mathcal{H}$ over a vocabulary $(R_i)_{i\in J}$, $J \subseteq I$, and a structure $\mathcal{G}$ over the same vocabulary, whether there exists a homomorphism from $\mathcal{G}$ to $\mathcal{H}'$.*

If the vocabulary of a relational structure $\mathcal{H}$ is finite, then every finite reduct of $\mathcal{H}$ can be supposed to be equal to $\mathcal{H}$ and, therefore, it can be excluded from the input. In this case $\mathrm{CSP}(\mathcal{H})$ is called *non-uniform*. Otherwise the problem is said to be *uniform*.

The intuition behind the mentioned equivalence between the constraint satisfaction problem and the homomorphism problem is that the instance in the latter problem represents the variables and the constraint scopes, while the template represents the domain of values and the constraint relations. Moreover, the homomorphisms between the structures are precisely the solutions to the constraint satisfaction problem. Thus the problems $\mathrm{CSP}(\Gamma)$ and $\mathrm{CSP}(\mathcal{H})$, where $\mathcal{H} = (H; (R_i^{\mathcal{H}})_{i\in I})$ and $\Gamma = \{R_i^{\mathcal{H}} \mid i \in I\}$, are equivalent. In this paper we mostly use the terminology of constraint languages.

We shall be concerned with distinguishing between those constraint languages (relational structures) which give rise to *tractable* problems, i.e., problems for which there exists a polynomial-time solution algorithm, and those which do not. A relational structure $\mathcal{H}$ is said to be *tractable* if $\mathrm{CSP}(\mathcal{H}')$ is tractable for every finite reduct $\mathcal{H}'$ of $\mathcal{H}$; it is said to be *NP-complete* if $\mathrm{CSP}(\mathcal{H}')$ is NP-complete for a certain finite reduct of $\mathcal{H}$. A constraint language $\Gamma = \{R_i \mid i \in I\}$ is called *tractable* if the relational structure $\mathcal{H}_\Gamma = (H; (R_i)_{i\in I})$ is tractable. The language Γ is said to be NP-complete if $\mathcal{H}_\Gamma$ is NP-complete.

Problem (Classification problem) *Characterize tractable relational structures [constraint languages].*

We use the standard correspondence between relations and predicates defined on the same set. In particular, we use the same symbol for a relation and for the corresponding predicate. In [25, 27], it has been shown that adding to a constraint language relations derived using certain rules does not change the complexity of the corresponding CSP. Let Γ be a constraint language. The set of relations derivable from Γ is defined to be the set of relations definable by *primitive positive formulas* (*pp-formulas* for short), that is existential conjunctive formulas, involving the relations of Γ and the equality relation:

Definition 3 *For any set of relations Γ over D, the set $\mathsf{def}(\Gamma)$ consists of all relations that can be expressed using*

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

2. conjunction, and

3. existential quantification.

We say that a relation R is *definable* in a constraint language Γ if $R \in \mathsf{def}(\Gamma)$.

Example 1 (Multiplication of binary relations) Let R_1, R_2 be binary relations on a set H. Then the relation $R_1 \circ R_2$, the *product* of R_1, R_2, is the relation definable by the pp-formula $(R_1 \circ R_2)(x,y) = \exists z(R_1(x,z) \wedge R_2(z,y))$.

The connection between pp-formulas and complexity is provided by the following result.

Proposition 1 ([25, 27]) *Let Γ be a constraint language on a finite set. If Γ is tractable then $\mathsf{def}(\Gamma)$ is tractable. If $\mathsf{def}(\Gamma)$ is NP-complete then Γ is NP-complete.*

Unary definable relations (that is subsets) and definable equivalence relations play a special role in our study. Let Γ be a constraint language on a set D. Slightly abusing terminology[1] we call a *subalgebra* of Γ a unary relation definable in Γ, and a *congruences* of Γ an equivalence relation definable in Γ. For a subset $B \subseteq D$, the language $\Gamma|_B$ is defined to be $\{R|_B \mid R \in \Gamma\}$, where $R|_B = R \cap B^m$, R is m-ary. A congruence of a subalgebra B is an equivalence relation $T \in \mathsf{def}(\Gamma)$ on B. For an equivalence relation T and $a \in D$, the class of T containing a is denoted by $a/_T$ and the set of all classes of T by $D/_T$. The *quotient language* $\Gamma/_T$ is defined to be $\Gamma/_T = \{R/_T \mid R \in \Gamma\}$, where $R/_T = \{(\mathbf{a}[1]/_T, \ldots, \mathbf{a}[m]/_T) \mid (\mathbf{a}[1], \ldots, \mathbf{a}[m]) \in R\}$.

Proposition 2 ([9, 10]) *Let Γ be a constraint language, B and T its subalgebra and congruence respectively.*

(1) If Γ is tractable then so are $\Gamma|_B$ and $\Gamma/_T$.

(2) If $\Gamma|_B$ or $\Gamma/_T$ are NP-complete then Γ is NP-complete.

2.2. Polymorphisms

Our another main tool is polymorphisms. Every relational structure $\mathcal{H}$ has a collection of associated operations on the same universe. The unary operations associated with the structure are widely known: they are the *endomorphisms* of $\mathcal{H}$ that is homomorphisms of the structure into itself. We shall use operations of arbitrary arity.

An n-ary operation f *preserves* an m-ary relation R (or f is a *polymorphism* of R, or R is *invariant* under f) if, for any $(a_{11}, \ldots, a_{m1}), \ldots, (a_{1n}, \ldots, a_{mn}) \in R$, the tuple $(f(a_{11}, \ldots, a_{1n}), \ldots, f(a_{m1}, \ldots, a_{mn}))$ belongs to R. If f preserves every relation from a constraint language Γ, we say that f is a polymorphism of Γ. The set of all polymorphisms of Γ is denoted by $\mathsf{Pol}(\Gamma)$. For instance, every member of $\mathsf{Pol}(\mathbb{R}_D)$ is a *projection*, that is an operation $f(x_1, \ldots, x_n)$ such that there is $i \in \{1, \ldots, n\}$ and $f(x_1, \ldots, x_n) = x_i$ for any choice of $x_1, \ldots, x_n \in D$. If C is a set of operations then $\mathsf{Inv}(C)$ denotes the set of relations which are invariant under every operation from C.

Example 2 Let $K_{k,\ell}$ be a complete bipartite graph with a bipartition V_1, V_2, where $|V_1| = k$, $|V_2| = \ell$. We show that the ternary operation d defined by the rule $d(x,y,z) = x$ if y,z belong to the same component of the bipartition, $d(x,y,z) = y$ if x,z belong to the same component of the bipartition but y does not, $d(x,y,z) = z$ if x,y belong to the same component of the bipartition but z does not, is a polymorphism of $K_{k,\ell}$. To do this we have to check that, for any edges $(a_1,b_1), (a_2,b_2), (a_3,b_3)$ from $K_{k,\ell}$, the tuple

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f(a_1,a_2,a_3) \\ f(b_1,b_2,b_3) \end{pmatrix} = f\left(\begin{pmatrix} a_1 \\ b_1 \end{pmatrix}, \begin{pmatrix} a_2 \\ b_2 \end{pmatrix}, \begin{pmatrix} a_3 \\ b_3 \end{pmatrix} \right)$$

is an edge of $K_{k,\ell}$ or, equivalently, that a, b belong to different components of the bipartition. Indeed, if $a_2, a_3 \in V_1$ then $b_2, b_3 \in V_2$, hence, $(a,b) = (a_1, b_1)$. All other cases are similar. □

The connection between polymorphisms and definable relations is established by the following

1 In fact, unary and equivalence definable relations are subalgebras and congruences of the *universal algebra* related to Γ. See [10] for details.

Proposition 3 (see e.g. [32] or [33]) *If* Γ *is a constraint language on a finite set, then* $\mathsf{Pol}(\Gamma) = \mathsf{Pol}(\mathsf{def}(\Gamma))$ *and* $\mathsf{def}(\Gamma) = \mathsf{Inv}\,\mathsf{Pol}(\Gamma)$.

Making use of Proposition 1 we infer the following connection between polymorphisms and complexity.

Corollary 1 ([25, 27]) *Let* Γ_1, Γ_2 *be constraint languages on the same set,* Γ_1 *is finite. If* $\mathsf{Pol}(\Gamma_2) \subseteq \mathsf{Pol}(\Gamma_1)$ *then* $\mathrm{CSP}(\Gamma_1)$ *is polynomial time reducible to* $\mathrm{CSP}(\Gamma_2)$.

In a sense, Corollary 1 amounts to say that, in the study of $\mathrm{CSP}(\Gamma)$, a reasonable strategy is to concentrate on polymorphisms of constraint languages rather than constraint languages themselves. In many cases, this strategy has proved to be successful, see e.g. [2, 5, 7, 10].

In all known cases, the NP-completeness of CSPs can be explained using a special type of polymorphisms. An operation $f(x_1, \ldots, x_n)$ on a set A is said to be *essentially unary surjective* if there is a bijection $g: A \to A$ and $i \in \{1, \ldots, n\}$ such that $f(x_1, \ldots, x_n) = g(x_i)$ for any $x_1, \ldots, x_n \in A$.

Proposition 4 ([25, 27]) *If every polymorphism of a constraint language* Γ *is an essentially unary surjective operation, then* Γ *is NP-complete.*

In the case of constraint languages on a 2-element set, Proposition 4 characterizes all NP-complete languages. A binary operation f on a set A is said to be a *semilattice operation*, if it satisfies the conditions: (a) $f(a, a) = a$ (idempotency); (b) $f(a, b) = f(b, a)$ (commutativity); (c) $f(f(a, b), c) = f(a, f(b, c))$ (associativity), for all $a, b, c \in A$. A ternary operation g is said to be a *majority* operation if $g(a, a, b) = g(a, b, a) = g(b, a, a) = a$, for all $a, b \in A$. It is called an *affine* operation if $g(x, y, z) = x - y + z$, where $+, -$ are the operations of a certain Abelian group.

Proposition 5 (Schaefer's Dichotomy Theorem) *A constraint language* Γ *on the 2-element set* $\{0, 1\}$ *is tractable if and only if* $\mathsf{Pol}(\Gamma)$ *contains an operation which is not essentially unary surjective. The latter condition holds if and only if* $\mathsf{Pol}(\Gamma)$ *contains one of the following:*

– *the constant* 0 *or constant* 1 *operations;*
– *the conjunction or disjunction operations (they are semilattice operations);*
– *the majority operation* $(x \vee y) \wedge (y \vee z) \wedge (z \vee x)$*;*
– *the affine operation* $x - y + z \pmod 2$.

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

For constraint languages on a bigger set, the sufficient condition of NP-completeness can be weakened. Recall that if $\mathcal{H}$ is a substructure of $\mathcal{H}'$, then a *retraction* of $\mathcal{H}'$ to $\mathcal{H}$ is a homomorphism $\varphi: \mathcal{H}' \to \mathcal{H}$ such that $\varphi(h) = h$ for all $h \in \mathcal{H}$. A structure is a *core* if it does not admit a retraction to a proper substructure. As is easily seen every retraction is a unary operation preserving every relation of $\mathcal{H}$ and satisfying the condition $\varphi(\varphi(h)) = \varphi(h)$ for $h \in \mathcal{H}'$. Therefore, we can extend this notion to constraint languages: a language Γ is said to be a core if every unary polymorphism f such that $f(f(x)) = f(x)$ is the identity operation. It is easy to see that every structure $\mathcal{H}'$ contains a unique (up to isomorphism) substructure $\mathcal{H}$ which is a core and admits a retraction $\varphi: \mathcal{H}' \to \mathcal{H}$; we call $\mathcal{H}$ the core of $\mathcal{H}'$. Note that if $\mathcal{H}$ is the core of $\mathcal{H}'$ then $\mathrm{CSP}(\mathcal{H})$ and $\mathrm{CSP}(\mathcal{H}')$ are polynomial time equivalent (see e.g. [27]). Therefore we may assume that all relational structures and constraint languages we study are cores.

A relation of the form $K_a = \{(a)\}$, that is a unary relation containing only one tuple, is called a *constant relation*. If Γ is a constraint language on a set D then Γ^c denotes the language $\Gamma \cup \{K_a \mid a \in D\}$.

Proposition 6 ([9, 10]) *A constraint language* Γ*, which is a core, is tractable [NP-complete] if and only if* Γ^c *is tractable [NP-complete].*

Every polymorphism f of the language Γ^c satisfies the identity $f(x, \ldots, x) = x$ for any $x \in D$. Operations possessing this property are said to be *idempotent*.

Let C be a set of operations on a set A, B a subset of A and T an equivalence relation on A such that every operation from C preserves B and T. Then we denote $C|_B = \{f|_B \mid f \in C\}$, $C/_T = \{f/_T \mid f \in C\}$, where $f|_B$ denotes the restriction of f onto B, and $f/_T$ denotes the operation on $A/_T$ defined as follows: for any $a_1, \ldots, a_n \in A$, $f/_T(a_1/_T, \ldots, a_n/_T) = (f(a_1, \ldots, a_n))/_T$. We tend to use f instead of $f|_B$ and $f/_T$ whenever it does not cause a confusion. It is well known and easy to prove that if Γ is a constraint language, B is its subalgebra and T is its congruence, then $(\mathsf{Pol}(\Gamma))|_B \subseteq \mathsf{Pol}(\Gamma|_B)$ and $(\mathsf{Pol}(\Gamma))/_T = \mathsf{Pol}(\Gamma/_T)$.

Proposition 7 ([9, 10]) *Let* Γ *be a constraint language and* Γ *is a core. Let also* B *be a subalgebra of* Γ^c *and* T *an equivalence relation on* B *definable in* Γ^c*. If every operation from* $((\mathsf{Pol}(\Gamma))|_B)/_T$ *is essentially unary surjective, then* Γ *is NP-complete.*

The property stated in Proposition 7 is the only reason known so far for the NP-completeness of a constraint language. Therefore the following conjecture seems to be plausible [10].

Conjecture 1 *Let* Γ *be a constraint language and let* Γ *be a core. The language* Γ *is tractable if and only if,*

(∗) *for any subalgebra* B *of* Γ^c*, and any equivalence relation* T *on* B *definable in* Γ^c*, the set* $((\mathsf{Pol}(\Gamma))|_B)/_T$ *contains an operation which is not essentially unary surjective.*

Otherwise it is NP-complete.

3. A graph of a constraint language

3.1. Three types of edges

In the rest of the paper we will assume that every constraint language we consider contains all constant relation and, hence, every polymorphism is an idempotent operation. By Proposition 6, this assumption does not cause any loss of generality.

A constraint language Γ on a set D is said to be *conservative* if every unary relation on D belongs to Γ. In [5], we proved that a conservative language Γ is tractable if and only if, for any 2-element subset $B \subseteq D$, there is a polymorphism f of Γ such that $f|_B$ is one of the three types of idempotent operations listed in Proposition 5: a semilattice operation, a majority or affine operation. Otherwise Γ is NP-complete. Therefore, every tractable conservative language can be associated with a complete graph whose edges are colored with 3 colors: an edge ab is *red* if there is $f \in \mathsf{Pol}(\Gamma)$ such that $f|_{\{a,b\}}$ is a semilattice operation, ab is *yellow* if it is not red and there is $f \in \mathsf{Pol}(\Gamma)$ such that $f|_{\{a,b\}}$ is a majority operation, and ab is *blue* if it is not red or yellow and there is $f \in \mathsf{Pol}(\Gamma)$ such that $f|_{\{a,b\}}$ is an affine operation. It is also shown in [5] that many properties of $\mathrm{CSP}(\Gamma)$ including a solution algorithm strongly depend on the structure of the associated graph.

The first result of this paper states that a similar graph can be associated with an arbitrary constraint language. Let Γ be a constraint language on a set D and $B \subseteq D$. The smallest subalgebra of Γ containing B is called the *subalgebra generated* by B and denoted by $\langle B\rangle$. We introduce a graph $\mathcal{G}(\Gamma)$ of Γ as follows. The vertex set is the set D. A pair ab of vertices is an edge if and only if there exists a congruence θ of $\langle a,b\rangle$ and a polymorphism f of Γ such that either $f/_\theta$ (or rather $(f|_{\langle a,b\rangle})/_\theta$) is an affine operation on $\langle a,b\rangle/_\theta$, or $f/_\theta$ is a semilattice operation on $\{a/_\theta, b/_\theta\}$, or $f/_\theta$ is a majority operation on $\{a/_\theta, b/_\theta\}$.

The color of an edge is defined in the same way as for conservative languages[2]. If there exists a congruence θ and a polymorphism $f \in \mathsf{Pol}(\Gamma)$ such that $f/_\theta$ is a semilattice operation on $\{a/_\theta, b/_\theta\}$ then ab is said to be *red* or to have *semilattice type*. An edge ab is *yellow* or of *majority type* if it is not red and there are a congruence θ and $f \in \mathsf{Pol}(\Gamma)$ such that $f/_\theta$ is a majority operation on $\{a/_\theta, b/_\theta\}$. Finally, ab is *blue* or of *affine type* if it is not red or yellow and there are a congruence θ and $f \in \mathsf{Pol}(\Gamma)$ such that $f/_\theta$ is an affine operation on $\langle a,b\rangle/_\theta$.

Note that, for every edge ab of $\mathcal{G}(\Gamma)$, there is the associated pair $a/_\theta, b/_\theta$ from the quotient language. We will need both of these types of pairs and will sometimes call $a/_\theta b/_\theta$ a *thick* edge and ab a *thin* edge (see Fig. 1). The smallest congruence certifying the type of an edge ab will be denoted by θ_{ab}. If Γ is a conservative language, then $\mathcal{G}(\Gamma)$ coincides with the graph introduced in [5], because in this case $\langle a,b\rangle = \{a,b\}$ for any $a,b \in D$ and the only proper congruence of $\langle a,b\rangle$ is the equality relation. In particular, for conservative languages every thin edge is also a thick one and vice versa.

2 As we mentioned in the Introduction the color of an edge reflects the local structure of polymorphisms. Another way of typing the local structures of universal algebras and theory based on it constitute a major development in algebra [24]. There is a connection between that concept and the concept we introduce, and this connection can be used e.g. when proving Propositions 8 and 9, and Lemma 1. However, this connection is ambiguous and not very strong.

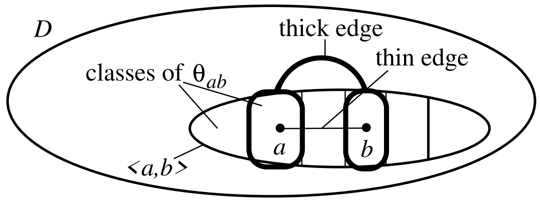

Figure 1. Thin and thick edges

We show that $\mathcal{G}(\Gamma)$ possesses several properties that make it a useful tool in the study of the complexity of $\mathrm{CSP}(\Gamma)$. The first one is connectedness.

Proposition 8 *A constraint language Γ on D satisfies condition $(*)$ if and only if every $a,b \in D$ are connected in the subgraph of $\mathcal{G}(\Gamma)$ induced by $\langle a,b\rangle$.*

Example 3 Let $D = \{0,1,2\}$ and $\Gamma = \{R_1, R_2, R_3\}$, where the binary relations R_1, R_2, R_3 are defined by lists of tuples (written vertically):

$$R_1 = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \end{pmatrix},\ R_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},\ R_3 = \begin{pmatrix} 2 & 0 & 2 \\ 0 & 2 & 2 \end{pmatrix}.$$

It can be straightforwardly checked that the operation f defined by its Cayley table

$f(x,y)$	0	1	2
0	0	1	2
1	1	1	0
2	2	0	2

is a polymorphism of Γ. (In fact, f occurs in [4]; in that paper it is called operation (6).) We have: $\langle 0,1\rangle = \{0,1\}$, $\langle 0,2\rangle = \{0,2\}$, $\langle 1,2\rangle = D$; the only congruence of $\langle 1,2\rangle$ such that $1,2$ belong to its different classes is the equality relation; f witnesses that 01 and 02 are red edges (thin and thick at the same time); 12 cannot be a red edge because no semilattice operation on $\{1,2\}$ preserves R_1; however, the operation $g(x,y,z) = f(f(x,f(y,z)), f(f(x,y),z))$ is a majority operation on $\{1,2\}$. Thus, $\mathcal{G}(\Gamma)$ is the graph shown in Fig. 2. Note also that this graph was implicitly used in [4] to prove the tractability of Γ. $\Box$

For a conservative language Γ satisfying $(*)$, every edge of $\mathcal{G}(\Gamma)$ is a subalgebra. This is not so in the general case. However, it is possible to transform Γ so that the obtained language Γ' satisfies condition $(*)$ and every thick edge of Γ' having semilattice or majority type forms a subalgebra. We say that $\mathcal{G}(\Gamma)$ is *red* [*red/yellow*]-*connected* if every two

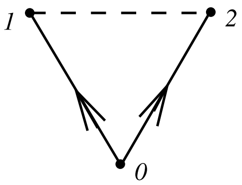

Figure 2. $\mathcal{G}(\Gamma)$; red edges are drawn solid, yellow edges are dotted

vertices a, b from $\mathcal{G}(\Gamma)$ are connected in the subgraph induced by $\langle a, b\rangle$ by a path consisting of red [red and yellow] edges.

Proposition 9 *Let Γ be a constraint language satisfying condition ($*$), ab an edge of $\mathcal{G}(\Gamma)$ of semilattice or majority type, and $R_{ab} = a/_{\theta_{ab}} \cup b/_{\theta_{ab}}$ the thick edge ab.*

(1) $\Gamma \cup \{R_{ab}\}$ satisfies condition ($*$).

(2) If ab is red and $\mathcal{G}(\Gamma)$ is red-connected, then $\mathcal{G}(\Gamma \cup \{R_{ab}\})$ is red-connected.

(3) If ab is yellow and $\mathcal{G}(\Gamma)$ is red/yellow-connected, then $\mathcal{G}(\Gamma \cup \{R_{ab}\})$ is red/yellow-connected.

As the following example shows, adding an edge of affine type to a language Γ can destroy condition ($*$) and even make a tractable language NP-complete.

Example 4 Let $D = \{0, 1, 2\}$, $+$ denote the operation of addition modulo 3 and $\Gamma = \{R\}$, where $R = \{(a, b, c, d) \in D^4 \mid a + b = c + d\}$. It is well known (see e.g. [33]) and easy to prove that the polymorphisms of Γ^c are the operations of the form $\alpha_1 x_1 + \ldots + \alpha_n x_n$, where $\alpha_1, \ldots, \alpha_n$ are integers and $\alpha_1 + \ldots + \alpha_n = 1 \pmod 3$. Therefore, for any $a, b \in D$, $\langle a, b\rangle = D$, the only proper congruence of $\langle a, b\rangle$ is the equality relation and ab is an edge of affine type.

Since the affine operation $x - y + z$ is a polymorphism of Γ^c, the problem CSP(Γ^c) can be solved by Gaussian elimination [27]. Take an edge of $\mathcal{G}(\Gamma^c)$, say 01 and a polymorphism $f(x_1, \ldots, x_n) = \alpha_1 x_1 + \ldots + \alpha_n x_n$ of Γ^c. If f preserves $\{0, 1\}$, then, for any $i \in \{1, \ldots, n\}$, we have $f(0, \ldots, 0, 1, 0, \ldots, 0) = \alpha_i \in \{0, 1\}$ (1 is on the ith place). Furthermore, if $\alpha_i, \alpha_j = 1$, then $f(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0) = \alpha_i + \alpha_j = 2 \notin \{0, 1\}$ (1s are on the ith and jth places). Thus, only one of the αs is non-zero, which means that f is a projection. Hence, every polymorphism of $\Gamma^c \cup \{01\}$ is a projection and CSP$(\Gamma^c \cup \{01\})$ is NP-complete. $\Box$

Proposition 9 amounts to say that, since our goal is to prove that every constraint language satisfying condition ($*$) is tractable, we may restrict our attention to languages Γ such that every thick edge of $\mathcal{G}(\Gamma)$ is a subalgebra. In Section 3.2 we show that in some cases one may use only thin edges.

To conclude this section we prove that the polymorphisms certifying the type of edges can be significantly unified (cf. Proposition 2 from [5]).

Proposition 10 *Let Γ be a constraint language. There are polymorphisms f, g, h of Γ such that*

$f|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}$ is a semilattice operation if ab is a red edge, it is the first projection if ab is a yellow or blue edge;

$g|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}$ is a majority operation if ab is a yellow edge, it is the first projection if ab is a blue edge, and $g|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}(x, y, z) = f|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}(x, f|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}(y, z))$ if ab is red;

$h|_{\langle ab\rangle/_{\theta_{ab}}}$ is an affine operation if ab is a blue edge, it is the first projection if ab is a yellow edge, and $h|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}(x, y, z) = f|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}(x, f|_{\{a/_{\theta_{ab}}, b/_{\theta_{ab}}\}}(y, z))$ if ab is red.

Example 3 (continued) Let us reconsider the constraint language Γ from Example 3. By Proposition 9, since 12 is a yellow edge, the constraint language $\Gamma' = \Gamma \cup \{12\}$ satisfies condition ($*$). The operations f, g, h satisfying the conditions of Proposition 10 can be chosen as follows: g is the operation obtained in Example 3, $f(x, y) = g(x, x, y)$ (the binary operation defined in Example 3 does not fit, because it does not preserve $\{1, 2\}$) and $h(x, y, z) = f(x, f(y, z))$.

3.2. Red edges

In this section we focus on red edges of the graph $\mathcal{G}(\Gamma)$. Note first that if one fix a polymorphism f such that f is a semilattice operation on every thick red edge of $\mathcal{G}(\Gamma)$, then one can define an orientation of every red edge. A red edge ab is oriented from a to b if $f(a/_{\theta_{ab}}, b/_{\theta_{ab}}) = f(b/_{\theta_{ab}}, a/_{\theta_{ab}}) = b/_{\theta_{ab}}$. For instance, the edges 01, 02 of the graph from Example 3 are oriented from 0 to 1 and 2 respectively. Clearly, the orientation strongly depends on the choice of the polymorphism f. The graph $\mathcal{G}(\Gamma)$ oriented accordingly to a polymorphism f will be denoted by $\mathcal{G}_f(\Gamma)$. We then can define *red-connected* and *strongly red-connected* components of $\mathcal{G}_f(\Gamma)$. We will also use the natural order on the set of strongly red-connected components of $\mathcal{G}_f(\Gamma)$: for components A, B, $A \le B$ if there is a directed path in $\mathcal{G}_f(\Gamma)$ consisting of red edges and connecting a vertex from A with a vertex from B. In Section 4.2, we show that certain restrictions on the set of strongly red-connected components of $\mathcal{G}_f(\Gamma)$ suffice for the tractability of CSP(Γ).

First we show that those red edges ab for which θ_{ab} is not the equality relation can be thrown out of the graph $\mathcal{G}(\Gamma)$ such that the graph remains connected. Therefore, we can assume that every red edge ab is both thin and thick, and that f is a semilattice operation on $\{a, b\}$.

Proposition 11 *Let Γ be a constraint language satisfying $(*)$ and $\mathcal{G}'(\Gamma)$ the subgraph of $\mathcal{G}(\Gamma)$ obtained by omitting red edges ab such that θ_{ab} is not the equality relation. Then $\mathcal{G}'(\Gamma)$ is connected. If $\mathcal{G}(\Gamma)$ is red-connected then $\mathcal{G}'(\Gamma)$ is red-connected. If $\mathcal{G}(\Gamma)$ is red/yellow-connected then $\mathcal{G}'(\Gamma)$ is red/yellow-connected.*

The graph $\mathcal{G}'(\Gamma)$ oriented accordingly to a binary polymorphism f will be denoted by $\mathcal{G}'_f(\Gamma)$.

Using Proposition 11 we are able to impose more restrictions on the polymorphism f.

Proposition 12 *Let Γ be a constraint language on a set D. There is a binary polymorphism f of Γ such that f is a semilattice operation on every thick red edge of $\mathcal{G}(\Gamma)$ and, for any $a, b \in D$, either $a = f(a, b)$ or the pair $(a, f(a, b))$ is a red edge of $\mathcal{G}'(\Gamma)$.*

4. Problems of bounded width

4.1. Omitting blue edges and bounded width

One of the most frequently used propagation algorithms for CSPs tries to guess a solution to the whole problem by finding 'partial' solutions for certain sets of variables. In some cases this strategy is successful.

For an n-ary relation R, $\mathbf{a} \in R$, and $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, by $\mathrm{pr}_I \mathbf{a}$ we denote the tuple $(\mathbf{a}[i_1], \ldots, \mathbf{a}[i_k])$, and by $\mathrm{pr}_I R$ the set $\{\mathrm{pr}_I \mathbf{b} \mid \mathbf{b} \in R\}$. If $\langle s, R\rangle$ is a constraint in a CSP, $s = (v_1, \ldots, v_n)$, R is an n-ary relation, and $W = (v_{i_1}, \ldots, v_{i_k}) \subseteq s$, then it will be convenient to use $\mathrm{pr}_W R$ instead of $\mathrm{pr}_{\{i_1,\ldots,i_k\}} R$.

Let $\mathcal{P} = (V; A; \mathcal{C})$ be an instance of the CSP, and $W \subseteq V$. By $\mathcal{P}_W$ we denote the *restricted problem*, that is the problem instance defined as $(W; A; \mathcal{C}')$ where, for every $\langle s, R\rangle \in \mathcal{C}$, there is $\langle s', R'\rangle \in \mathcal{C}'$ with $s' = s \cap W$, and $R' = \mathrm{pr}_{s'} R$. Every solution of $\mathcal{P}_W$ is said to be a *partial solution* of $\mathcal{P}$ on W. We use $\mathcal{S}_W$ to denote the set of all partial solutions on W.

The problem instance $\mathcal{P}$ is said to be *k-consistent* if, for any $(k-1)$-element subset W of V and any $v \in V$, every partial solution on W can be extended to a partial solution on $W \cup \{v\}$. The problem instance $\mathcal{P}$ is said to be *k-minimal* if, for any k-element subset W of V, there is a constraint $\langle s, R\rangle \in \mathcal{C}$ such that $W \subseteq s$, and for any $\langle s, R\rangle \in \mathcal{C}$, we have $\mathrm{pr}_{s\cap W} R = \mathrm{pr}_{s\cap W} \mathcal{S}_W$.

Any problem instance $\mathcal{P}$ can be modified to obtain a k-consistent or k-minimal instance without changing the set of solutions. In order to obtain a k-consistent problem, for every k-element subset $W \subseteq V$, we add to $\mathcal{P}$ the constraint $\langle W, \mathcal{S}_W\rangle$ and then we just repeat the following procedure until the obtained problem instance coincides with the previous one: for each k-element subset $W \subseteq V$, replace the constraint relation of $\langle W, \mathcal{S}_W\rangle$ with the updated set of partial solutions. To obtain a k-minimal problem, we run a similar procedure, each iteration of which does the following: for each k-element set $W \subseteq V$, calculate the relation $\mathcal{S}_W$, and then, for each constraint $\langle s, R\rangle \in \mathcal{C}$, remove from R every tuple $\mathbf{a}$ such that $\mathrm{pr}_{s\cap W} \mathbf{a} \notin \mathrm{pr}_{s\cap W} \mathcal{S}_W$. The obtained instance $\mathcal{P}'$ is k-minimal. The instances resulting from these procedures are referred to as the k-consistent and k-minimal instance *associated* with $\mathcal{P}$, respectively. As is easily seen the solution sets of the associated instances are equal to that of the original instance.

A class $\mathbf{C}$ of constraint satisfaction problems is said to have *width k* if any problem instance $\mathcal{P}$ from $\mathbf{C}$ has a solution if and only if the k-consistent problem associated with $\mathcal{P}$ contains no constraint with empty constraint relation. If $\mathbf{C}$ has width k for a certain k, then $\mathbf{C}$ is said to have *bounded width*. Analogously, a class $\mathbf{C}$ of constraint satisfaction problems is said to have *relational width k* if any problem instance $\mathcal{P}$ from $\mathbf{C}$ has a solution if and only if the k-minimal problem associated with $\mathcal{P}$ contains no constraint with empty constraint relation. If $\mathbf{C}$ has relational width k for a certain k, then $\mathbf{C}$ is said to have *bounded relational width*. Problems of this and similar types have been widely studied in computer science, see, for example, [13, 18, 19, 26]. We should also mention that problems of bounded width admit several alternative characterization, including those in terms of *Datalog* and *existential pebble games* (see, e.g., [19, 29].

As is easily seen, every k-minimal problem is also k-consistent. Therefore, every problem of width k has also relational width k; and, hence, every problem of bounded width have bounded relational width. The converse is true in the important case of problems of the form $\mathrm{CSP}(\Gamma)$, where Γ is finite.

Recall that a *module* is a concept generalizing that of a vector space, in which scalars belong to a certain ring rather than a field.

Lemma 1 *If ab is a blue edge of $\mathcal{G}(\Gamma)$, then every polymorphism of Γ can be represented on $\langle a, b\rangle/_{\theta_{ab}}$ as an operation $\alpha_1 x_1 + \ldots + \alpha_n x_n$ of a certain module with $\alpha_1 + \ldots + \alpha_n = 1$.*

Theorem 1 *Let Γ be a constraint language. If $\mathrm{CSP}(\Gamma)$ is of bounded relational width, then $\mathcal{G}(\Gamma)$ contains no blue edges.*

Proof (sketch): It is not hard to see that if $\Gamma' \subseteq \mathsf{def}(\Gamma)$ and $\mathrm{CSP}(\Gamma)$ has bounded relational width then $\mathrm{CSP}(\Gamma')$ also has bounded relational width.

Now suppose for contrary that ab is a blue edge in $\mathcal{G}(\Gamma)$. Then every polymorphism f of Γ on $B = \langle a, b\rangle/_{\theta_{ab}}$ can be represented as an operation $\alpha_1 x_1 + \ldots + \alpha_n x_n$ of a certain module. As is easily seen the relations R_1, R_2, where $R_1 = \{(x, y, z, t) \in B^4 \mid x + y = z + t\}$ and $R_2 = \{(x, y, z, t) \in B^4 \mid x + y = z + t + c\}$, c is not the zero element of the module, belong to $\mathsf{def}(\Gamma)$; and $\mathrm{CSP}(\{R_1, R_2\})$ is reducible to $\mathrm{CSP}(\Gamma)$.

For any n we consider an instance $\mathcal{P} = (V; B; \mathcal{C})$ of $\mathrm{CSP}(\{R_1, R_2\})$ defined as follows: $V = \{x_1, y_1, \ldots, x_n, y_n\}$, $\mathcal{C}$ contains the costraint $((x_i, y_i, x_{i+1}, y_{i+1}), R_1)$ for $i \in \{1, \ldots, n-1\}$ and also the constraint $((x_n, y_n, x_1, y_1), R_2)$. Then $\mathcal{P}$ is n-minimal, all its constraint relations are nonempty and yet it has no solution. $\square$

The problem of characterizing CSPs having bounded (relational) width is still open and it is not clear in which terms such a characterization can be achieved. We hope that graphs $\mathcal{G}(\Gamma)$ and Theorem 1 will be helpful in tackling this problem. In particular, it is proved in [6] and can be easily deduced from the results of [2] that the converse to Theorem 1 holds for conservative constraint languages and languages on a 3-element set. We therefore make the following conjecture.

Conjecture 2 (Bounded width conjecture) *Let Γ be a constraint language satisfying condition (∗). The problem $\mathrm{CSP}(\Gamma)$ has bounded relational width if and only if $\mathcal{G}(\Gamma)$ contains no blue edges.*

4.2. Languages with a unique maximal strongly connected component

In this section we consider constraint languages Γ with a binary polymorphism f such that, for every subalgebra B of Γ, the subgraph of $\mathcal{G}'_f(\Gamma)$ induced by B has a unique maximal strongly red-connected component. This condition we shall call the *maximal red component condition*.

Theorem 2 *If a constraint language Γ satisfies the maximal red component condition, then $\mathrm{CSP}(\Gamma)$ has relational width 3.*

Some already studied constraint languages satisfy the maximal red component conditions. They include languages with a semilattice [27, 25], *binary commutative conservative* [8] or *2-semilattice* polymorphism [1]. A binary operation f is said to be commutative conservative if it is commutative and $f(a,b) \in \{a,b\}$ for any a,b. A binary operation f is said to be a 2-semilattice operation if it is idempotent, commutative and, for any a,b, we have $f(a, f(a,b)) = f(f(a,a),b)$. It is not hard to see that 2-semilattice operation is the most general concept among them; every semilattice and binary commutative conservative operation is also a 2-semilattice operation.

If Γ has a 2-semilattice polymorphism f, then f is a semilattice operation on $\{a, f(a,b)\}$ and $\{b, f(a,b) = f(b,a)\}$. This means that $\mathcal{G}'_f(\Gamma)$ is red-connected. Moreover, if a,b belong to different maximal strongly red-connected components A and B, then $f(a,b)$ belongs to a strongly red-connected component C such that $A \le C$ and $B \le C$, a contradiction with the maximality of A,B. The same argument is valid for any subalgebra of Γ, thus, Γ satisfies the maximal red component condition.

5. Maximal constraint languages

In this section we apply the results obtained above to obtain a complexity classification of maximal constraint languages. It is clear that, for any constraint language Γ on a set D such that $\mathsf{def}(\Gamma) = \mathbb{R}_D$, the problem $\mathrm{CSP}(\Gamma)$ is NP-complete. We introduce constraint languages such that $\mathsf{def}(\Gamma)$ is just smaller than $\mathbb{R}_D$.

A *maximal* constraint language is a constraint language Γ, such that $\mathsf{def}(\Gamma) \ne \mathbb{R}_D$, but $\mathsf{def}(\Gamma \cup \{R\}) = \mathbb{R}_D$ for every $R \notin \mathsf{def}(\Gamma)$. This can also be stated in terms of polymorphisms: a constraint language Γ is maximal if and only if $\mathsf{Pol}(\Gamma)$ contains a nontrivial operation, but, for any $R \notin \mathsf{def}(\Gamma)$, every polymorphism of $\Gamma \cup \{R\}$ is a projection.

In order to give a characterization of maximal constraint languages we need to define one more type of operations. An operation $f(x_1, \ldots, x_n)$, $n \ge 3$, is said to be a *semiprojection* if there exists $i \in \{1, \ldots, n\}$ such that for all $a_1, \ldots, a_n \in D$ we have $f(a_1, \ldots, a_n) = a_i$ whenever $|\{a_1, \ldots, a_n\}| < n$, but f is not a projection.

To date, there is no complete description of maximal constraint languages over an arbitrary finite set. However, the following strong classification theorem follows from a result of I.G.Rosenberg [34].

Theorem 3 ([34]) *For every maximal constraint language Γ, the language $\mathsf{def}(\Gamma)$ has the form $\mathsf{Inv}(\{f\})$, where the operation f is of one of the following types:*

– a unary operation which is either a permutation or else acts identically on its range;

– a binary idempotent operation which is not a projection;

– a majority operation;

– an affine operation;

– a semiprojection.

The complexity of some maximal constraint languages over an arbitrary finite set has already been determined.

Theorem 4 ([11, 27]) *(1) A constraint language $\mathsf{Inv}(\{f\})$ is tractable if f is an operation of one of the following types: a constant operation, a semilattice operation, a majority operation, an affine operation.*

(2) A constraint language Γ such that $\mathsf{def}(\Gamma) = \mathsf{Inv}(\{f\})$ is NP-complete, whenever f is either a permutation, or a unary non-constant operation acting identically on its range, or a semiprojection.

Thus, the only type of maximal constraint languages Γ whose complexity is not yet known is $\mathsf{def}(\Gamma) = \mathsf{Inv}(\{f\})$, where f is a binary idempotent operation. In [11], it has been proved that a maximal constraint language of this type is either NP-complete or f can be chosen to be commutative. In the same paper, the tractability of $\mathsf{Inv}(\{f\})$ has been proved for commutative operations on a 3-element set, and, in [8, 1], it has been proved for commutative operations on a 4-element set. In this paper we complete a classification of maximal constraint languages.

Theorem 5 *If a maximal constraint language* Γ *has a commutative binary polymorphism, then* $\mathrm{CSP}(\Gamma)$ *is tractable.*

Proof: Let Γ be a language on a set D and h a commutative polymorphism of Γ. We consider two cases.

CASE 1. $\mathcal{G}(\Gamma)$ contains a red edge.

By Propositions 10 and 12, there is a binary polymorphism f such that f is a semilattice operation on every red edge of $\mathcal{G}'(\Gamma)$ and, for any $a, b \in D$, the pair $(a, f(a,b))$ is a red edge of $\mathcal{G}'_f(\Gamma)$. Since f is not a projection, $\Gamma = \mathsf{Inv}(\{f\})$. We need the procedure from [11] that allows us to derive a commutative polymorphism of Γ from f.

Let us consider the operations $f_n(x,y)$ where $f_1(x,y) = f(x,y)$ and $f_{n+1}(x,y) = f(f_n(x,y), f_n(y,x))$. Using the definition of a semilattice operation it can be verified that $f_n|_{\{a,b\}} = f|_{\{a,b\}}$ for every red edge ab of $\mathcal{G}'(\Gamma)$. It is easy to prove by induction that $\mathsf{Inv}(\{f\}) \subseteq \mathsf{Inv}(\{f_n\}) \subseteq \mathbb{R}_D$. Since f_n is not a projection, $\mathsf{Inv}(\{f_n\}) \neq \mathbb{R}_D$. Therefore, as $\mathsf{Inv}(\{f\})$ is maximal, for every n, we have $\mathsf{Inv}(\{f\}) = \mathsf{Inv}(\{f_n\})$.

There is n such that f_n is either commutative or $f_n|_{\{a,b\}}$ is a projection for some $a, b \in D$ (see [11] for details). In the former case we are done; in the latter case we get a contradiction. Indeed, since $\mathsf{Inv}(\{f_n\}) = \mathsf{def}(\Gamma)$, we have $\{a, b\} \in \mathsf{def}(\Gamma)$, and h certifies that ab is a red edge. However, f_n is a semilattice operation on every red edge. Let us denote the obtained commutative operation by f'.

Next we show that Γ satisfies the maximal red component condition. Let for contradiction A, B be different maximal strongly red-connected components of a subalgebra of Γ and $a \in A$, $b \in B$. Then by the choice of f, the sequences $(a, f_1(a,b)), (f_1(a,b), f_2(a,b)), \dots, (f_{n-1}(a,b), f'(a,b))$ and $(b, f_1(b,a)), (f_1(b,a), f_2(b,a)), \dots, (f_{n-1}(b,a), f'(b,a))$ are paths from a, b to $f'(a,b) = f'(b,a)$, a contradiction with the maximality of A and B.

The result now follows from Theorem 2.

CASE 2. $\mathcal{G}(\Gamma)$ contains no red edge.

Suppose first that $\mathcal{G}(\Gamma)$ contains a yellow edge. Let $B' = \{a', b'\}$, where $a' = a/_{\theta_{ab}}$, $b' = b/_{\theta_{ab}}$, be a thick yellow edge. By Proposition 9, we may assume that every polymorphism of Γ preserves B'. Notice that, as Γ is a maximal constraint language and $\Gamma \cup \{a' \cup b'\}$ has a nontrivial homomorphism, $a' \cup b' \in \mathsf{def}(\Gamma)$. Since h is commutative on $\{a, b\}$, it is also commutative on B'. Therefore, h is a semilattice operation on B', a contradiction with the assumption that B' is yellow.

Let ab be a blue edge and $B' = \langle a, b\rangle/_{\theta_{ab}}$. Then every polymorphism of Γ can be represented on B' as an operation of a module. In particular, every binary polymorphism f is idempotent and, hence, can be represented on B' in the form $f(x,y) = \alpha x + \beta y$ with $\alpha + \beta = 1$.

To complete the proof we make use of the result obtained by B.Szczepara [36], who proved that if f is a binary operation such that $\mathsf{Inv}(\{f\})$ is a maximal constraint language, then f can be supposed either

– to be the operation $\frac{p+1}{2}(x + y)$, where $+$ is the operation of an Abelian group of prime order p, or

– to satisfy the identities $f(f(x,y),y) = y$ and $\underbrace{f(x, f(x, \dots f}_{n \text{ times}}(x,y)\dots)) = y$, or

– to satisfy the identities $f(f(x,y),y) = f(x,y)$ and $f(x, f(x,y)) = y$, or

– to satisfy the identities $f(f(x,y),y) = f(x,y)$ and $\underbrace{f(x, f(x, \dots f}_{n+1 \text{ times}}(x,y)\dots)) = \underbrace{f(x, f(x, \dots f}_{n \text{ times}}(x,y)\dots))$.

It can be straightforwardly checked that if an operation of the form $f(x,y) = \alpha x + \beta y$ satisfies one of the systems of identities listed above, then it is a projection, a contradiction with the assumption that $\mathsf{Inv}(\{f\})$ is maximal. Thus f is the operation $\frac{p+1}{2}(x + y)$ of an Abelian group. In this case the tractability of Γ follows from [25, 27]. $\square$

6. Conclusion

We have introduced a new object, an edge-colored graph, related to an arbitrary relational structure or a constraint language, and have started to investigate its properties. Even on this initial stage, this new construction makes it possible to generalize several known results and to make progress in the study of the complexity of the CSP. We strongly believe that, like its prototype for conservative structures, the graph we have introduced will play an important role in solving the classification problem for the CSP.

References

[1] A. Bulatov. Combinatorial problems raised from 2-semilattices. (Manuscript).

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

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

[4] A. Bulatov. Tractable constraints on a three-element set. Technical Report PRG-RR-02-06, Computing Laboratory, University of Oxford, Oxford, UK, 2002.

[5] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th Annual IEEE Simposium on Logic in Computer Science*, pages 321–330, Ottawa, Canada, June 2003. IEEE Computer Society.

[6] A. Bulatov. Tractable conservative constraint satisfaction problems. Technical Report PRG-RR-03-01, Computing Laboratory, University of Oxford, Oxford, UK, 2003.

[7] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science, FOCS'03*, pages 562–571, Cambridge, MA, USA, October 2003. IEEE Computer Society.

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

[9] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying complexity of constraints using finite algebras. Submitted to SIAM Journal of Computing, 2000.

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

[11] A. Bulatov, P. Jeavons, and A. Krokhin. The complexity of maximal constraint languages. In *Proceedings of the 33rd Annual ACM Simposium on Theory of Computing*, pages 667–674, Hersonissos, Crete, Greece, July 2001. ACM Press.

[12] A. Bulatov, P. Jeavons, and M. Volkov. Finite semigroups imposing tractable constraints. In Gracinda M.S.Gomes, Jean-Eric Pin, Pedro V.Silva, editor, *Semigroups, Algorithms, Automata and Languages*, pages 313–329. World Scientific, Singapore, 2002.

[13] M. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41:89–95, 1989.

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

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

[16] V. Dalmau. Constraint satisfaction problems in non-deterministic logarithmic space. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming, ICALP'02*, volume 2380 of *Lecture Notes in Computer Science*, pages 414–425. Springer-Verlag, 2002.

[17] V. Dalmau, P. Kolaitis, and M. Vardi. Constraint satisfaction, bounded treewidth, and finite variable logics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, CP'02*, Lecture Notes in Computer Science, pages 311–326. Springer-Verlag, 2002.

[18] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–107, 1992.

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

[20] G. Gottlob, L. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.

[21] G. Gottlob, L. Leone, and F. Scarcello. Hypertree decompositions: A survey. In *Proceedings 26th International Symposium on Mathematical Foundations of Computer Science, MFCS'01*, volume 2136 of *Lecture Notes in Computer Science*, pages 37–57. Springer-Verlag, 2001.

[22] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proceedings of the 44th Annual Simposium on Foundations of Computer Science*, pages 552–561, Cambridge, Massachusets, USA, October 2003. IEEE Computer Society.

[23] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Ser.B*, 48:92–110, 1990.

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

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

[26] P. Jeavons, D. Cohen, and M. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

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

[28] P. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61:302–332, 2000.

[29] P. Kolaitis and M. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings of the 17th National (US) Conference on Artificial Intellignece, AAAI'00*, pages 175–181, 2000.

[30] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

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

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

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

[34] I. Rosenberg. Minimal clones I: the five types. In *Lectures in Universal Algebra (Proc. Conf. Szeged 1983)*, volume 43 of *Colloq. Math. Soc. János Bolyai*, pages 405–427. North-Holland, 1986.

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

[36] B. Szczepara. *Minimal clones generated by groupoids*. PhD thesis, Université de Móntreal, 1996.