FINITE SEMIGROUPS IMPOSING TRACTABLE CONSTRAINTS

ANDREI BULATOV
Department of Mathematics and Mechanics, Ural State University,
620083 Ekaterinburg, RUSSIA
E-mail: Andrei.Bulatov@usu.ru

PETER JEAVONS
Computing Laboratory, University of Oxford
Oxford OX1 3QD, UNITED KINGDOM
E-mail: Peter.Jeavons@comlab.ox.ac.uk

MIKHAIL VOLKOV
Department of Mathematics and Mechanics, Ural State University,
620083 Ekaterinburg, RUSSIA
E-mail: Mikhail.Volkov@usu.ru

The Constraint Satisfaction Problem provides a framework in which a wide variety of combinatorial problems can be expressed in a natural way. In the present paper, after defining the general Constraint Satisfaction Problem and giving several concrete examples, we briefly survey a recent approach to studying constraint satisfaction problems which is based on characterizing the complexity of problem classes via finite algebras and their pseudovarieties. Then we investigate the case when the finite algebra assigned to a subclass of the constraint satisfaction problem with a finite set of values turns out to be a semigroup. Here we show that the subclass is tractable if and only if the assigned semigroup is a block group; otherwise it always contains an NP-complete problem.

Introduction

Many combinatorial problems can be naturally expressed in the framework of the Constraint Satisfaction Problem (CSP, for short). The aim in a CSP is to assign values to a given set of variables such that certain constraints are satisfied on the values that can be assigned simultaneously to some subsets of variables [31, 26]. A standard satisfiability problem [11], where the variables must be assigned Boolean values, gives a typical example of such problem.

Throughout the paper we assume that P$\neq$NP. Recall that problems from P are said to be *tractable*. The general CSP is known to be NP-hard, and hence, intractable [26, 31]. However, certain restrictions on the form of constraints may affect the complexity of the corresponding problem class and give rise to a tractable subclass of the general CSP. There is, therefore, a fundamental research direction aiming to recognize tractable subclasses of the CSP. A progress in this direction may provide efficient algorithms for those applications, which fall in one of the known tractable classes. The direction is also very important from a theoretical perspective, as it helps to clarify the boundary between tractability and intractability in a wide range of combinatorial search problems.

For the important special case of Boolean CSPs, Schaefer [38] managed to find the precise boundary between tractability and intractability. Namely, Schaefer established that for Boolean CSPs (which he called "Generalised Satisfiability Problems") there are exactly six different families of tractable constraints, and any problem involving constraints not contained in one of these six families is NP-complete. This important result is known as Schaefer's Dichotomy Theorem for Boolean relations.

Schaefer [38] raised the question of how this result could be generalised to larger domains (that is, sets with more than two elements). Several approaches tackling this problem have appeared recently. For instance, Feder and Vardi [10] used techniques from logic programming and group theory to identify three broad families of tractable constraints which include all of Schaefer's six classes. In a series of papers the second named author and his coauthors have characterized a number of tractable constraint classes over finite domains using invariance properties of relations [21, 16, 18, 19, 20, 17].

In [20] and then in [4, 5] it has been shown that, for any subclass of CSP over a finite set of values, there is a finite algebra such that the complexity of the subclass depends only on the corresponding algebra. The first aim of the present paper is to give a brief overview of this link between computational complexity theory and universal algebra. Our second aim is to analyze finite semigroups from the complexity-theoretical point of view. It is interesting and somewhat unexpected that these complexity-theoretical considerations lead to a class which is known to play a distinguished role in finite semigroup theory: the class of so-called block-groups [35].

The paper is organized as follows. In Section 1 we introduce the CSP, define some connected notions, and provide a few typical examples. Then in Section 2 we present the algebraic approach to the CSP. Section 3 contains a treatment of the semigroup case. We describe all finite semigroups such that the corresponding subclass of the CSP is tractable and deduce a semigroup version of the Dichotomy Theorem: if a subclass defined by a finite semigroup is not tractable, then it always contains an NP-complete problem. In conclusion we compare the semigroup case with the general situation.

The reader is referred to the books [11, 34] and [6, 2] for the basic notions of respectively complexity theory and universal algebra that we use below.

1 Constraint Satisfaction Problem

1.1 Definition

The 'constraint satisfaction problem' was introduced by Montanari [31] in 1974 and has been widely studied [26, 27, 3, 9, 28, 29, 41, 22, 25, 12, 10, 23].

Definition 1.1. *The* constraint satisfaction problem (CSP) *is the combinatorial decision problem with*

INSTANCE: *a triple* $(V, A, \mathcal{C})$ *where*

- V *is a set of* variables*;*
- A *is a set* [domain]*;*
- $\mathcal{C}$ *is a set of* constraints*.*
- *Each constraint* $C \in \mathcal{C}$ *is a pair* $\langle s, \varrho \rangle$*, where*
  - $s = (v_1, \ldots, v_m)$ *is a tuple of variables (whose length depends on* C*), called the* constraint scope*;*
  - $\varrho \subseteq A^m$ *is an* m*-ary relation over* A *called the* constraint relation*.*

QUESTION: *does there exist a* solution*, i.e. a function* $\varphi : V \to A$*, such that, for each constraint* $\langle s, \varrho \rangle \in \mathcal{C}$*, with* $s = (v_1, \ldots, v_m)$*, the tuple* $(\varphi(v_1), \ldots, \varphi(v_m))$ *belongs to* ϱ*?*

The size of a problem instance is the length of the encoding of all tuples in all constraints.

For some applications it is necessary to consider a more general version of the CSP, where each variable $v \in V$ has its own domain A_v and, for each constraint $\langle s, \varrho \rangle \in \mathcal{C}$ with the scope $s = (v_1, \ldots, v_m)$, the constraint relation ϱ is a subset in $A_{v_1} \times \cdots \times A_{v_m}$. This general version is usually referred to as the *multi-sorted* CSP in contrast to the *one-sorted* CSP as defined above. Even though we concentrate on the one-sorted version of the CSP in the present paper, we would like to mention that the general version admits a similar treatment.

1.2 Examples

The CSP provides a framework for a wide range of problems from various areas of computer science such as database theory [42, 24], temporal and spatial reasoning [39], machine vision [31], belief maintenance [8], technical design [33], natural language comprehension [1], programming language analysis [32], etc.

Here we restrict ourselves to a few examples showing how certain well known problems from logic, graph theory, linear algebra, planning can be naturally expressed as instances of the CSP.

Example 1.1. An instance of the standard propositional SATISFIABILITY problem [11, 34] is specified by giving a formula in conjunctive normal form, that is, a conjunction of clauses, and asking whether there are truth values for the variables which make the formula true.

Suppose that $\Phi = X_1 \wedge \cdots \wedge X_n$ is such a formula, where the X_i are clauses. The satisfiability question for Φ can be expressed as the instance $(V, \{0,1\}, \mathcal{C})$ of the CSP, where V is the set of all variables appearing in the clauses X_i, and $\mathcal{C}$ is the set of constraints $\{\langle s_1, \varrho_1\rangle, \ldots, \langle s_n, \varrho_n\rangle\}$, where each constraint $\langle s_k, \varrho_k\rangle$ is constructed as follows:

- $s_k = (x_1^k, \ldots, x_{m_k}^k)$ where $x_1^k, \ldots, x_{m_k}^k$ are the variables appearing in the clause X_k;
- $\varrho_k = \{0,1\}^{m_k} \setminus \{(a_1, \ldots, a_{m_k})\}$ where $a_i = 1$ if x_i^k is negated in X_k and $a_i = 0$ otherwise (i.e., ϱ_k consists of all m_k-tuples that make X_k true).

The solutions of this CSP instance are exactly the assignments which make the formula Φ true. Hence, any instance of SATISFIABILITY can be expressed as an instance of the CSP.

We notice that Example 1.1 shows that the CSP in general is NP-hard.

Example 1.2. An instance of GRAPH UNREACHABILITY consists of a graph $G = (V, E)$ and a pair of vertices, $v, w \in V$. The question is whether there is no path in G from v to w. This can expressed as the CSP instance

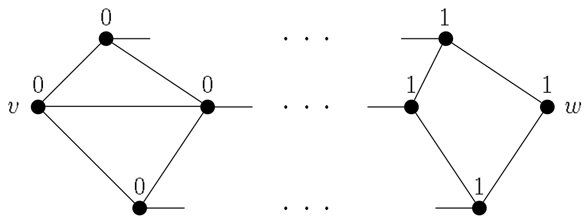

Figure 1. GRAPH UNREACHABILITY as an instance of the CSP

$(V, \{0,1\}, \mathcal{C})$ where

$$\mathcal{C} = \{\langle e, \{(0,0),(1,1)\}\rangle \mid e \in E\} \cup \{\langle v, \{0\}\rangle, \langle w, \{1\}\rangle\}.$$

Indeed, this CSP instance has a solution if and only if one can label the vertices of G by 0's and 1's such that v is labelled 0, w is labelled 1 and adjacent vertices are assigned equal labels, see Fig. 1. Clearly, this is possible if and only if there is no path from v to w.

Example 1.3. A system of linear equations over a field F can be expressed as the CSP instance $(V, F, \mathcal{C})$ where V is the set of variables of the system, and each constraint $\langle s_i, \varrho_i \rangle$ from $\mathcal{C}$ corresponds to an equation. Then s_i is the set of variables appearing in the equation, and ϱ_i is the set of solutions of the equation, that is, a hyperplane.

Example 1.4. A graph $G = (V, E)$ is said to be *interval* if its vertices can be represented by intervals of the real line such that two vertices are adjacent if and only if intervals which represent them have a common point, see Fig. 2. This notion plays an important role in planning and temporal reasoning. The

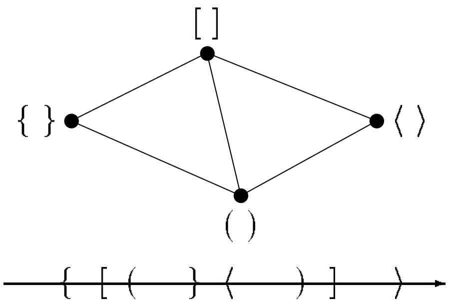

Figure 2. An interval graph

question whether a given graph G is interval is expressed by the CSP instance $(V, I(\mathbb{R}), \{\langle e, \varrho \rangle \mid e \in E\} \cup \{\langle (u,v), \tau \rangle \mid u \neq v,\ (u,v) \notin E\})$ where $I(\mathbb{R})$ is the set of real intervals, and ϱ and τ are the binary realtions defined as

$$\varrho = \{((a,b),(c,d)) \mid (a,b) \cap (c,d) \neq \varnothing\},$$
$$\tau = \{((a,b),(c,d)) \mid (a,b) \cap (c,d) = \varnothing\}.$$

All the examples above are essentially one-sorted. Multi-sorted CSPs arise naturally in the study of databases; in fact, even though it uses a rather different vocabulary, the notion of a relational database is easily seen to be equivalent to the notion of an instance of the multi-sorted CSP.

The list of examples can easily be extended. The reader is encouraged to translate his own favorite combinatorial problem to the CSP language, although sometimes this is a nontrivial exercise[a].

[a] Of course, it is known that any problem from NP can be expressed by an instance of SATISFIABILITY, and thus, by an instance of the CSP, see Example 1.1. However, we are seeking for simple and natural reductions only, as in the examples above.

1.3 Restricted classes of CSP

As already mentioned, the CSP is NP-hard in general. However, some restrictions on the form of constraints may ensure tractability of a problem class. There are two natural ways to restrict the CSP: first, restrictions on the form of the constraint relations; and second, restrictions on the way in which the constraint scopes may interact. In this paper we adopt the first way; some results and references concerning the second direction can be found in [12].

Definition 1.2. *For a given set of relations* Γ *over a set* A, $\mathrm{CSP}(\Gamma)$ *is defined to be the decision problem with*

INSTANCE: *An instance,* $P = (V, A, \mathcal{C})$, *of the CSP, in which, for each constraint* $\langle s, \varrho\rangle$, *the relation* ϱ *is an element of* Γ.

QUESTION: *Does* P *have a solution?*

The set Γ is said to be *tractable* if, for each finite subset $\Gamma' \subseteq \Gamma$, the class $\mathrm{CSP}(\Gamma')$ is tractable, Γ is said to be *NP-complete* if $\mathrm{CSP}(\Gamma')$ is NP-complete for some finite subset $\Gamma' \subseteq \Gamma$.

Many well-known combinatorial problems can be represented in the form $\mathrm{CSP}(\Gamma)$.

Example 1.5. An instance of GRAPH q-COLORABILITY consists of a graph $G = (V, E)$. The question is whether the vertices of G can be labelled with q colors so that adjacent vertices are assigned different colors, see Fig. 3. This can expressed as the CSP instance $(V, A, \{\langle e, \neq_A\rangle \mid e \in E\})$ where A is

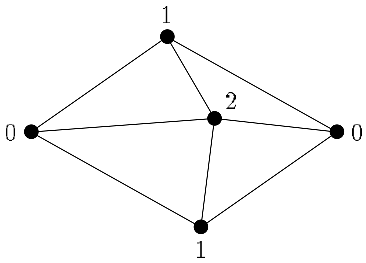

Figure 3. 3-coloring of a graph

a q-element set (of colors) and $\neq_A = \{(a, b) \in A^2 \mid a \neq b\}$ is the disequality relation on A. Observe that this is in fact an instance of $\mathrm{CSP}(\{\neq_A\})$. Conversely, we can assign a graph G to each instance $P = (V, A, \langle s, \neq_A\rangle)$ of $\mathrm{CSP}(\{\neq_A\})$ so that P has a solution if and only if G admits a coloring with q colors: let V and the set of all constraint scopes of P be respectively the vertex and the edge sets of G. Thus, we see that GRAPH q-COLORABILITY is precisely CSP($\{\neq_A\}$) with A being a q-element set.

We note that for $q \geq 3$ GRAPH q-COLORABILITY is known to be NP-complete [11, 34].

Example 1.6. The NOT-ALL-EQUAL SATISFIABILITY problem (NAE) [11, 34] is a restricted version of the standard SATISFIABILITY problem which remains NP-complete. An instance of NAE is a conjunction $\Phi = X_1 \wedge \cdots \wedge X_n$ of 3-literal clauses $X_i = z_1^i \vee z_2^i \vee z_3^i$ (recall that a *literal* is either a variable or the negation of a variable). The question is whether there are assignments of truth values for the variables appearing in Φ such that in no clause X_i all three literals become equal in truth value. (Clearly, such assignments make the formula Φ true so that NAE is indeed a version of SATISFIABILITY.) This can be expressed as the instance $(V, \{0,1\}, \mathcal{C})$ of CSP($\{\nu\}$), where ν is the ternary relation $\{(a,b,c) \mid \{a,b,c\} = \{0,1\}\}$ on $\{0,1\}$, V is the set of all literals appearing in the clauses X_i, and $\mathcal{C}$ consists of the constraints $\langle (z_1^i, z_2^i, z_3^i), \nu \rangle$ for each clause X_i and the constraints $\langle (x, \neg x, \neg x), \nu \rangle$ for each variable x appearing in Φ. Indeed, the latter constraints ensure that each solution of the instance is induced by an assignment of values for the variables while the former constraints guarantee that the values of the literals in X_i are not all equal under the assignment. Conversely, each instance of CSP($\{\nu\}$) can be easily converted into an equivalent instance of NAE. Thus, NAE can be identified with CSP($\{\nu\}$).

While the examples above give NP-complete classes, the next problem class is tractable in spite of the infinite family of relations involved.

Example 1.7. The family of all linear systems over a field F corresponds to the class CSP(H) where H is the class of all hyperplanes of finite-dimensional F-vector spaces (compare with Example 1.3).

2 Algebraic structure of CSP

Throughout the rest of the paper we assume the domains to be finite. The relation families, however, may (and often will) be infinite.

2.1 From arbitrary sets of relations to relational clones

The main goal in the study of CSP(Γ) is to find a way to distinguish between tractable and intractable sets of relations. In this connection, it is natural to ask how a tractable family of relations can be extended so that the extended set of relations remains tractable. In other words, which operations on relations do preserve tractability? An important class of such operations has been discovered in [15].

We make use of the standard correspondence between relations and logical predicates: to each m-ary relation $\varrho \subseteq A^m$ we assign the m-ary predicate on A which becomes true precisely on m-tuples from ϱ. Recall that an existential first order formula is said to be *primitive positive* if its quantifier-free part is a conjunction of predicates. These formulas can be used as operations that allow one to derive new relations from given ones.

Example 2.1. The standard product of two binary relations ϱ_1 and ϱ_2 may be defined by the following primitive positive formula:

$$(x, z) \in \varrho_1 \circ \varrho_2 \iff (\exists y)\ (x, y) \in \varrho_1 \wedge (y, z) \in \varrho_2.$$

A set of relations on a set A closed with respect to all operations defined by primitive positive formulas is called a *relational clone*. Given a set Γ of relations, the smallest relational clone containing Γ is said to be *generated by* Γ and denoted by $\langle\Gamma\rangle$.

Theorem 2.1 ([15]). *For any set of relations Γ over a finite set, and any finite set of relations $\Gamma' \subseteq \langle\Gamma\rangle$, there is a polynomial time reduction from* CSP(Γ') *to* CSP(Γ).

Corollary 2.1. *If a set of relation Γ is tractable, then the relational clone $\langle\Gamma\rangle$ is also tractable.*

In a sense, Corollary 2.1 amounts to saying that, in the study of CSP(Γ), a reasonable strategy is to concentrate on relational clones rather than arbitrary families of relations. In many cases, this strategy has proved to be successful: for example, in [15] it is shown that Schaefer's results [38] on CSPs over a two-element domain can easily be deduced from Theorem 2.1 and the classical description of clones on a two-element set [37].

2.2 From relational clones to finite algebras

An n-ary operation $f : A^n \to A$ on a set A *preserves* an m-ary relation $\varrho \subseteq A^m$ (or ϱ is *invariant* under f) if applying f to the rows of an arbitrary $m \times n$-matrix over A whose columns belong to ϱ, one always gets a column in ϱ:

$$\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \\ \cap\!\!\shortparallel & \cap\!\!\shortparallel & & \cap\!\!\shortparallel \\ \varrho & \varrho & & \varrho \end{array} \implies \begin{array}{c} f(a_{11}, a_{12}, \ldots, a_{1n}) \\ f(a_{21}, a_{22}, \ldots, a_{2n}) \\ \cdots\cdots \\ f(a_{m1}, a_{m2}, \ldots, a_{mn}) \\ \cap\!\!\shortparallel \\ \varrho \end{array} .$$

This can be also expressed by saying that $(\varrho; f)$ is a subalgebra of the m-th direct power of the algebra $(A; f)$. For a given set of operations, F, the set of all relations invariant under operations from F is denoted by $\operatorname{Inv} F$. Conversely, for a set of relations, Δ, the set of all operations preserving relations from Δ is denoted by $\operatorname{Pol} \Delta$.

It is well known [36, 40] that, for any finite algebra $\mathbb{A} = (A; F)$, the set $\operatorname{Inv} F$, that is, the set of subalgebras of finite direct powers of $\mathbb{A}$, forms a relational clone. Moreover, for any relational clone Δ over a finite set A, there is a finite algebra $\mathbb{A} = (A; F)$ such that $\Delta = \operatorname{Inv} F$. In fact, F can be chosen to be $\operatorname{Pol} \Delta$.

Thus, every finite algebra $\mathbb{A}$ determines a subclass CSP($\operatorname{Inv} F$) of the CSP which will also be denoted by CSP($\mathbb{A}$). We say that the algebra $\mathbb{A}$ is *tractable* [*NP-complete*] if the set $\operatorname{Inv} F$ is tractable [NP-complete]. Conversely, for any problem class of the form CSP(Γ), where Γ is a family of relations over a finite set A, there is a corresponding finite algebra (for instance, $(A; \operatorname{Pol} \Gamma)$), and the tractability [NP-completeness] of the problem class depends only on this algebra.

Example 2.2. Every left zero semigroup $\mathbb{A} = (A; \cdot)$ with $|A| > 1$ is NP-complete.

Recall that the multiplication in left zero semigroups is simply the projection on the left component: $x \cdot y = x$ for all $x, y \in A$. Clearly, this multiplication preserves **every** relation on A. Therefore if $|A| = 2$ the class CSP($\mathbb{A}$) contains the NP-complete problem CSP($\{\nu\}$), cf. Example 1.6, and for $|A| > 2$ we can refer to Example 1.5 since in this case CSP($\mathbb{A}$) contains the NP-complete problem CSP($\{\neq_A\}$).

The reader may be surprised by the fact that a fairly obvious algebra such as the two-element left zero semigroup, say, turns out to be intractable in our sense. However, there is no contradiction in that. The correspondence $F \mapsto \operatorname{Inv} F$, $\Delta \mapsto \operatorname{Pol} \Delta$ between operations and relations is in fact a Galois connection, so 'weak' operations preserve many relations (among which one can easily find NP-complete ones) while 'strong' operations that preserve fewer relations may lead to tractable constraints.

Thus, computational complexity theory suggests a fresh look at finite algebras, and describing algebraic properties which reflect in the tractability or the intractability of the corresponding problem classes has proved to be an interesting task which is highly non-trivial in its general setting. The papers [21, 16, 19, 20, 17, 4, 7, 5] provide a number of partial results going in this direction. In the next section we shall apply some of these results to classify tractable semigroups.

3 Tractable semigroups

Originally the theory and practice of the CSP dealt with sets of relations rather than algebras. It will be fair to say that from this point of view, finite semigroups do not arise naturally: given a set Γ of relations, the requirement that Γ is invariant under a binary associative operation appears to be rather exotic. However, in studying many important problems which can be posed for arbitrary algebras, semigroups constitute a 'golden middle' case that is sufficiently rich and diverse with respect to the problem under consideration and at the same time is not hopelessly difficult as is the general case. Classifying tractable algebras in general and even tractable groupoids seems to be hard enough; on the other hand, Feder and Vardi [10] proved that an arbitrary finite group is tractable. This suggests that it may be reasonable to have a look at the intermediate case of semigroups.

We have seen that non-trivial finite left zero semigroups are NP-complete (Example 2.2). Of course, the same conclusion holds for non-trivial finite right zero semigroups. The following observation from [4] allows one to enlarge the supply of NP-complete semigroups.

Lemma 3.1 **([4],** Theorem 3.9**).** *Let $\mathbb{A}$ be a finite algebra. If $\mathbb{A}$ is tractable then so is every subalgebra of $\mathbb{A}$. If $\mathbb{A}$ has an NP-complete subalgebra then $\mathbb{A}$ is NP-complete itself.*

Corollary 3.1. *If a finite semigroup $\mathbb{S} = (S;\cdot)$ contains two distinct idempotents e and f such that either $ef = e$ and $fe = f$ or $ef = f$ and $fe = e$, then $\mathbb{S}$ is NP-complete.*

Proof. The subsemigroup $(\{e, f\};\cdot)$ of $\mathbb{S}$ is a non-trivial left or right zero semigroup which is NP-complete. Now Lemma 3.1 applies. ☐

We conclude that every finite semigroup $\mathbb{S}$ which is not NP-complete must satisfy the following two conditions:

$$ef = e^2 = e \wedge fe = f^2 = f \Rightarrow e = f, \tag{1}$$

$$ef = f^2 = f \wedge fe = e^2 = e \Rightarrow e = f. \tag{2}$$

Finite semigroups satisfying (1) and (2) are called *block groups.* We refer the reader to [35] for a brilliant survey of the theory of block groups which reveals remarkable connections between block groups, progroup topology of the free monoid and recognizable languages. A jewel of the theory is a characterization of block groups in terms of power semigroups of groups due to Henckell and Rhodes [14], see also [13]. As the main result of the present paper shows, from the computational complexity point of view, block groups also play a distinguished role:

Theorem 3.1. *A finite semigroup is tractable if and only if it is a block group.*

Proof. The 'only if' part follows from Corollary 3.1. One of the crucial ingredients for the proof of the 'if' part is a recent result by Dalmau [7] which may be formulated as follows. By a *set function* on a set A we mean an arbitrary mapping from the power set $\mathcal{P}(A)$ to A. A set function $f : \mathcal{P}(A) \to A$ *preserves* an m-ary relation $\varrho \subseteq A^m$ if applying f to the sets of elements in the rows of an arbitrary $m \times n$-matrix over A whose columns belong to ϱ, one always gets a column in ϱ:

$$\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \\ \\ \Cap & \Cap & & \Cap \\ \varrho & \varrho & & \varrho \end{array} \implies \begin{array}{c} f(\{a_{11}, a_{12}, \ldots, a_{1n}\}) \\ f(\{a_{21}, a_{22}, \ldots, a_{2n}\}) \\ \cdots\cdots \\ f(\{a_{m1}, a_{m2}, \ldots, a_{mn}\}) \\ \\ \Cap \\ \varrho \end{array}.$$

Proposition 3.1 ([7]). *Let Γ be a set of relations on a finite set A. If there is a set function preserving all relations from Γ, then there exists a uniform algorithm that solves arbitrary instance from* $\mathrm{CSP}(\Gamma)$ *in polynomial time; in particular, Γ is tractable.*

Now let $\mathbb{S} = (S; \cdot)$ be a block group and let $|S| = k$. For arbitrary $s_1, \ldots, s_n \in S$, we consider the element

$$f(s_1, \ldots, s_n) = (s_1^{k!} \cdots s_n^{k!})^{k!}. \tag{3}$$

First we prove that the right-hand side of (3) depends only on the set $\{s_1, \ldots, s_n\}$; in other words, that f is in fact a set function. Let $e_i = s_i^{k!}$; it is known (and easy to check) that each e_i is an idempotent. Then it is easy to realize that in order to show that f is a set function, it suffices to verify that

$$(e_1 \cdots e_n)^{k!} = (e_{1\pi} \cdots e_{n\pi})^{k!} \tag{4}$$

for every permutation π of the set $\{1, \ldots, n\}$. We shall employ the following observation:

Lemma 3.2 ([30], proof of Proposition 2.3**).** *If $e_1, \ldots, e_n$ are idempotents of a k-element block group. Then for every $i = 1, \ldots, n$*

$$(e_1 \cdots e_n)^{k!} = (e_1 \cdots e_n)^{k!} e_1 \cdots e_i.$$

Now, for an arbitrary permutation π of the set $\{1, \ldots, n\}$, we have

$$
\begin{aligned}
(e_1 \cdots e_n)^{k!} &= (e_1 \cdots e_n)^{k!} e_1 \cdots e_{n\pi} && \text{by Lemma 3.2,} \\
&= e_1 \cdots e_{n\pi} (e_{n\pi+1} \cdots e_n e_1 \cdots e_{n\pi})^{k!} \\
&= (e_{n\pi+1} \cdots e_n e_1 \cdots e_{n\pi})^{k!} && \text{by the dual of Lemma 3.2,} \\
&= (e_{n\pi+1} \cdots e_n e_1 \cdots e_{n\pi})^{k!} e_{n\pi} && \text{by Lemma 3.2,} \\
&= (e_1 \cdots e_n)^{k!} e_{n\pi} \\
&= (e_1 \cdots e_n)^{k!} e_{1\pi} \cdots e_{n\pi} && \text{by the same trick applied} \\
& && \text{to } e_{(n-1)\pi}, \ldots, e_{1\pi}, \\
&= (e_1 \cdots e_n)^{k!} (e_{1\pi} \cdots e_{n\pi})^{k!} \\
&= (e_{1\pi} \cdots e_{n\pi})^{k!} && \text{by an analogous process} \\
& && \text{applied on the left.}
\end{aligned}
$$

Thus, the equality (4) is proved, and the formula (3) indeed defines a set function.

Let Γ be the set of all relations on S which are invariant under the multiplication in $\mathbb{S}$; then $\mathrm{CSP}(\Gamma) = \mathrm{CSP}(\mathbb{S})$ and the set function f respects all relations from Γ because by (3) f is expressed in terms of the multiplication. Hence Proposition 3.1 applies. $\square$

Let us list several consequences of Theorem 3.1 and its proof.

A set Γ of relations is said to be *globally tractable* if a uniform algorithm solves arbitrary instances from $\mathrm{CSP}(\Gamma)$ in polynomial time. (This requirement is stronger than the usual tractability which we defined in Subsection 1.3 as the existence, for each finite subset $\Gamma' \subseteq \Gamma$, of a polynomial time algorithm which solves the problem $\mathrm{CSP}(\Gamma')$.) The most well known example of a globally tractable set of relations is provided by linear systems over a field, cf. Example 1.7. We notice that our proof shows that the set of all relations which are invariant under the multiplication in a block group is globally tractable. In particular, constraints imposed by an arbitrary finite group are globally tractable — a conclusion which cannot be extracted from the tractability proof by Feder and Vardi [10].

Combining Theorem 3.1 and Corollary 3.1, we deduce a semigroup analogue of the Dichotomy Theorem: a finite semigroup is either tractable or NP-complete.

Given the multiplication table of a finite semigroup $\mathbb{S} = (S; \cdot)$, one can check if $\mathbb{S}$ satisfies the conditions (1) and (2) in quadratic (of $|S|$) time. Thus, by Theorem 3.1, the problem of recognizing the tractability of a semigroup is tractable.

We conclude with a short comparison between the semigroup case and the general situation. For arbitrary finite algebras, several tempting conjectures have been formulated in [4]. We reproduce here the most important of them.

An algebra $\mathbb{B} = (B; F)$ where F is a collection of permutations of the set B is said to be a *G-set.* A G-set $\mathbb{B}$ is non-trivial if $|B| > 1$. Since permutations preserve the disequality relation $\neq_B$ and also the not-all-equal relation ν (in the case $|B| = 2$), it follows from Examples 1.5 and Example 1.6 that every non-trivial G-set is NP-complete. The main conjecture of [4] basically means that a non-trivial G-set hides behind every NP-complete algebra. Recall that a *factor* (semigroupists would say *divisor*) of an algebra $\mathbb{A}$ is a homomorphic image of a subalgebra of $\mathbb{A}$.

G-set Conjecture. *Every NP-complete finite algebra $\mathbb{A}$ has a factor $\mathbb{A}'$ such that* $\mathrm{CSP}(\mathbb{A}') = \mathrm{CSP}(\mathbb{B})$ *for a non-trivial G-set $\mathbb{B}$.*

The next conjecture is inspired by Schaefer's Dichotomy Theorem for Boolean relations (which confirms the conjecture for two-element algebras):

Dichotomy Conjecture. *Every finite algebra is either tractable or NP-complete.*

It is known that if both the G-set conjecture and the Dichotomy Conjecture hold true, then so does also the following

Algorithmic Conjecture. *There exits a algorithm which, given the Cayley tables of a finite algebra $\mathbb{A} = (A; F)$, determines whether $\mathbb{A}$ is tractable in polynomial of $|A|$ time.*

Finally, we mention

Local-Global Conjecture. *Every tractable finite algebra is globally tractable.*

Returning to the semigroup case, we see that Theorem 3.1 and its consequences are consistent with all the above conjectures.

Acknowledgments

A preliminary version of this paper was presented in the third named author's lectures at the School on Algorithmic Aspects of the Theory of Semigroups and its Applications held at Centro Internacional de Matemática, Coimbra, Portugal, in May 2001 as a part of the Thematic term on Semigroups, Algorithms, Automata and Languages. He is very much indebted to the organizers of the term: Prof. Gracinda M. S. Gomes, Prof. Jean-Eric Pin and Prof. Pedro V. Silva for the invitation and for great help during his stay in Coimbra. The participation of the third named author in the School was financially supported by Fundaoção Calouste Gulbenkian (FCG), Fundaoção para a Ciência e a Tecnologia (FCT), Faculdade de Ciências da Universidade de Lisboa (FCUL) and Reitoria da Universidade do Porto. The research presented in the paper was also supported by Russian Education Ministry, grant E00-1.0-92, by the Russian Foundation of Basic Research, grant 01-01-00258, and by the INTAS through the Network project 99-1224 'Combinatorial and Geometric Theory of Groups and Semigroups and its Applications to Computer Science'.

References

[1] J. F. Allen, Natural Language Understanding, Benjamin Cummings, Menlo Park, 1994.

[2] J. Almeida, Finite Semigroups and Universal Algebra [Ser. in Algebra **3**], World Scientific, Singapore–New Jersey–London–Hong Kong, 1995.

[3] W. Bibel, *Constraint satisfaction from a deductive viewpoint*, Artif. Intelligence **35** (1988) 401–413.

[4] A. A. Bulatov, P. G. Jeavons, and A. A. Krokhin, *Constraint satisfaction problems and finite algebras*, in U.Montanari et al. (eds.), Proc. 27th Int. Colloq. on Automata, Languages and Programming — ICALP'00 [Lect. Notes Comp. Sci. **1853**], Springer-Verlag, Berlin–Heidelberg–N.Y., 2000, 272–278.

[5] A. A. Bulatov, P. G. Jeavons, and A. A. Krokhin, *Classifying the complexity of constraints using universal algebras*, SIAM J. Computing (submitted).

[6] S. Burris and H. P. Sankappanavar, A Course in Universal Algebra, Springer-Verlag, Berlin–Heidelberg–N.Y, 1981.

[7] V. Dalmau, Computational Complexity of Problems over Generalised Formulas, PhD thesis, Department LSI, Universitat Politècnica de Catalunya, Barcelona, 2000.

[8] R. Dechter and A. Dechter, *Structure-driven algorithms for truth maintenance*, Artif. Intelligence **82** (1996) 1–20.

[9] R. Dechter and J. Pearl, *Network-based heuristics for constraint satisfaction problems*, Artif. Intelligence **34** (1988) 1–38.

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

[11] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, San Francisco, 1979.

[12] M. Gyssens, P. Jeavons, and D. Cohen, *Decomposing constraint satis-*

faction problems using database techniques Artif. Intelligence **66** (1994) 57–89.

[13] K. Henckell, S. W. Margolis, J.-E. Pin, and J. Rhodes, *Ash's Type II Theorem, profinite topologies and Malcev products*, Int. J. Algebra and Computation **1** (1991) 411–436.

[14] K. Henckell and J. Rhodes, *The theorem of Knast, the PG = BG and Type II Conjectures*, in J.Rhodes (ed.), Monoids and Semigroups with Applications, World Scientific, Singapore, 1991, 453–463.

[15] P. G. Jeavons, *On the algebraic structure of combinatorial problems*, Theor. Comp. Sci. **200** (1998) 185–204.

[16] P. G. Jeavons and D. A. Cohen, *An algebraic characterization of tractable constraints*, in Computing and Combinatorics. 1st Int. Conf. — COCOON'95 (Xian, China, August 1995) [Lect. Notes Comput. Sci. **959**], Springer-Verlag, Berlin–Heidelberg–N.Y., 1995, 633–642.

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

[18] P. G. Jeavons, D. A. Cohen, and M. Gyssens, *A unifying framework for tractable constraints*, in Proc. 1st Int. Conf. on Constraint Programming — CP'95 (Cassis, France, September 1995) [Lect. Notes Comput. Sci. **976**], Springer-Verlag, Berlin–Heidelberg–N.Y., 1995, 276–291.

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

[20] P. G. Jeavons, D. A. Cohen, and J. K. Pearson, *Constraints and universal algebra*, Ann. Math. Artif. Intelligence **24** (1998) 51–67.

[21] P. G. Jeavons and M. C. Cooper, *Tractable constraints on ordered domains*, Artif. Intelligence **79** (1995) 327–339.

[22] L. Kirousis, *Fast parallel constraint satisfaction*, Artif. Intelligence, **64** (1993) 147–160.

[23] Ph. G. Kolaitis and M. Y. Vardi, *A game-theoretic approach to constraint satisfaction*, in Proc. 17th National Conf. on Artif. Intelligence and 12th Innovative Applications of Artif. Intelligence Conf. — AAAI'00/IAAI'00 (Austin, TX, July 30–August 3, 2000), MIT Press, Cambridge, 2000, 175–181.

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

[25] P. B. Ladkin and R. D. Maddux, *On binary constraint problems*, J. ACM **41** (1994) 435–469.

[26] A. K. Mackworth, *Consistency in networks of relations*, Artif. Intelligence **8** (1977) 99–118.

[27] A. K. Mackworth, *Constraint satisfaction*, in S.C.Shapiro and D.Eckroth

(eds.), Encyclopedia of Artificial Intelligence, Vol.I, John Wiley & Sons, N.Y., 1987, 285–293.

[28] A. K. Mackworth, *The logic of constraint satisfaction*, Artif. Intelligence **58** (1992) 3–20.

[29] A. K. Mackworth and E. C. Freuder, *The complexity of constraint satisfaction revisited*, Artif. Intelligence **59** (1993) 57–62.

[30] S. W. Margolis and J.-E. Pin, *Varieties of finite monoids and topology for the free monoid*, in K.Byleen et al. (eds.), Proc. of the 1984 Marquette Conf. on Semigroups, Marquette Univ., Milwaukee, 1984, 113–130.

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

[32] B. A. Nadel, *Constraint satisfaction in Prolog: Complexity and theory-based heuristics*, Information Sciences **83** (1995) 113–131.

[33] B. A. Nadel and J. Lin, *Automobile transmission design as a constraint satisfaction problem: Modelling the kinematics level*, Artif. Intelligence for Engineering Design, Analysis and Manufacturing **5** (1991) 137–171.

[34] C. H. Papadimitriou, Computational complexity, Addison-Wesley Publishing Company, Reading–Menlo Park–N.Y., 1994.

[35] J.-E. Pin, *$BG = PG$, a success story*, in J.Fountain (ed.), Semigroups, Formal Languages and Groups [NATO ASI Ser., Ser. C: Math. Phys. Sci. **466**], Kluwer Academic Publishers, Dordrecht–Boston–London, 1995, 33–47.

[36] R. Pöschel and L.A. Kalužnin, Funktionen- und Relationenalgebren, VEB Deutscher Verlag der Wissenschaften, Berlin 1979; [Math. Reihe **67**] Birkhäuser Verlag, Basel–Stuttgart, 1979.

[37] E. L. Post, The two-valued iterative systems of mathematical logic [Ann. Math. Studies **5**], Princeton Univ. Press, 1941.

[38] T. J. Schaefer, *The complexity of satisfiability problems*, in Proc. 10th Ann. ACM Symp. on Theory of Computing — STOC'78, Association for Computer Machinery, N.Y., 1978, 216–226.

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

[40] Á. Szendrei, Clones in universal algebra [Séminaire de Mathématiques Supérieurs **99**], Les Presses de l'Université de Montréal, Montréal, 1986.

[41] E. Tsang, Foundations of constraint satisfaction, Academic Press, London, 1993.

[42] M. Y. Vardi, *Constraint satisfaction and database theory: a tutorial*, in Proc. 19th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems — PODS'00 (Dallas, TX, May 2000), 2000 (electronic, see `http://www.acm.org/sigmod/pods/proc00/`).