A Dichotomy Theorem for Constraints on a Three-Element Set

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

Abstract

The Constraint Satisfaction Problem (CSP) provides a common framework for many combinatorial problems. The general CSP is known to be NP-complete; however, certain restrictions on the possible form of constraints may affect the complexity, and lead to tractable problem classes. There is, therefore, a fundamental research direction, aiming to separate those subclasses of the CSP which are tractable, from those which remain NP-complete.

In 1978 Schaefer gave an exhaustive solution of this problem for the CSP on a 2-element domain. In this paper we generalise this result to a classification of the complexity of CSPs on a 3-element domain. The main result states that every subclass of the CSP defined by a set of allowed constraints is either tractable or NP-complete, and the criterion separating them is that conjectured in [6, 8]. We also exhibit a polynomial time algorithm which, for a given set of allowed constraints, determines whether if this set gives rise to a tractable problem class. To obtain the main result and the algorithm we extensively use the algebraic technique for the CSP developed in [17] and [6, 8].

1. Introduction

In the Constraint Satisfaction Problem (CSP) [24] we aim to find an assignment to a set of variables subject to specified constraints. Many combinatorial problems appearing in computer science and artificial intelligence can be expressed as particular subclasses of the CSP. The standard examples include the propositional satisfiability problem, in which the variables must be assigned Boolean values, graph colorability, scheduling problems, linear systems and many others. One advantage of considering a common framework for all of these diverse problems is that it makes it possible to obtain generic structural results concerning the computational complexity of constraint satisfaction problems that can be applied in many different areas such as database theory [21, 33], temporal and spatial reasoning [30], machine vision [24], belief maintenance [11], technical design [26], natural language comprehension [1], programming language analysis [25], etc.

The general CSP is NP-complete; however, certain restrictions on the allowed form of the constraints involved may ensure tractability. Therefore, one of the main approaches in the study of the CSP is identifying tractable subclasses of the general CSP obtained in this way [14, 15, 9, 17, 29]. Developments in this direction provide an efficient algorithm solving a particular problem, if the problem falls in one of the known tractable subclasses, or assist in speeding up of general superpolynomial algorithms [12, 13, 22]. To formalize the idea of restricting the allowed constraints, we make use of the notion of a *constraint language* [16], which is simply a set of possible relations that can be used to specify constraints in a problem. We say that a constraint language is *tractable* [*intractable*] if the corresponding problem class is tractable [intractable. The ultimate goal of this research direction is to find the precise border between tractable and intractable constraint languages.

This goal was achieved by Schaefer [29] in the important case of Boolean constraints; he has characterised tractable constraint languages on a 2-element set, and proved that the rest are NP-complete. This Schaefer's result is known as Shaefer's Dichotomy Theorem. Dichotomy theorems are of particular interests in study of the CSP, because, on the one hand, they determine the precise complexity of constraint languages, and on the other hand, the a priori existence of a dichotomy result cannot be taken for granted. For a short survey of dichotomy results the reader is referred to [9].

The analogous problem, which is referred to as the *classification problem*, for the CSP in which the variables can be assigned more than 2 values, remains open since 1978, in spite of intensive efforts. For instance, Feder and Vardi, in [14], used database technique and group theory to identify some large tractable families of constraints; Jeavons and coauthors have characterised many tractable and NP-complete constraint languages using invariance properties of constraints [17, 18, 19]; in [8], a possible form of a dichotomy result for the CSP on finite domains was conjectured; in [7], a dichotomy result was proved for a certain type of constraint languages on a 3-element domain. In this paper we generalise the results of [29] and [7], and prove the dichotomy conjecture from [8] for the constraint satisfaction problem on a 3-element domain. In particular, we completely characterise tractable constraint languages in this case, and prove that the rest are NP-complete. The main result will be precisely stated at the end of Section 2.

The classification problem for constraint languages on a set containing more than 2 elements, even on a 3-element set, turns out to be much harder than that for the 2-element case. Besides the obvious reason that Boolean CSPs closely relate to various problems from propositional logic, and therefore, are much better investigated, there is another deep reason. As is showed in [19, 20, 17], when studying the complexity of constraint languages we may restrict ourselves with a certain class of languages, so called *relational clones*. There are only countably many relational clones on a 2-element set, and all of them are known [28]. However, the class of relational clones on a 3-element set already contains continuum many elements, and any its explicit characterization is believed to be unreachable.

Another problem tackled here is referred to, in [9], as the *meta-problem*: given a constraint language determine if this language gives rise to a tractable problem class. Making use of the dichotomy theorem obtained we exhibit an effective algorithm solving the meta-problem for the CSP on a 3-element domain.

The technique used in this paper relies upon the idea, that was developed in [8, 6, 17] (and also mentioned in [14] as a possible direction for future research), that algebraic invariance properties of constraints can be used for studying the complexity of the corresponding constraint satisfaction problems. The main advantage of this technique is that it allows us to employ structural results from universal algebra. The algebraic approach has proved to be very fruitful in identifying tractable classes of the CSP [2, 4, 18]. We strongly believe that the synthesis between complexity theory and universal algebra which we describe here is likely to lead to new results in both fields.

2. Algebraic structure of CSP classes

2.1. The Constraint Satisfaction Problem

The set of all n-tuples with components from a set A is denoted A^n. The ith component of a tuple $\mathbf{a}$ will be denoted $\mathbf{a}[i]$. Any subset of A^n is called an n-ary *relation* on A; and a *constraint language* on A is an arbitrary set of finitary relations on A.

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

Instance: V *is a set of* variables*;* A *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* $\langle s_i, R_i\rangle$ *with* s_i *is a tuple of variables of length* m_i*, called the* constraint scope*, and* $R_i \in \Gamma$ *an* m_i*-ary relation on* A*, called the* constraint relation*.*

Question: *is whether there exists a* solution *to* $(V; A; C)$*, that is, a function from* V *to* A*, such that, for each constraint in* $\mathcal{C}$*, the image of the constraint scope is a member of the constraint relation.*

We shall be concerned with distinguishing between those constraint languages which give rise to *tractable* problems (i.e., problems for which there exists a polynomial-time solution algorithm), and those which do not.

Definition 2 *A constraint language,* Γ *is said to be* tractable*, if* $\mathrm{CSP}(\Gamma')$ *is tractable for each finite subset* $\Gamma' \subseteq \Gamma$*. It is said to be* NP-complete*, if* $\mathrm{CSP}(\Gamma')$ *is NP-complete for some finite subset* $\Gamma' \subseteq \Gamma$.

By a Boolean constraint language we mean a constraint language on a 2-element set, usually $\{0, 1\}$. In [29], Schaefer has classified Boolean constraint languages with respect to complexity. This result is known as Schaefer's Dichotomy theorem.

Theorem 1 (Schaefer, [29]) *A constraint language,* Γ*, on* $\{0, 1\}$ *is tractable if one of the following conditions holds:*

(1) every R *in* Γ *contains* $(0, \ldots, 0)$*;*

(2) every R *in* Γ *contains* $(1, \ldots, 1)$*;*

(3) every R *in* Γ *is definable by a CNF formula in which each clause has at most one negated variable;*

(4) every R *in* Γ *is definable by a CNF formula in which each clause has at most one unnegated variable;*

(5) every R *in* Γ *is definable by a CNF formula in which each clause has at most two literals;*

(6) every R *in* Γ *is the solution space of a linear system over* $\mathrm{GF}(2)$*.*

Otherwise Γ *is NP-complete.*

More examples of both tractable and NP-complete constraint languages will appear later in this paper and can also be found in [8, 10, 14, 19]. It follows from Theorem 1 that every Boolean constraint language is either tractable or NP-complete; and so, there is no language of intermediate complexity. Some dichotomy results have been obtained for other variations of Boolean CSP [9]. The classification problem for larger domains is still open and seems to be very interesting and hard [14].

Problem 1 (classification problem) *Characterise all tractable constraint languages on finite domains.*

2.2. Algebraic structure of problem classes

Schaefer's technique heavily uses the natural representation of Boolean relations by propositional formulas. Such a representation does not exist for larger domains. Instead, we shall use algebraic properties of relations. In our algebraic definitions we mainly follow [23].

Definition 3 *An* algebra *is an ordered pair $\mathbb{A} = (A, F)$ such that A is a nonempty set and F is a family of finitary operations on A. The set A is called the* universe *of $\mathbb{A}$, the operations from F are called* basic. *An algebra with a finite universe is referred to as a* finite algebra.

Every constraint language on a set A can be assigned an algebra with the universe A.

Definition 4 *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 as well.*

The set of all polymorphisms of a constraint language Γ is denoted $\mathsf{Pol}\ \Gamma$*; and the set of all relations invariant under all operations from a set F is denoted* $\mathsf{Inv}\ F$.

Given a constraint language, Γ, on A, the algebra $(A, \mathsf{Pol}\ \Gamma)$ is called the algebra associated with Γ, *and is denoted $\mathbb{A}_\Gamma$.*

Conversely, for any finite algebra $\mathbb{A} = (A; F)$, there is a constraint language associated with $\mathbb{A}$, the language $\mathsf{Inv}\ F$, and the associated problem class $\mathrm{CSP}(\mathbb{A}) = \mathrm{CSP}(\mathsf{Inv}\ F)$. Notice that $\mathsf{Inv}\ F$ is the largest constraint language Γ such that $\mathbb{A}_\Gamma = \mathbb{A}$, see, e.g. [27, 31]. A connection between the complexity of a constraint language and the associated algebra is provided by the following theorem.

Theorem 2 ([17]) *A constraint language Γ on a finite set is tractable [NP-complete] if and only if $\mathsf{Inv}\ \mathsf{Pol}\ (\Gamma)$ is tractable [NP-complete].*

Informally speaking, Theorem 2 says that the complexity of Γ is determined by the algebra $\mathbb{A}_\Gamma$. We, therefore, make the following definition: for a constraint language Γ, the algebra $\mathbb{A}_\Gamma$ is said to be *tractable* [*NP-complete*] if Γ is tractable [NP-complete].

In [18, 19], Jeavons and coauthors have identified certain types of algebras which give rise to tractable problem classes.

Definition 5 *Let A be a finite set. An operation f on A is called*

- *a* projection *if there is $i \in \{1, \ldots, n\}$ such that $f(x_1, \ldots, x_n) = x_i$ for any $x_1, \ldots, x_n \in A$;*
- essentially unary *if $f(x_1, \ldots, x_n) = g(x_i)$, for some unary operation g, and any $x_1, \ldots, x_n \in A$;*
- *a* constant *operation if there is $c \in A$ such that $f(x_1, \ldots, x_n) = c$, for any $x_1, \ldots, x_n \in A$;*
- idempotent *if $f(x, \ldots, x) = x$ for any $x \in A$.*
- *a* semilattice *operation*[1]*, if it is binary idempotent and for any $x, y, z \in A$ satisfies the following two conditions:*
  (a) $f(x, f(y, z)) = f(f(x, y), z)$ (Associativity),
  (b) $f(x, y) = f(y, x)$ (Commutativity);
- *a* majority *operation if it is ternary, and $f(x, x, y) = f(x, y, x) = f(y, x, x) = x$, for any $x, y \in A$;*
- affine *if $f(x, y, z) = x - y + z$ where $+, -$ are the operations of an Abelian group.*

For a finite algebra $\mathbb{A}$, an operation from $\mathsf{Pol}\ \mathsf{Inv}\ F$ is said to be a *term* operation[2] of $\mathbb{A}$. If Γ is a constraint language, the term operations of $\mathbb{A}_\Gamma$ are the polymorphisms of Γ.

Proposition 1 ([18, 19]) *If a finite algebra $\mathbb{A}$ has a term operation which is constant, semilattice, affine, or majority, then $\mathbb{A}$ is tractable.*

The 2-element algebras associated with Schaefer's six types of constraint languages have the constant term operation 0 or 1 in cases (1),(2); the semilattice term operation $\vee$ or $\wedge$ in cases (3),(4); the majority term operation $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$ in case (5); and the affine term operation $x - y + z$ in case (6).

An algebra is said to be a *G-set* if every term operation is essentially unary and the corresponding unary operation is a permutation.

Proposition 2 ([18, 19]) *A finite G-set is NP-complete.*

By combining those two results, and the classical result of E.Post [28], the algebraic version of Schaefer's theorem can be derived [8].

Theorem 3 (Schaefer) *A constraint language Γ on a 2-element set is tractable if $\mathbb{A}_\Gamma$ is not a G-set. Otherwise Γ is NP-complete.*

2.3. Algebraic constructions and the complexity of constraint languages

Certain transformations of constraint languages preserve the complexity, and may lead to languages with certain desirable properties. Let Γ be a constraint language on A, and

[1] In some earlier papers [17, 18] the term *ACI operation* is used.

[2] Every term operation can be obtained from operations of F by superposition.

g a unary polymorphism of Γ such that $g(g(x)) = g(x)$. By $g(\Gamma)$ we denote the set $\{g(R) \mid R \in \Gamma\}$ where $g(R) = \{(g(\mathbf{a}[1]), \ldots, g(\mathbf{a}[n])) \mid (\mathbf{a}[1], \ldots, \mathbf{a}[n]) \in R\}$; and by Γ^+ the set $\Gamma \cup \{\{(a)\} \mid a \in A\}$. If $g \in \mathsf{Pol}\ \Gamma$ is a unary operation whose range is minimal among ranges of unary operations from $\mathsf{Pol}\ \Gamma$ with the property $g(g(x)) = g(x)$, then the constraint language $g(\Gamma)^+$ will be denoted Γ_g^{id}.

Proposition 3 ([8]) *Let Γ be a constraint language on A, and $g \in \mathsf{Pol}\ \Gamma$ a unary operation on A with a minimal range and such that $g(g(x)) = g(x)$. Then Γ is tractable [NP-complete] if and only if Γ_g^{id} is tractable [NP-complete].*

If Γ and g satisfy the conditions of Proposition 3 then the algebra $\mathbb{A}_{\Gamma_g^{\mathrm{id}}}$ is *idempotent*, that is, all its basic operations are idempotent. The complexity of the constraint language Γ_g^{id} does not depend on the choice of g, and we shall denote every such language by Γ^{id}.

Due to Theorem 2 and Proposition 3 the study of the complexity of constraint languages is completely reduced to the study of properties of idempotent algebras.

Definition 6 *Let $\mathbb{A} = (A, F)$ be an algebra, and B a subset of A such that, for any $f \in F$ (n-ary), and for any $b_1, \ldots, b_n \in B$, we have $f(b_1, \ldots, b_n) \in B$. Then the algebra $\mathbb{B} = (B, F\big|_B)$, where $F\big|_B$ consists of restrictions of operations from F onto B, is called a* subalgebra *of $\mathbb{A}$. The universe of a subalgebra of $\mathbb{A}$ is called a* subuniverse *of $\mathbb{A}$.*

An equivalence relation $\theta \in \mathsf{Inv}\ F$ is said to be a congruence *of $\mathbb{A}$. The θ-class containing $a \in A$ is denoted a^θ, the set $A/_\theta = \{a^\theta \mid a \in A\}$ is said to be the* factor-set, *and the algebra $\mathbb{A}/_\theta = (A/_\theta; F^\theta)$, $F^\theta = \{f^\theta \mid f \in F\}$ where $f^\theta(a_1^\theta, \ldots, a_n^\theta) = (f(a_1, \ldots, a_n))^\theta$, is said to be the* factor-algebra.

Proposition 4 ([8]) *Let Γ be a tractable constraint language on A, $B \subseteq A$ a subuniverse of $\mathbb{A}_\Gamma$, and θ an equivalence relation invariant under $\mathsf{Pol}\ \Gamma$. Then*
(1) the subalgebra $\mathbb{B} = (B; (\mathsf{Pol}\ \Gamma)\big|_B)$ of $\mathbb{A}_\Gamma$ is tractable;
(2) $\mathbb{A}_\Gamma/_\theta$, and the set $\Gamma^\theta = \{R^\theta \mid R \in \Gamma\}$ where $R^\theta = \{(\mathbf{a}[1]^\theta, \ldots, \mathbf{a}[n]^\theta) \mid (\mathbf{a}[1], \ldots, \mathbf{a}[n]) \in R\}$, are tractable.

Hence, every subalgebra and every factor-algebra of a tractable algebra are tractable. Furthermore, every *factor* of a tractable algebra, that is, a factor-algebra of a subalgebra, is tractable and cannot be a G-set. Thus, every tractable algebra $\mathbb{A}$ satisfies the condition

$$\text{none of the factors of } \mathbb{A} \text{ is a } G\text{-set.} \qquad (\text{NO-G-SET})$$

Moreover, all known examples of NP-complete subclasses of the CSP have a G-set associated in some way. We, therefore, make the following conjecture.

Conjecture 1 *A constraint language Γ on a finite set A is tractable if $\mathbb{A}_{\Gamma^{\mathrm{id}}}$ satisfies (*NO-G-SET*). Otherwise it is NP-complete.*

By Theorem 3, the conjecture holds for constraint languages on a 2-element set. The main result of this paper is that the conjecture holds for languages on a 3-element set.

Theorem 4 *A constraint language Γ on a 3-element set is tractable if and only if the algebra $\mathbb{A}_{\Gamma^{\mathrm{id}}}$ satisfies* (NO-G-SET)*. Otherwise Γ is NP-complete.*

3. Tractable constraint languages on a 3-element set

The necessity of the condition (NO-G-SET) for the tractability of a finite algebra follows from Propositions 2,4. To show that the condition is sufficient, we have to exhibit, for any idempotent 3-element algebra $\mathbb{A}$ satisfying (NO-G-SET), an algorithm that solves the problem CSP$(\mathbb{A})$ in polynomial time. We split the proof into two parts. The first part is 'algebraic'; we show that, for any idempotent 3-element algebra $\mathbb{A} = (A; F)$ satisfying (NO-G-SET), the relations from $\mathsf{Inv}\ F$ satisfy one of 10 properties. The second part is 'algorithmic'; for each of those 10 properties, we exhibit a polynomial time algorithm that solves the constraint satisfaction problem over relations satisfying this property. The properties of constraint languages ensuring tractability will be defined in this section, the algorithms will be presented in Section 4, while the algebraic part based on an elaborate case analysis can be found in the full version of the paper [3].

Throughout the rest of this section $\mathbb{A} = (A; F)$ is a 3-element algebra satisfying (NO-G-SET). By $\underline{n}$, for a natural number n, we will denote the set $\{1, \ldots, n\}$. Let R be an n-ary relation, and $I = \{i_1, \ldots, i_k\} \subseteq \underline{n}$; then R_I denotes the k-ary relation $\{\mathbf{a}_I \mid \mathbf{a} \in R\}$ where $\mathbf{a}_I = (\mathbf{a}[i_1], \ldots, \mathbf{a}[i_k])$. We will often consider relations whose coordinate positions are indexed by not necessarily natural numbers, but elements of some arbitrary set, for example, the coordinate positions of constraint relations will be supposed to be indexed by variables. In the definition below we use the following notation. For an (n-ary) relation $R \in \mathsf{Inv}\ F$ and a 2-element subuniverse B of $\mathbb{A}$ we set $U = \{i \in \underline{n} \mid R_{\{i\}} = A\}$; $W = \{i \in \underline{n} \mid B \subseteq R_{\{i\}}\}$; we denote by $\theta_B(R)$ the equivalence relation on W generated by the set

$$\{(i, j) \mid \text{for any } \mathbf{a} \in R, \mathbf{a}[i], \mathbf{a}[j] \in B \text{ or } \mathbf{a}[i], \mathbf{a}[j] \notin B\},$$

and $W_1, \ldots, W_k$ the classes of this equivalence relation. Recall that the *graph* of a mapping $f: A \to B$ is the binary relation $\{(a, f(a)) \mid a \in A\}$. The relation R is said to be *irreducible* if, for every pair i, j of coordinate positions, the projection $R_{\{i,j\}}$ is not the graph of a mapping.

Definition 7 *The algebra* $\mathbb{A}$

- *satisfies the* partial zero property *if there exist a set of its subuniverses, Z, and $z_B \in B$ for each $B \in Z$, such that (a) $A \in Z$; (b) for any relation $R \in \mathsf{Inv}\ F$, and any $\mathbf{a} \in R$, there is $\mathbf{b} \in R$ with*
$$\mathbf{b}[i] = \begin{cases} z_B, & \text{if } R_{\{i\}} = B \in Z, \\ \mathbf{a}[i], & \text{otherwise.} \end{cases}$$

- *satisfies the* splitting *property if any (n-ary) relation $R \in \mathsf{Inv}\ F$ can be represented in the form $R_U \times R_{\underline{n}-U}$, and $R_U = A^{|U|}$.*

- *satisfies the* $(a-b)$-replacement property, *$a \in A - B$, and $b \in B$, if, for any (n-ary) $R \in \mathsf{Inv}\ F$, and any $\mathbf{a} \in R$, there is $\mathbf{b} \in R$ with*
$$\mathbf{b}[i] = \begin{cases} b, & \text{if } \mathbf{a}[i] = a \text{ and } R_{\{i\}} = A, \\ \mathbf{a}[i], & \text{otherwise.} \end{cases}$$

- *is called* B-rectangular, *if for any relation $R \in \mathsf{Inv}\ F$,*
$$R_W \cap B^{|W|} = (R_{W_1} \cap B^{|W_1|}) \times \ldots \times (R_{W_k} \cap B^{|W_k|});$$
for any $\mathbf{a} \in R$ such that $\mathbf{a}[i] \in B$ whenever $i \in W$, there is $\mathbf{b} \in R$ with
$$\mathbf{b}[i] = \begin{cases} \mathbf{a}[i], & \text{if } i \in W \text{ or } |R_{\{i\}}| = 1; \\ c, & \text{otherwise, with } \{c\} = B \cap R_{\{i\}}. \end{cases}$$

- *is said to be* B-semirectangular *if the equivalence relation η with classes B and $A-B = \{c\}$ is a congruence of $\mathbb{A}$, and, for any (n-ary) relation $R \in \mathsf{Inv}\ F$, any tuple $\mathbf{b} \in R$, and any $\mathbf{a}_j \in R_{W_j} \cap B^{|W_j|}$, $j \in \underline{k}$, R contains the tuple $\mathbf{a}$ with*
$$\mathbf{a}[i] = \begin{cases} \mathbf{b}[i], & \text{if } B \not\subseteq R_{\{i\}}; \\ c, & \text{if } B \subseteq R_{\{i\}} \text{ and } \mathbf{b}[i] = c; \\ \mathbf{a}_j[i], & \text{if } i \in W_j \text{ and } \mathbf{b}[i] \in B; \end{cases}$$

- *satisfies the* B-semisplitting *property if, for any irreducible (n-ary) relation $R \in \mathsf{Inv}\ F$, we have (i) $(R_U \cap B^{|U|}) \times R_{\underline{n}-U} \subseteq R$; and (ii) for any $i, j \in U$ and any $(a_i, a_j) \in R_{\{i,j\}} \cap B^2$, there is a tuple $\mathbf{a} \in R_U \cap B^{|U|}$ such that $\mathbf{a}[i] = a_i, \mathbf{a}[j] = a_j$.*

- *satisfies the* B-extendibility property *if, for any (n-ary) relation $R \in \mathsf{Inv}\ F$,*
  - *for any $k \in W$ [$k, l \in W$], and any $a \in B$ [$(a, b) \in R_{\{k,l\}}$], there is $\mathbf{a} \in R$ with $\mathbf{a}[i] \in B$ for all $i \in W$, and $\mathbf{a}[k] = a$ [$\mathbf{a}[k] = a, \mathbf{a}[l] = b$];*
  - *for any $\mathbf{a} \in B^{|W|}$ such that $(\mathbf{a}[i], \mathbf{a}[j]) \in R_{\{i,j\}}$, for any $i, j \in W$, there is $\mathbf{b} \in R$ such that*
$$\mathbf{b}[i] = \begin{cases} \mathbf{a}[i], & \text{if } i \in W \text{ or } |R_{\{i\}}| = 1, \\ c, & \text{otherwise, with } \{c\} = R_{\{i\}} \cap B. \end{cases}$$

A ternary operation f is said to be *Mal'tsev* if it satisfies the identities $f(x, y, y) = f(y, y, x) = x$. A standard example of a Mal'tsev operation is provided by the operation $x - y + z$ of an Abelian group, or the operation $xy^{-1}z$ of an arbitrary group. A binary operation $f(x, y)$ on the set A is said to be *conservative commutative* if, for any $x, y \in A$, $f(x, y) = f(y, x)$, and $f(x, y) \in \{x, y\}$.

Theorem 5 *If an idempotent 3-element algebra $\mathbb{A} = (A; F)$ satisfies* (NO-G-SET)*, then there is a set F' of its term operations such that the algebra $\mathbb{A}' = (A; F')$ satisfies* (NO-G-SET) *and one of the following conditions holds.*

(1) $\mathbb{A}'$ satisfies the partial zero property.

(2) $\mathbb{A}'$ satisfies the splitting property.

(3) $\mathbb{A}'$ satisfies the $(a-b)$-replacement property for a 2-element subuniverse B, and $a \in A - B$, $b \in B$.

(4) $\mathbb{A}'$ is B-rectangular for a 2-element subuniverse B.

(5) $\mathbb{A}'$ satisfies the B-semirectangular property for a 2-element subuniverse B.

(6) $\mathbb{A}'$ satisfies the B-semisplitting property for a 2-element subuniverse B, and $\mathbb{B}$ has a majority term operation.

(7) $\mathbb{A}'$ satisfies the B-extendibility property for a 2-element subuniverse $B \subseteq A$.

(8) $\mathbb{A}'$ has a majority term operation.

(9) $\mathbb{A}'$ has a conservative commutative term operation.

(10) $\mathbb{A}'$ has a Mal'tsev term operation.

4. Algorithms

In this section we present algorithms that solve the CSP on a 3-element set in the case when it is tractable. It turns out, that we need only three types of algorithms: the first one is based on finding partial solutions, the second one reduces CSP($\mathbb{A}$) to the case of a 2-element domain, and the third one is quite similar to algorithms of linear algebra.

4.1. Partial solutions and bounded width

Definition 8 *Let $\mathcal{P} = (V; A; \mathcal{C})$ be a constraint satisfaction problem, and $W \subseteq V$. The* restricted *problem $\mathcal{P}_W$ is defined to be $(W; A; \mathcal{C}_W)$ where, for each $\langle s, R\rangle \in \mathcal{C}$, there is $\langle s \cap W; R_{s\cap W}\rangle$ in $\mathcal{C}_W$. A solution to $\mathcal{P}_W$ is called a* partial solution*, and the set of all such solutions is denoted $\mathcal{S}_W$.*

The problem $\mathcal{P}$ is said to be k-minimal *if, for any k-element subset $W \subseteq V$, any $\langle s, R\rangle \in \mathcal{C}$, and any $\mathbf{a} \in R$, the tuple $\mathbf{a}_{s\cap W}$ is a part of a solution from $\mathcal{S}_W$.*

Any constraint satisfaction problem instance $\mathcal{P}$ can be modified to obtain a k-minimal problem instance $\mathcal{P}'$ without changing the set of solutions by repeating the following procedure until the instance stays unchanged: solve all restricted problems involving k variables, and then remove from each constraint $\langle s, R\rangle$ all tuples $\mathbf{a} \in R$ such that $\mathbf{a}_{s\cap W}$ is a part of no partial solution for a certain k-element set of variables W. This procedure is called 'establishing k-minimality', and $\mathcal{P}'$ is said to be the *k-minimal instance associated with $\mathcal{P}$*.

Definition 9 *A class $\mathbf{C}$ of constraint satisfaction problems is said to be of* width[3] *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 empty constraint.*

Every class of finite width is tractable, because, assuming k fixed, establishing k-minimality takes polynomial time.

4.2. Multi-sorted constraint satisfaction problems

In [5], an algebraic approach to a generalised version of the constraint satisfaction problem was developed. In this generalised version every variable is allowed to have its own domain. In this paper we need the notion of *multi-sorted* constraint satisfaction problem, and some results from [5] as an auxiliary tool.

Definition 10 *For any collection of sets $\mathcal{A} = \{A_i \mid i \in I\}$, and any list of indices $(i_1, i_2, \ldots, i_m) \in I^m$, a subset R of $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_m}$, together with the list $(i_1, i_2, \ldots, i_m)$, will be called an (m-ary)* relation *over $\mathcal{A}$ with* signature *$(i_1, i_2, \ldots, i_m)$. For any such relation R, the jth component of the signature of R will be denoted $\sigma(j)$.*

Definition 11 *The* multi-sorted constraint satisfaction problem *is the combinatorial decision problem with*

Instance: *a quadruple $(V; \mathcal{A}; \delta; \mathcal{C})$ where V is a set of* variables*; $\mathcal{A} = \{A_i \mid i \in I\}$ is a collection of sets of values [*domains*]; δ is a mapping from V to I, called the* domain function*; $\mathcal{C}$ is a set of* constraints*. Each constraint $C \in \mathcal{C}$ is a pair $\langle s, \varrho\rangle$, where $s = (v_1, \ldots, v_{m_C})$ is a tuple of variables of length m_C, called the* constraint scope*; R is an m_C-ary relation over $\mathcal{A}$ with signature $(\delta(v_1), \ldots, \delta(v_{m_C}))$, called the* constraint relation.

Question: *does there exist a* solution*, i.e. a function φ, from V to $\bigcup_{A\in\mathcal{A}} A$, such that, for each variable $v \in V$, $\varphi(v) \in A_{\delta(v)}$, and for each constraint $\langle s, R\rangle \in \mathcal{C}$, with $s = (v_1, \ldots, v_m)$, the tuple $(\varphi(v_1), \ldots, \varphi(v_m))$ belongs to R?*

It is possible to introduce the algebraic structure of the multi-sorted CSP in a very similar way to the usual one.

[3] Actually, several notions of the width of a problem class appear in the literature. For instance, Feder and Vardi [14] characterised this concept in terms of Datalog programs. In this paper we use the weakest version, which, therefore, gives the widest possible family of problem classes.

Definition 12 *Algebras $\mathbb{A}_1 = (A_1, F_1)$, $\mathbb{A}_2 = (A_2, F_2)$ are said to be* similar *(or of the* same type*) if there exists a set I such that $F_1 = \{f_i^1 \mid i \in I\}$, $F_2 = \{f_i^2 \mid i \in I\}$ and, for all $i \in I$, f_i^1, f_i^2 are of the same arity.*

Thus, a class of similar algebras can be viewed as a collection of sets, and a set of *operation symbols* such that each operation is assigned the *arity* and has an *interpretation* in each set from the collection, that is an operation of the arity assigned. A class of similar algebras naturally arises when we consider the collection of all factors of an algebra.

Let $\mathcal{A}$ be a class of similar algebras. We say that an (n-ary) operation symbol f *preserves* an (m-ary) multi-sorted relation R over the collection of the universes of algebras from $\mathcal{A}$ (or R is *invariant* with respect to f) if, for any $(a_{11}, \ldots, a_{m1}), \ldots, (a_{1n}, \ldots, a_{mn}) \in R$ we have

$$f\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} f^{\mathbb{A}_{\sigma(1)}}(a_{11}, \ldots, a_{1n}) \\ \vdots \\ f^{\mathbb{A}_{\sigma(m)}}(a_{m1}, \ldots, a_{mn}) \end{pmatrix} \in R$$

where σ is the signature of R. The set of all invariants of a set of operation symbols F is denoted by $\mathsf{m\text{-}Inv}\ F$.

Definition 13 *For a given collection of similar finite algebras, $\mathcal{A}$, $\mathrm{CSP}(\mathcal{A})$ is defined to be the decision problem with*

Instance: *An instance, $\mathcal{P} = (V; \mathcal{B}; \delta; \mathcal{C})$, of the multi-sorted constraint satisfaction problem, in which for each variable v, the domain $A_{\delta(v)}$ is the universe of an algebra $\mathbb{A}_{\delta(v)} \in \mathcal{A}$; for each constraint $\langle s, R\rangle$, the relation R is invariant with respect to all operation symbols of $\mathcal{A}$.*

Question: *Does $\mathcal{P}$ have a solution?*

A class $\mathcal{A}$ of similar finite algebras is said to be *tractable* if $\mathsf{m\text{-}Inv}\ F$ is tractable where F denotes the set of operation symbols of $\mathcal{A}$. If, for every algebra from a class, there is a operation symbol which is interpreted in the algebra as one of the operations listed in Proposition 1, then the class is tractable.

Theorem 6 ([5]) *Let $\mathbb{A}_1, \ldots, \mathbb{A}_l, \mathbb{B}_1, \ldots, \mathbb{B}_n$ be similar finite algebras, each $\mathbb{B}_i$ have either a constant term, or a semilattice or near-unanimity term, and let each $\mathbb{A}_1, \ldots, \mathbb{A}_l$ have an affine term whose interpretations on other algebras are idempotent. Then $\{\mathbb{A}_1, \ldots, \mathbb{A}_l, \mathbb{B}_1, \ldots, \mathbb{B}_n\}$ is tractable.*

We say that a problem instance $\mathcal{P} = (V; A; \mathcal{C}) \in \mathrm{CSP}(\mathbb{A})$ where $\mathbb{A}$ is a 3-element algebra is *2-valued* if, for any $v \in V$, $\mathcal{S}_{\{v\}}$ contains at most 2 elements. Such a problem can be treated as a multi-sorted problem over the family of all proper subalgebras of $\mathbb{A}$. If $\mathbb{A}$ satisfies (NO-G-SET), then by Theorem 6 and Schaefer's theorem, $\mathcal{P}$ can be solved in polynomial time.

Corollary 1 *If a 3-element idempotent algebra satisfies* (NO-G-SET) *then any 2-valued problem instance from* $\mathrm{CSP}(\mathbb{A})$ *can be solved in polynomial time.*

Most of the 'good' properties of relations allow us, first, to reduce an arbitrary problem instance to a 2-valued problem instance, and, second, to solve the obtained instance as a multi-sorted problem instance by making use of the algorithms from [5].

4.3. Why 'good' properties are good

4.3.1. Relations invariant with respect to a special operation. In condition (8) of Theorem 5, the tractability of $\mathbb{A}$ follows from Proposition 1. In (9), $\mathrm{CSP}(\mathbb{A})$ is of width 3, as is proved in [4]. The result of [2] states that any finite algebra with a Mal'tsev operation is tractable, and the solution algorithm is very similar to algorithms from linear algebra.

4.3.2. The partial zero property. In this case any problem instance can be reduced to a 2-valued one. Indeed, if $\mathbb{A}$ satisfies the partial zero property, and a 1-minimal problem instance $\mathcal{P} = (V; A; \mathcal{C}) \in \mathrm{CSP}(\mathbb{A})$ has a solution φ, then $\mathcal{P}$ also has the solution ψ such that $\psi(v) = \varphi(v)$ if $\mathcal{S}_v \notin Z$, and $\psi(v) = z_{\mathcal{S}_{\{v\}}}$ otherwise. Thus, to solve $\mathcal{P}$ we assign the value $z_{\mathcal{S}_{\{v\}}}$ to each variable $v \in V$ with $\mathcal{S}_{\{v\}} \in Z$. Since $A \in Z$, the obtained problem instance is 2-valued.

4.3.3. The replacement property. In this case any problem instance can also be reduced to a 2-valued one. If $\mathbb{A}$ satisfies the $(a-b)$-replacement property, and a 1-minimal problem instance $\mathcal{P} = (V; A; \mathcal{C}) \in \mathrm{CSP}(\mathbb{A})$ has a solution φ, then the mapping $\psi: V \to A$ such that $\psi(v) = b$ if $\mathcal{S}_{\{v\}} = A$, $\varphi(v) = a$, and $\psi(v) = \varphi(v)$ otherwise, is a solution to $\mathcal{P}$. We therefore, may reduce $\mathcal{P}$ to a 2-valued problem instance $\mathcal{P}' = (V; A; \mathcal{C}')$ where for each $C = \langle s, R\rangle \in \mathcal{C}$ there is $C' = \langle s, R'\rangle$ such that $\mathbf{a} \in R'$ if and only if $\mathbf{a} \in R$ and $\mathbf{a}[v] \neq a$ whenever $\mathcal{S}_{\{v\}} = A$.

4.3.4. The extendibility property. We prove that in this case $\mathrm{CSP}(\mathbb{A})$ is of width 3. Suppose that $\mathbb{A}$ satisfies the B-extendibility property, and take a 3-minimal problem instance $\mathcal{P} = (V; A; \mathcal{C})$. An easy proof of the following lemma is left to the reader.

Lemma 1 *Let* $W = \{v \in V \mid B \subseteq \mathcal{S}_{\{v\}}\}$. *There is* $\mathbf{a} \in B^{|W|}$ *such that* $(\mathbf{a}[v], \mathbf{a}[w]) \in \mathcal{S}_{\{v,w\}}$, *for any* $v, w \in W$.

Finally, the B-extendibility property of $\mathbb{A}$ implies that the mapping $\varphi: V \to A$ where

$$\varphi(v) = \begin{cases} \mathbf{a}[v], & \text{if } v \in W; \\ c, & \text{if } \{c\} = \mathcal{S}_{\{v\}} \cap B \text{ or } \mathcal{S}_{\{v\}} = \{c\} \end{cases}$$

is a solution to $\mathcal{P}$.

4.3.5. Rectangularity and semirectangularity. Suppose that $\mathbb{A}$ is B-rectangular or B-semirectangular, and $\{c\} = A - B$. We show that any problem instance in this case can be reduced to a 2-valued one. Take a problem instance $\mathcal{P} = (V; A; \mathcal{C}) \in \mathrm{CSP}(\mathbb{A})$. Without loss of generality we may assume that $\mathcal{P}$ is 3-minimal. Let W denote the set of all variables $v \in V$ with $B \subseteq \mathcal{S}_{\{v\}}$. Let $\theta(\mathcal{P})$ be the equivalence relation on W generated by $\bigcup_{\langle s,R\rangle \in \mathcal{C}} \theta_B(R)$. Notice that, since $\mathcal{P}$ is 3-minimal, for any $\langle s, R\rangle \in \mathcal{C}$, any $u, v \in s \cap W$ such that $(u, v) \in \theta(\mathcal{P})$, and any $\mathbf{a} \in R$, either $\mathbf{a}[u], \mathbf{a}[v] \in B$, or $\mathbf{a}[u] = \mathbf{a}[v] = c$. Repeat the following procedure until the obtained problem instance coincides with the previous one.

- **For** each class W' of $\theta(\mathcal{P})$, **solve** the problem $\mathcal{P}'_{W'} = (W'; B; \mathcal{C}')$ where, for each $\langle s, R\rangle \in \mathcal{C}$, we make the constraint $\langle s \cap W', (R_{s\cap W'}) \cap B^{|s\cap W'|}\rangle \in \mathcal{C}'$.
- **If**, for a class W' of $\theta(\mathcal{P})$, the problem instance $\mathcal{P}'_{W'}$ has no solution **then, for** each constraint $\langle s, R\rangle \in \mathcal{C}$, **remove** from R all the tuples $\mathbf{a}$ such that $\mathbf{a}[v] \in B$ for some $v \in s \cap W'$.
- **Replace** the obtained problem instance with the associated 3-minimal problem instance $\mathcal{P}$.
- **Remove** from W those variables v for which $\mathcal{S}_{\{v\}}$ no longer equals A or B.
- **Calculate** the relation $\theta(\mathcal{P})$ for the obtained problem instance and the set W.

Obviously, the obtained problem instance $\mathcal{P}$ has a solution if and only if the original problem instance has.

Suppose first that $\mathbb{A}$ is B-rectangular. Then, if $\mathcal{P}$ has no empty constraint, then there is a solution φ to $\mathcal{P}$ such that $\varphi(v) \in B$ whenever $B \subseteq \mathcal{S}_{\{v\}}$. Indeed, let $W_1, \dots, W_k$ be the classes of $\theta(\mathcal{P})$, and ψ_i a solution to $\mathcal{P}'_{W_i}$, $i \in \underline{k}$. It follows straightforwardly from the B-rectangularity that the mapping $\varphi: V \to A$ where

$$\varphi(v) = \begin{cases} \psi_i(v), & \text{if } v \in W_i; \\ a, & \text{if } \mathcal{S}_{\{v\}} = \{a, c\}, a \in B; \\ b, & \text{if } \mathcal{S}_{\{v\}} = \{b\}, b \in A, \end{cases}$$

is a solution to $\mathcal{P}$.

Now, suppose that $\mathbb{A}$ satisfies the B-semirectangular property. Denote by $\mathcal{P}^\eta$ the *factor-problem*, that is the problem $(V; \{C_0 = A/_\eta\} \cup \{C_i \mid i \in I\}; \delta; \mathcal{C}')$ where

- C_i, $i \in I$ are the subuniverses of $\mathbb{A}$;
- $\delta(v) = i$ if and only if $\mathcal{S}_{\{v\}} = C_i$, and $\delta(v) = 0$ if $\mathcal{S}_{\{v\}} = A$;
- for each $\langle s, R\rangle \in \mathcal{C}$, there is $\langle s, R^\eta\rangle \in \mathcal{S}^\eta$ where $\mathbf{b} \in R^\eta$ if and only if there is $\mathbf{a} \in R$ such that

$$\mathbf{b}[v] = \begin{cases} \mathbf{a}[v], & \text{if } R_v \neq A; \\ (\mathbf{a}[v])^\eta, & \text{if } R_v = A. \end{cases}$$

It is not hard to see that if $\mathcal{P}$ has a solution, then $\mathcal{P}^\eta$ has a solution (see also [4]). Let φ be a solution to $\mathcal{P}^\eta$, and $\psi_1, \ldots, \psi_k$ solutions to $\mathcal{P}_{W_1}, \ldots, \mathcal{P}_{W_k}$. The mapping ψ where

$$\psi[v] = \begin{cases} \varphi(v), & \text{if } B \not\subseteq \mathcal{S}_{\{v\}}; \\ \psi_i(v), & \text{if } B \subseteq \mathcal{S}_{\{v\}}, v \in W_i, \text{and } \varphi(v) \neq c^\eta; \\ c, & \text{if } B \subseteq \mathcal{S}_{\{v\}}, v \in W_i, \text{and } \varphi(v) = c^\eta. \end{cases}$$

is a solution to $\mathcal{P}$. Indeed, take a constraint $\langle s, R\rangle \in \mathcal{C}$. Since for each $i \in \underline{k}$ such that $\varphi(v) = B$, $v \in W_i$, ψ_i is a solution to $\mathcal{P}_{W_i}$, the tuple $(\psi(v))_{v \in s \cap W_i}$ belongs to $R_{v \in s \cap W_i}$. Moreover, φ is a solution to the factor-problem, therefore, there is $\mathbf{b} \in R$ such that $\mathbf{b}[v] = \varphi(v)$ when $v \in s - W$, $\mathbf{b}[v] = c$ when $\varphi(v) = c^\eta$, $\mathbf{b}[v] \in B$ when $\varphi(v) = B$. The semirectangularity of $\mathbb{A}$ implies that $(\psi(v))_{v \in s} \in R$.

Finally, the factor-problem satisfies the conditions of Theorem 6, and therefore, can be solved in polynomial time.

4.3.6. The splitting and semisplitting property. If $\mathbb{A}$ satisfies the splitting property, then for any 1-minimal problem instance $\mathcal{P} = (V; A; \mathcal{C})$, denote U the set $\{v \in V \mid \mathcal{S}_{\{v\}} = A\}$, $U' = V - U$, and notice that $\mathcal{P}_{U'}$ is 2-valued and any solution to $\mathcal{P}_{U'}$ can be arbitrarily extended to a solution to $\mathcal{P}$.

Suppose that B is a 2-element subuniverse of $\mathbb{A}$, there is a term operation f such that $f|_B$ is the majority operation, and $\mathbb{A}$ satisfies the B-semisplitting property. A problem instance is said to be *irreducible* if every constraint relation is irreducible. Every 3-minimal problem instance $\mathcal{P} = (V; A; \mathcal{C})$ can be reduced to an equivalent irreducible problem instance in polynomial time.

Indeed, denote by κ the binary relation on V such that $(u, v) \in \kappa$ if and only if $\mathcal{S}_{\{u,v\}}$ is the graph of a bijective mapping $\pi_{u,v}\colon \mathcal{S}_{\{u\}} \to \mathcal{S}_{\{v\}}$. Since $\mathcal{P}$ is 3-minimal, $\mathcal{S}_{\{u,v\}} \circ \mathcal{S}_{\{v,w\}} \supseteq \mathcal{S}_{\{u,w\}}$, for any $u, v, w \in V$, where $\circ$ denotes the composition of binary relations; hence, κ is an equivalence relation. Choose a representative from each class of κ, and let W be the set of the representatives. Then, for no pair $v, w \in W$ of variables, $\mathcal{S}_{\{v,w\}}$ is the graph of a bijective mapping, and for any $v \in V - W$, there is $v' \in W$ such that $\mathcal{S}_{\{v',v\}}$ is the graph of a bijective mapping. We transform $\mathcal{P}$ in three steps.

- **For** each constraint $\langle s, R\rangle \in \mathcal{C}$ and any $\mathbf{a} \in R$, **replace** $\mathbf{a}$ with $\mathbf{b}$ where $\mathbf{b}[v] = \pi_{v,v'}(\mathbf{a}[v])$, $v \in s$ and $v' \in W$ is the representative of the κ-class containing v.
- **For** each constraint $\langle s, R\rangle \in \mathcal{C}$ and each $v \in s$, **replace** v with v'.
- **Replace** every constraint $\langle s, R\rangle \in \mathcal{C}$ with $\langle s \cap W, R_{s \cap W}\rangle$.

Now, let $\mathcal{P} = (V; A; \mathcal{C}) \in \mathrm{CSP}(\mathbb{A})$ be a 3-minimal irreducible problem instance, and consider the instance $\mathcal{P}' = (V; A; \mathcal{C}')$ where, for each $\langle s, R\rangle \in \mathcal{C}$, there is $\langle s, R'\rangle \in \mathcal{C}'$ with $R' = \{\mathbf{a} \in R \mid \mathbf{a}[v] \in B$ for all $v \in s$ such that $R_{\{v\}} = A\}$. The problem instance $\mathcal{P}'$ is 2-valued, therefore, we just have to show that $\mathcal{P}$ and $\mathcal{P}'$ are equivalent.

Clearly, if $\mathcal{P}'$ has a solution then $\mathcal{P}$ has a solution. Conversely, let $\mathcal{P}$ have a solution, and set $U = \{v \in V \mid \mathcal{S}_{\{v\}} = A\}$, and $U' = V - U$. By condition (i) of the definition of the semisplitting property, $\mathcal{P}'$ has a solution if and only if both $\mathcal{P}'_U$ and $\mathcal{P}'_{U'}$ have. Since $\mathcal{P}'_{U'} = \mathcal{P}_{U'}$, the instance $\mathcal{P}'_{U'}$ has a solution. By condition (ii), for any $v, w \in U$, $\mathcal{S}'_{\{v,w\}} = \mathcal{S}_{\{v,w\}} \cap B^2$ where $\mathcal{S}'_{\{v,w\}}$ denotes the set of partial solutions to $\mathcal{P}'_U$ for $\{v, w\}$. Moreover, since $\mathcal{P}$ is 3-minimal, for any $u \in U$ every such partial solution can be extended to a solution from $\mathcal{S}'_{\{v,w,u\}}$. (The last property is called *strong 2-consistency* [18].) Recall that any relation $R \in \mathsf{Inv}\ F$ such that $R \subseteq B^n$ is invariant with respect to a majority operation m, in particular, all the constraint relations of $\mathcal{P}'_U$ satisfy this condition. By Theorem 3.5 of [18], if $\mathcal{S}_{\{v,w\}} \neq \varnothing$ for any $v, w \in U$ then strong 2-consistency ensures existence of a solution to $\mathcal{P}'_U$.

5. Recognising tractable cases

From a practical perspective, we need a method that allows us to recognise if a given constraint language Γ is tractable. The following problem is, therefore, very tempting.

TRACTABLE-LANGUAGE. Is a given finite constraint language Γ on a finite set tractable?

Schaefer's Dichotomy Theorem [29] does not solve this problem satisfactorily. Indeed, it can be easily verified if a relation satisfies conditions (1) or (2) of Theorem 1, however, the way of recognising if one of conditions (3)–(6) holds is not obvious (see also [21]). Theorem 3, the algebraic version of Schaefer's result, fills this gap: to check the tractability of a Boolean constraint language one just has to check whether all relations from the language are invariant under one of the 6 Boolean operations corresponding to conditions (1)–(6).

In the general case, such a method can hopefully be derived from a description of tractable algebras. For example, in [6], a polynomial time algorithm has been exhibited that checks if a finite algebra, whose basic operations are given explicitely by their operation tables, satisfies (NO-G-SET). Therefore, if Conjecture 1 holds then the tractability of an algebra can be tested in polynomial time. In particular, this algorithm is valid in the case of 3-element algebras.

However, this algorithm does not solve TRACTABLE-LANGUAGE even under the assumption of Conjecture 1, because in this problem we are given a constraint language, not an algebra. Actually, we need to solve the problem

NO-G-SET-LANGUAGE. Given a finite constraint language Γ on a finite set, does the algebra $\mathbb{A}_{\Gamma^{\mathrm{id}}}$ satisfy (NO-G-SET)?

By the results of [6], this problem is NP-complete. However, its restricted version remains tractable.

NO-G-SET-LANGUAGE(k). Given a finite constraint language Γ on a finite set A, $|A| \le k$, does the algebra $\mathbb{A}_{\Gamma^{\mathrm{id}}}$ satisfy (NO-G-SET)?

This means that the tractability of a constraint language on a 3-element set can be tested in polynomial time.

Theorem 7 *There is a polynomial time algorithm that given a constraint language Γ on a 3-element set determines if Γ is tractable.*

An example of such an algorithm is provided by the general algorithm from [6]. That algorithm employs some deep algebraic results and sophisticated constructions. In the particular case of a 3-element domain, we may avoid using hard algebra, and apply a simpler and easier algorithm.

To this end, notice that if a 3-element idempotent algebra $\mathbb{A}$ has a 2-element subuniverse or a nontrivial congruence, and there is a term operation f which is not a projection on the subalgebra or the factor-algebra, then f witnesses that the algebra $\mathbb{A}$ itself is also not a G-set. We, therefore, have two cases to consider.

CASE 1. $\mathbb{A}$ has no 2-element subuniverse, and no proper congruence.
Such an algebra is said to be *strictly simple*. There is a complete description of finite strictly simple algebras [32]. In particular, if a strictly simple algebra satisfies (NO-G-SET) then one of the following operations is its term operation: a majority operation, the affine operation $x - y + z$ of an Abelian group, or the operation

$$t_0(x, y) = \begin{cases} 0, & \text{if } 0 \in \{x, y\}, \\ x, & \text{otherwise} \end{cases}$$

for some element $0 \in A$.

CASE 2. $\mathbb{A}$ has either a 2-element subalgebra, or a proper congruence.
In this case, $\mathbb{A}$ satisfies (NO-G-SET) if and only if every 2-element subalgebra and every proper factor-algebra (which is also 2-element) is not a G-set. In its turn, the latter condition holds if and only if, for any 2-element subuniverse B of $\mathbb{A}$, and any congruence θ, there is a polymorphism f of Γ such that $f\big|_B$ (or f^θ) is one of the 4 Boolean operations: $\wedge$, $\vee$, the majority operation $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$, the affine operation $x - y + z$ (since $\mathbb{A}$ is idempotent, a constant cannot be its term operation).

As is well known [23], the subuniverses and congruences of a k-element algebra are completely determined by its k-ary term operations. Hence, we may restrict ourselves with finding ternary polymorphisms of a constraint language Γ.

In the first 3 steps, the algorithm below constructs the language Γ^{id}.

Algorithm

INPUT: A finite constraint language Γ on a 3-element set A.
OUTPUT: "YES" if Γ is tractable, "NO" otherwise.

- **Find** all the unary operations on A that preserve each relation from Γ.
- **If** there is a unary non-identity operation f such that $f(f(x)) = f(x)$ **then take** one with a minimal range, and **replace** Γ with $f(\Gamma)$, and A with $f(A)$.
- **Add** the relations $\{(a)\}$, $a \in A$, to Γ.
- **Find** the set F of all ternary operations preserving each relation from Γ.
- **Find** the set S of all 2-element subsets from A and the set C of all proper equivalence relations invariant under operations from F.
- **If** $S = C = \varnothing$ **then**
  - **if** F contains either a majority operation, **or** the operation t_a for some $a \in A$, **or** the affine operation $x - y + z$ of an Abelian group, **then output** "YES";
  - **otherwise output** "NO".
- **Otherwise**, for each $B \in S$ (each $\theta \in C$), **do**
  - **check** if there is $f \in F$ such that $f\big|_B$ (f^θ) is one of the Boolean operations $\wedge$, $\vee$, $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$, $x + y + z$;
  - **if** not **then output** "NO".
- **Output** "Yes".

This algorithm is polynomial time, because the hardest step, finding the set F, requires inspecting of all ternary operations on a 3-element set; and, since their number does not depend on Γ, takes cubic time.

Recognising which of the 10 properties a tractable algebra satisfies also can be completed in polynomial time. However, to establish this requires a more detailed study of the set of ternary polymorphisms, see [3].

6 Conclusion

In fact, Theorem 5 implies a stronger result than that claimed in Theorem 4. The difference appears when considering infinite constraint languages satisfying the conditions of Conjecture 1. Theorem 4 claims that, for any finite subset Γ of such a language, there is its own polynomial time algorithm $\mathrm{Alg}(\Gamma)$ solving $\mathrm{CSP}(\Gamma)$, and for different subsets the corresponding algorithms might be quite different.

Theorem 5 yields a uniform polynomial time algorithm that solves any problem from the class associated with the constraint language. Moreover, from the proof of Theorem 5 a general algorithm can be derived, which solves any problem instance $\mathcal{P}$ on a 3-element set provided that $\mathcal{P} \in \mathrm{CSP}(\Gamma)$, for some tractable Γ.

Note that Theorem 4 is proved by a 'brute force' method, that is, by analysing a large number of operations which provide the condition (NO-G-SET). We believe that development of algebraic tools and more subtle usage of results from universal algebra will make it possible to obtain dichotomy results for larger domains, and eventually, for an arbitrary finite domain.

References

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

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

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

[4] 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.

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

[6] A. Bulatov and P. Jeavons. Algebraic structures in combinatorial problems. Technical Report MATH-AL-4-2001, Technische universität Dresden, Dresden, Germany, 2001.

[7] 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.

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

[9] 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.

[10] V. Dalmau. *Computational Complexity of Problems over Generalised Formulas*. PhD thesis, Department LSI of the Universitat Politecnica de Catalunya (UPC), Barcelona., March, 2000.

[11] R. Dechter and A. Dechter. Structure-driven algorithms for truth maintenance. *Artificial Intelligence*, 82(1-2):1–20, 1996.

[12] R. Dechter and I. Meiri. Experimental evaluation of preprocessing algorithms for constraint satisfaction problems. *Artificial Intelligence*, 68:211–241, 1994.

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

[14] 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.

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

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

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

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

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

[20] P. Jeavons, D. Cohen, and J. Pearson. Constraints and universal algebra. *Annals of Mathematics and Artificial Intelligence*, 24:51–67, 1998.

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

[22] V. Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

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

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

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

[26] B. Nadel and J. Lin. Automobile transmission design as a constraint satisfaction problem: Modeling the kinematik level. *Artificial Intelligence for Engineering Design, Anaysis and Manufacturing (AI EDAM)*, 5(3):137–171, 1991.

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

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

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

[30] E. Schwalb and L. Vila. Temporal constraints: a survey. *Constraints*, 3(2-3):129–149, 1998.

[31] A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathematiques Superieures*. Université de Móntreal, 1986.

[32] A. Szendrei. Simple surjective algebras having no proper subalgebras. *Journal of the Australian Mathematical Society (Series A)*, 48:434–454, 1990.

[33] M. Vardi. Constraint satisfaction and database theory: a tutorial. In *Proceedings of 19th ACM Symposium on Priciples of Database Systems (PODS'00)*, 2000.