Classifying the Complexity of Constraints using Finite Algebras

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

Andrei A. Krokhin
Department of Computer Science
University of Warwick
e-mail: Andrei.Krokhin@dcs.warwick.ac.uk

September 19, 2003

Abstract

Many natural combinatorial problems can be expressed as constraint satisfaction problems. This class of problems is known to be **NP**-complete in general, but certain restrictions on the form of the constraints can ensure tractability. Here we show that any set of relations used to specify the forms of constraints allowed can be associated with a finite universal algebra and we explore how the computational complexity of the corresponding constraint satisfaction problem is connected to the properties of this algebra. Hence, we completely translate the problem of classifying the complexity of restricted constraint satisfaction problems into the language of universal algebra.

We introduce a notion of 'tractable algebra', and investigate how the tractability of an algebra relates to the tractability of the smaller algebras which may be derived from it, including its subalgebras and homomorphic images. This allows us to reduce significantly the types of algebras which need to be classified. Using these results we exhibit a common structural property of all known intractable constraint satisfaction problems and suggest for the first time a general criterion separating the tractable and intractable cases of this problem. We also show that, if the decision problem associated with a given collection of constraint types can be solved efficiently, then so can the corresponding search problem. Finally, we classify all finite strictly simple surjective algebras with respect to tractability; this result is a dichotomy theorem which generalises Schaefer's dichotomy for the Generalised Satisfiability problem.

1 Introduction

In a *constraint satisfaction problem* the aim is to find an assignment of values to a given set of variables, subject to *constraints* on the values which can be assigned simultaneously to certain specified subsets of the variables [26, 30]. One common example of such a problem is the standard propositional satisfiability problem [12], where the variables must be assigned Boolean values and the constraints are specified by clauses.

The mathematical framework used to describe constraint satisfaction problems has strong links with many other areas of computer science and mathematics. For example, links with relational database theory [13, 14, **?**], with some notions of logic and group theory [1, 11], and with universal algebra [21] have been investigated. There is a survey of these results in [33].

Throughout the paper we assume that $\mathbf{P} \neq \mathbf{NP}$. The constraint satisfaction problem is known to be **NP**-hard in general [26, 30]. However, certain restrictions on the form of the constraints have been shown to ensure tractability [**?**, 4, 20, 22, 23, 43, 44].

One fundamental open research problem in this area is to characterise exactly the forms of constraint relations which give rise to tractable problem classes. This problem is important from a theoretical perspective, as it helps to clarify the boundary between tractability and intractability in a wide range of combinatorial search problems. It is also important from a practical perspective, as it allows the development of constraint programming languages which exploit the existence of diverse families of tractable constraints to provide more efficient solution techniques [25, 37].

The problem of characterising the tractable cases was completely solved for the important special case of Boolean constraint satisfaction problems by Schaefer in 1978 [39]. Schaefer established that for Boolean constraint satisfaction problems (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. Similar dichotomy results have also been obtained for other combinatorial problems over a Boolean domain which are related to the Boolean constraint satisfaction problem [**?**].

Schaefer [39] raised the question of how the analysis of complexity could be extended to larger domains (that is, sets with more than 2 elements). Some progress has been made with this question recently, and a number of tractable families of constraints have been identified, over both finite and infinite domains. In particular, Feder and Vardi [11] used techniques from logic programming and group theory to identify three broad families of tractable constraints which include all of Schaefer’s six classes. A series of papers by Jeavons and co-authors has shown that all tractable constraint classes over finite domains can be characterised using algebraic properties of relations [18, 19, 20, 22]. For temporal and spatial reasoning problems involving constraints over infinite sets of values, several tractable subclasses have been identified [9, 31, 38], and a dichotomy theorem has been obtained for qualitative temporal reasoning problems over intervals expressed using Allen's Interval Algebra [?].

However, there is still no complete classification for the complexity of constraints over finite domains with more than two elements, and no dichotomy has so far been obtained for arbitrary finite domains.

In our opinion, the main difficulty in addressing this question is the lack of a powerful and convenient language in which the properties of constraint satisfaction problems responsible for complexity can be expressed. Schaefer's Dichotomy Theorem is stated in terms of the syntactic properties of propositional forms, which is certainly not an appropriate language for non-Boolean constraints. A number of different attempts have been made to find such an appropriate language [6, 7, 11, 18, 20]; we believe that the most fruitful of these is the one that uses the algebraic properties of constraints [18, 20].

The first step in the algebraic approach exploits the well-known idea that, given an initial set of constraint relations, there will often be further relations that can be added to the set without changing the complexity of the associated problem class. In fact, it has been shown that it is possible to add all the relations that can be derived from the initial relations using certain simple rules. The larger sets of relations obtained using these rules are known as *relational clones* [35, ?]. Hence the first step in the analysis is to note that it is sufficient to analyse the complexity only for those sets of relations which are relational clones.

The next step in the algebraic approach is to note that relational clones can be characterised by their *polymorphisms*, which are algebraic *operations* on the same underlying set [20, 17]. As well as providing a convenient and concise method for describing large families of relations, the polymorphisms also reflect certain aspects of the structure of the relations that can be used for designing efficient algorithms. This link between relational clones and polymorphisms has already played a key role in identifying many tractable constraint classes and developing appropriate efficient solution algorithms for them [2, ?, 3, ?, 5, 18].

However, as we shall see later on in this paper, working directly with the polymorphisms of a set of relations is sometimes not the most convenient and powerful way to investigate the complexity of the corresponding constraint satisfaction problems. In this paper we take the algebraic approach one step further by linking constraint satisfaction problems with finite universal algebras (see Figure 1). We show that the language of finite algebras provides a number of very powerful new tools for analysing the complexity of constraint problems. In particular, using this language allows us to suggest a simple criterion distinguishing tractable and intractable cases. This criterion makes extensive use of notions related to universal algebras, and is difficult to formulate concisely without using this language. Another advantage of this new translation is that it allows us to make use of the deep structural results developed for classifying the structure of finite algebras [16, 29, 40].

As the first fruit of using this machinery we exhibit a new dichotomy theorem for a class of algebras which properly includes the two-element algebras, and hence provide a generalisation of Schaefer's Dichotomy Theorem.

The paper is organised as follows. In Section 2 we define the class of con-

$\boxed{\text{Structural properties of a finite universal algebra}}$

$\Updownarrow$

$\boxed{\text{Properties of polymorphisms}}$

$\Updownarrow$

$\boxed{\text{Properties of the corresponding relational clone}}$

$\Updownarrow$

$\boxed{\text{Complexity of a restricted constraint satisfaction problem}}$

Figure 1: The translation into algebra

straint satisfaction problems we are considering, where the constraints are chosen from a specified set of relations. Then we define the way in which these sets of relations may be classified according to the complexity of the corresponding constraint satisfaction problems. We show how this question can be translated into an equivalent question about relational clones, and hence further translated into a question about classifying sets of operations. In Section 3 we make the new step from sets of operations to finite algebras, which is the real focus of this paper. Using this final translation we introduce the notion of a tractable algebra. We are then able to restate Schaefer's Dichotomy Theorem in a much shorter and possibly clearer form, as a classification of two-element algebras with respect to tractability. In Section 4 we prove that, in this context, it suffices to consider certain restricted classes of algebras. As a by-product we show that, if the decision problem for a set of constraint types can be solved efficiently, then so can the corresponding search problem. In Section 5 we study how the tractability of a finite algebra relates to the tractability of its smaller derived algebras. In Section 6 we use the results obtained to classify all strictly simple surjective algebras, and hence generalise Schaefer's Dichotomy Theorem to a much larger class of problems over arbitrary finite domains. Finally, in Section 7 we use the new algebraic terminology to state a conjecture about the general structure of tractable algebras, and provide several examples supporting this conjecture.

2 Definitions and earlier results

2.1 Constraint satisfaction problems

The central notion in the study of constraints and constraint satisfaction problems is the notion of a *relation*.

Definition 2.1 For any set A, and any natural number n, the set of all n-tuples of elements of A is denoted by A^n. Any subset of A^n is called an n-ary *relation* on A. The set of all finitary relations on A is denoted by R_A.

A *constraint language* on A is a subset of R_A.

The 'constraint satisfaction problem' was introduced by Montanari in 1974 [30] and has been widely studied [8, 11, 24, 26, 27, 28]. In this paper we study a parameterised version of the standard constraint satisfaction problem, in which the parameter is a constraint language specifying the possible forms of the constraints.

Definition 2.2 For any set A and any constraint language Γ on A, the constraint satisfaction problem $\mathbf{CSP}(\Gamma)$ is the combinatorial decision problem with

Instance: A triple $(V, A, \mathcal{C})$ where

- V is a set of *variables*;
- $\mathcal{C}$ is a set of *constraints*, $\{C_1, \ldots, C_q\}$.
  Each constraint $C_i \in \mathcal{C}$ is a pair $\langle s_i, \rho_i \rangle$, where
  - s_i is a tuple of variables of length m_i, called the *constraint scope*;
  - $\rho_i \in \Gamma$ is an m_i-ary relation on A, called the *constraint relation*.

Question: Does there exist a *solution*, i.e., a function φ, from V to A, such that, for each constraint $\langle s_i, \rho_i \rangle \in \mathcal{C}$, with $s_i = (x_{i_1}, \ldots, x_{i_m})$, the tuple $(\varphi(x_{i_1}), \ldots, \varphi(x_{i_m}))$ belongs to ρ_i?

In this paper we shall only consider cases where the set A, of possible values for the variables, is finite. In such cases the size of a problem instance can be taken to be the length of a string containing all constraint scopes and all tuples of all constraint relations from the instance.

Example 2.3 An instance of the standard propositional 3-SATISFIABILITY problem [12, 32] is specified by giving a formula in propositional logic consisting of a conjunction of clauses, each of which contains at most 3 literals, and asking whether there are values for the variables which make the formula true.

Suppose that $\Phi = F_1 \wedge \cdots \wedge F_n$ is such a formula, where the F_i are clauses. The satisfiability question for Φ can be expressed as the constraint satisfaction problem instance $(V, \{0,1\}, \mathcal{C})$, where V is the set of all variables appearing in the clauses F_i, and $\mathcal{C}$ is the set of constraints $\{\langle s_1, \rho_1 \rangle, \ldots, \langle s_n, \rho_n \rangle\}$, where each constraint $\langle s_k, \rho_k \rangle$ is constructed as follows:

- $s_k = (x_1^k, x_2^k, x_3^k)$ where x_1^k, x_2^k, x_3^k are the variables appearing in clause F_k;
- $\rho_k = \{0,1\}^3 \setminus \{(a_1, a_2, a_3)\}$ where $a_i = 1$ if x_i^k is negated in F_k and $a_i = 0$ otherwise (i.e., ρ_k contains exactly those 3-tuples that make F_k true).

The solutions of this instance are exactly the assignments which make the formula Φ true.

If we define $\Gamma_{\text{3-SAT}}$ to be the constraint language on $\{0,1\}$ consisting of all relations expressible by 3-clauses, then any instance of 3-SATISFIABILITY can be expressed as an instance of $\mathbf{CSP}(\Gamma_{\text{3-SAT}})$ in this way, and vice versa. $\square$

Example 2.4 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 constraint satisfaction problem instance $(V, \{0,1\}, \mathcal{C})$ where

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

where $=_{\{0,1\}}$ denotes the equality relation on the set $\{0,1\}$.

If we define Γ_{GU} to be the constraint language on $\{0,1\}$ containing just the relations $=_{\{0,1\}}$, $\{(0)\}$ and $\{(1)\}$, then any instance of GRAPH UNREACHABILITY can be expressed as an instance of $\mathbf{CSP}(\Gamma_{\mathrm{GU}})$, in this way. □

Example 2.5 An instance of GRAPH q-COLORABILITY consists of a graph G. The question is whether the vertices of G can be labelled with q colours so that adjacent vertices are assigned different colours.

This problem corresponds to the problem $\mathbf{CSP}(\{\neq_A\})$ where A is a q-element set (of colours) and $\neq_A$ is the disequality relation on A, defined by

$$\neq_A \;= \{(a,b) \in A^2 \mid a \neq b\}.$$

□

Many other examples of standard combinatorial problems expressed as constraint satisfaction problems can be found in [21].

2.2 Tractable constraint languages

Throughout this paper we shall say that a problem is *tractable* if there exists a deterministic polynomial-time algorithm solving all instances of that problem. We can use Definition 2.2 to classify constraint languages according to the complexity of the corresponding constraint satisfaction problem.

In order to be able to classify infinite, as well as finite, constraint languages, we define the notion of tractable constraint language in a way that depends on finite subsets only.

Definition 2.6 For any set A, a finite constraint language $\Gamma \subseteq R_A$ is said to be *tractable* if $\mathbf{CSP}(\Gamma)$ is tractable.

An infinite constraint language $\Gamma \subseteq R_A$ is said to be tractable if every finite subset of Γ is tractable.

A constraint language $\Gamma \subseteq R_A$ is said to be **NP**-complete if $\mathbf{CSP}(\Delta)$ is **NP**-complete for some finite $\Delta \subseteq \Gamma$.

It follows from Example 2.5 that the finite constraint language containing just the single relation $\neq_A$ is tractable when $|A| \leq 2$ and **NP**-complete otherwise.

A finite or infinite constraint language Γ with the property that $\mathbf{CSP}(\Gamma)$ is tractable will be called *globally tractable*. Clearly any constraint language that is globally tractable is tractable in the sense of Definition 2.6, but it is not immediately clear whether or not the converse holds for all infinite languages. This is because for a certain infinite constraint language Γ, it may be the case that for each finite subset $\Delta \subseteq \Gamma$ there exists a polynomial-time algorithm $Alg(\Delta)$ solving $\mathbf{CSP}(\Delta)$, and yet there is no *uniform* polynomial-time algorithm solving $\mathbf{CSP}(\Gamma)$. However, we know of no examples where this is the case and we conjecture that any tractable constraint language is also globally tractable.

Example 2.7 Let A be a finite field, and let Γ_{LIN} be the constraint language consisting of all relations over A which consist of all solutions to some linear equation over A. Any relation from Γ_{LIN}, and therefore any instance of $\mathbf{CSP}(\Gamma_{\mathrm{LIN}})$, can be represented by a system of linear equations over A. Indeed, if $\rho \in \Gamma_{\mathrm{LIN}}$, then it is the solution space of the system of linear equations obtained by the following procedure:

Step 1 Pick an arbitrary element $\overline{a}_0 = (a_{01}, \ldots, a_{0n}) \in \rho$, and set $\rho_0 = \{\overline{b} - \overline{a}_0 \mid \overline{b} \in \rho\}$

Step 2 For every member $(a_1, \ldots, a_n)$ of ρ_0, form the equation $a_1x_1 + \cdots + a_nx_n = 0$, and find a basis, $\rho^{\perp}$, of the solution space of the resulting system of equations.

Step 3 For each $(b_1, \ldots, b_n) \in \rho^{\perp}$, output the equation $b_1x_1 + \ldots + b_nx_n = b_0$, where $b_0 = b_1a_{01} + \ldots + b_na_{0n}$.

Since any instance of $\mathbf{CSP}(\Gamma_{\mathrm{LIN}})$ may be solved in polynomial time (e.g., by Gaussian elimination), it follows that Γ_{LIN} is a (globally) tractable constraint language. $\square$

For $A = \{0, 1\}$, the time complexity of $\mathbf{CSP}(\Gamma)$ has been completely classified [39] for each possible choice of Γ. This result immediately leads to the following complete classification for Boolean constraint languages.

Theorem 2.8 (Schaefer [39]) *A Boolean constraint language, Γ, is (globally) tractable if (at least) one of the following six conditions holds:*

1. *Every relation in Γ contains a tuple in which all entries are 0;*
2. *Every relation in Γ contains a tuple in which all entries are 1;*
3. *Every relation in Γ is definable by a formula in conjunctive normal form in which each conjunct has at most one negated variable;*
4. *Every relation in Γ is definable by a formula in conjunctive normal form in which each conjunct has at most one unnegated variable;*
5. *Every relation in Γ is definable by a formula in conjunctive normal form in which each conjunct contains at most 2 literals.*
6. *Every relation in Γ is the set of solutions of a system of linear equations over the finite field GF(2).*

Otherwise it is **NP***-complete.*

In particular, Theorem 2.8 establishes that any Boolean constraint language can be classified as either tractable or **NP**-complete, and hence the result is known as Schaefers' Dichotomy Theorem. The classification problem for larger finite domains is still open [11], and is the central topic of this paper.

Problem 2.9 ("Tractable Relations Problem") Characterise all tractable constraint languages on finite sets.

Example 2.10 One of the first non-Boolean tractable constraint languages to be characterised was the set, Γ_{ZOA}, of '0/1/all' relations described in [4]. The set Γ_{ZOA} contains all relations over some fixed set A of the following forms:

- All unary relations;
- All binary relations of the form $A_1 \times A_2$ for subsets A_1, A_2 of A.
- All binary relations of the form $\{(a, \pi(a)) \mid a \in A_1\}$, for some subset A_1 of A and some permutation π of A.
- All binary relations of the form $\{(a, b) \in A_1 \times A_2 \mid a = a_1 \vee b = a_2\}$ for some subsets A_1, A_2 of A and some elements $a_1 \in A_1, a_2 \in A_2$.

It was shown in [4] that $\mathbf{CSP}(\Gamma_{\mathrm{ZOA}})$ is tractable, and that for any binary relation ρ over A which is *not* in Γ_{ZOA}, $\mathbf{CSP}(\Gamma_{\mathrm{ZOA}} \cup \{\rho\})$ is **NP**-complete. $\square$

2.3 From arbitrary constraint languages to relational clones

To describe the tractable Boolean constraint languages Schaefer used syntactic properties of propositional formulas representing Boolean relations. In the non-Boolean case this method can no longer be used. We therefore need an adequate language, in which it is possible to express the properties of constraint languages which are responsible for the complexity of the corresponding constraint satisfaction problems.

A useful first step in tackling this problem is to consider what additional relations can be added to a constraint language without changing the complexity of the corresponding problem class. This technique has been widely used in the analysis of Boolean constraint satisfaction problems [**?**, 39], and in the analysis of temporal and spatial constraints [9, 31, 38]; it was introduced for the study of constraints over finite domains in [17].

To use this technique we first define a method for deriving new relations from given ones. The method we use involves defining the new relations using certain kinds of logical formulas involving the given relations. To define such formulas we use the standard correspondence between relations and predicates: a relation consists of all tuples of values for which the corresponding predicate holds. (We will use the same symbol for a predicate and its corresponding relation, since the meaning will always be clear from the context.)

Definition 2.11 ([35]) A constraint language $\Gamma \subseteq R_A$ is called a *relational clone* if it contains every relation (predicate) expressible by a first-order formula involving:

- relations (predicates) from $\Gamma \cup \{=_A\}$ (where $=_A$ is the equality relation on the set A);
- conjunction; and
- existential quantification.

First-order formulas involving only conjunction and existential quantification are often called *primitive positive* (pp-) formulas.

For any constraint language, Γ, there is a unique smallest relational clone containing Γ, which is denoted $\langle\Gamma\rangle$, and is called the relational clone *generated by* Γ. The set $\langle\Gamma\rangle$ consists of all relations definable by primitive positive formulas over the relations in Γ together with the equality relation.

There are a number of different, but equivalent, definitions of relational clones [35, **?**], and a different definition was used in [17] to establish the following theorem. We give a proof here that uses Definition 2.11.

Theorem 2.12 ([17]) *For any set of relations* Γ, *and any finite set* $\Delta \subseteq \langle\Gamma\rangle$, *there is a polynomial time reduction from* $\mathbf{CSP}(\Delta)$ *to* $\mathbf{CSP}(\Gamma)$.

Proof: Let $\Delta = \{\varrho_1, \ldots, \varrho_k\}$ be a finite set of relations over the finite set A, where each ϱ_i is expressible by a pp-formula involving relations from Γ and the equality relation, $=_A$. Note that we may fix these representations.

Any instance $(V; A; \mathcal{C}) \in \mathbf{CSP}(\Delta)$ can be transformed as follows.

- For every constraint $\langle s, \rho\rangle \in \mathcal{C}$ where $s = (v_1, \ldots, v_l)$ and ρ is representable by the pp-formula

  $$\rho(v_1, \ldots, v_l) = \exists u_1, \ldots, u_m \; (\rho_1(w_1^1, \ldots, w_{l_1}^1) \wedge \ldots \wedge \rho_n(w_1^n, \ldots, w_{l_1}^n)),$$

  where $w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^n, \ldots, w_{l_n}^n \in \{v_1, \ldots, v_l, u_1, \ldots, u_m\}$,

  - add the auxiliary variables $u_1, \ldots, u_m$ to V (renaming if necessary so that none of them occurs before);
  - add the constraints $\langle (w_1^1, \ldots, w_{l_1}^1), \rho_1\rangle, \ldots, \langle (w_1^n, \ldots, w_{l_n}^n), \rho_n\rangle$ to $\mathcal{C}$.
  - remove $\langle s, \rho\rangle$ from $\mathcal{C}$;

It can easily be checked that the problem instance obtained by this procedure is equivalent to $(V; A; \mathcal{C})$ and belongs to $\mathbf{CSP}(\Gamma \cup \{=_A\})$. Moreover, since all the representations of relations from Δ are fixed, this transformation can be carried out in linear time in the size of the instance. Finally, all constraints of the form $\langle (v_1, v_2), =_A\rangle$ can be eliminated by replacing all occurrences of the variable v_1 with v_2. This transformation can also be carried out in polynomial time. ∎

Corollary 2.13 *A set of relations, Γ, is tractable if and only if the relational clone $\langle\Gamma\rangle$ is tractable.*

Similarly, Γ is **NP***-complete if and only if $\langle\Gamma\rangle$ is* **NP***-complete.*

This result reduces the problem of characterising tractable constraint languages to the problem of characterising tractable relational clones.

Example 2.14 Reconsider the tractable constraint language of 0/1/all relations, Γ_{ZOA}, defined in Example 2.10.

Note that for any fixed finite set A, the set of 0/1/all relations over A contains only unary and binary relations, and is therefore finite. However, it follows from Corollary 2.13 that the relational clone $\langle\Gamma_{\mathrm{ZOA}}\rangle$ is also tractable. This is an infinite set of relations containing relations of every possible arity.

In fact, the set $\langle\Gamma_{\mathrm{ZOA}}\rangle$ corresponds precisely to the *implicational* relations defined in [23]. Moreover, the set of 0/1/all relations, Γ_{ZOA}, is precisely the set of unary and binary relations in the relational clone $\langle\Gamma_{\mathrm{ZOA}}\rangle$. □

2.4 From relational clones to sets of operations

We have shown in the previous section that to analyse the complexity of arbitrary constraint languages it is sufficient to consider only relational clones. This considerably reduces the variety of languages to be studied. However, this immediately raises the question of how to represent and describe relational clones. For many relational clones the only known generating sets are rather sophisticated, and in some cases no generating sets are known [35].

Very conveniently, it turns out that there is a well-known alternative way to represent and describe any relational clone, using *operations*. In our definitions we follow [29] and [40].

Definition 2.15 For any set A, and any natural number n, a mapping $f : A^n \to A$ is called an *n-ary operation* on A. The set of all finitary operations on A is denoted by O_A.

We first describe a fundamental algebraic relationship between operations and relations. Note that any operation on a set A can be extended in a standard way to an operation on tuples of elements from A, as follows. For any (m-ary) operation f and any collection of tuples $\overline{a}_1, \ldots, \overline{a}_m \in A^n$, where $\overline{a}_i = (a_{1i}, \ldots, a_{ni})$, define $f(\overline{a}_1, \ldots, \overline{a}_m)$ to be $(f(a_{11}, \ldots, a_{1m}), \ldots, f(a_{n1}, \ldots, a_{nm}))$.

Definition 2.16 ([?, 35, 40]) An m-ary operation $f \in O_A$ *preserves* an n-ary relation $\rho \in R_A$ (or f is a *polymorphism* of ρ, or ρ is *invariant* under f) if $f(\overline{a}_1, \ldots, \overline{a}_m) \in \rho$ for all choices of $\overline{a}_1, \ldots, \overline{a}_m \in \rho$.

For any given sets $\Gamma \subseteq R_A$ and $F \subseteq O_A$, let

$$\begin{aligned}\mathsf{Pol}(\Gamma) &= \{f \in O_A \mid f \text{ preserves each relation from } \Gamma\},\\ \mathsf{Inv}(F) &= \{\rho \in R_A \mid \rho \text{ is invariant under each operation from } F\}.\end{aligned}$$

We remark that the operators Pol and Inv form a Galois correspondence between R_D and O_D (see Proposition 1.1.14 of [35]). Introductions to this correspondence can be found in [**?**, 34], and a comprehensive study in [35]. We note, in particular, that $\mathsf{Inv}(F) = \mathsf{Inv}(\mathsf{Pol}(\mathsf{Inv}(F)))$, for any set of operations F. Sets of the form $\mathsf{Inv}(F)$ are precisely the relational clones, as the next result indicates.

Proposition 2.17 ([35]) *For any set A, and any $F \subseteq O_A$, the set $\mathsf{Inv}(F)$ is a relational clone. Conversely, any relational clone can be represented in the form $\mathsf{Inv}(F)$ for some set $F \subseteq O_A$.*

Using Proposition 2.17 together with Corollary 2.13, we can now translate our original problem of characterising tractable sets of relations (Problem 2.9) into an equivalent problem for sets of operations. First, we define what it means for a set of operations to be tractable or **NP**-complete.

Definition 2.18 A set $F \subseteq O_A$ is said to be tractable if $\mathsf{Inv}(F)$ is tractable.
A set $F \subseteq O_A$ is said to be **NP**-complete if $\mathsf{Inv}(F)$ is **NP**-complete.

Using this definition, we obtain the following translation of Problem 2.9.

Problem 2.19 ("Tractable Operations Problem") Characterise all tractable sets of operations on finite sets.

In many cases the description of a set of operations provides a compact, concise, way to describe the associated relational clone, as the next example indicates.

Example 2.20 Recall the tractable constraint language Γ_{ZOA} of 0/1/all relations defined in Example 2.10, and extended to $\langle\Gamma_{\mathrm{ZOA}}\rangle$ in Example 2.14.

It was shown in [20] that $\langle\Gamma_{\mathrm{ZOA}}\rangle$ is precisely the constraint language consisting of all relations which are invariant under the ternary 'dual discriminator' operation d, defined as follows:

$$d(x,y,z) = \begin{cases} y, & \text{if } y = z, \\ x, & \text{otherwise} \end{cases}$$

Hence this infinite tractable constraint language, which is rather complicated to describe in detail, may be represented very simply as the set of relations invariant under the tractable set of operations $\{d\}$. □

In many cases, it has been shown that the presence of a single operation satisfying certain simple conditions is sufficient to ensure the tractability of a set of operations.

Example 2.21 A binary operation $f(x, y)$ satisfying the following three conditions is said to be a semilattice operation[1]:

- $f(x, f(y, z)) = f(f(x, y), z)$ (Associativity)

[1] Note that in some earlier papers [17, 20, 33] the term *ACI operation* is used.

- $f(x, y) = f(y, x)$ (Commutativity)
- $f(x, x) = x$ (Idempotency)

Theorem 16 of [19] says that for any finite set A, any set of operations $F \subseteq O_A$ containing a semilattice operation is tractable. □

In contrast, we will now consider the properties of operations that are associated with **NP**-complete constraint languages.

Definition 2.22 A operation $f : A^n \rightarrow A$ is called *essentially unary* if there exists a non-constant unary operation $g : A \rightarrow A$ and an index $i \in \{1, 2, \ldots, n\}$ such that $f(a_1, a_2, \ldots, a_n) = g(a_i)$ for all choices of $a_1, a_2, \ldots, a_n$. If g is the identity operation, then f is called a *projection.*

Any operation which is not essentially unary (including all constant operations) will be called *essentially non-unary.*

Proposition 2.23 (Jeavons [17]) *For any finite set A, and any $F \subseteq O_A$, if F contains essentially unary operations only then it is* **NP***-complete.*

Example 2.24 Consider the relation, N, over the set $\{0, 1\}$, defined by

$$N = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

It can be shown that $\mathsf{Pol}(\{N\})$ contains essentially unary operations only [36], and hence $\mathbf{CSP}(\{N\})$ is **NP**-complete, by Proposition 2.23.

We remark that the problem $\mathbf{CSP}(\{N\})$ was first shown to be **NP**-complete by Schaefer [39]; it corresponds to a restricted form of the NOT-ALL-EQUAL SATISFIABILITY problem [12, 32]. □

Boolean operations, that is operations on $A = \{0, 1\}$, are very well studied [36, 40]. In particular, it is known that if a Boolean constraint language is not contained in one of Schaefer's six tractable classes, then all of its polymorphisms are essentially unary operations [40]. Hence we may reformulate Schaefer's Dichotomy Theorem to obtain the following complete classification for Boolean operations.

Corollary 2.25 (Schaefer's Dichotomy for Operations) *A set of Boolean operations is tractable if it contains an essentially non-unary operation. Otherwise it is* **NP***-complete.*

3 Algebras

We have shown in the previous section that the problem of analysing the complexity of a constraint language can be translated into the problem of analysing the complexity of the set of operations which preserve all of the relations in that language. In the Boolean case, this is sufficient to obtain a complete classification of complexity, but over larger domains we need to develop more powerful analytical tools, as the next example indicates.

Example 3.1 Consider the binary operation $\circ$ on the set $\{0, 1, 2\}$ defined by the following Cayley table

$\circ$	0	1	2
0	0	1	1
1	1	1	0
2	2	2	2

This operation does not fall into any known tractable class, and nor is it essentially unary. Hence we cannot determine the complexity of this operation using the tools of the previous section (but see Example 5.5 below). □

In this section we shall open up the way to the use of a further set of powerful analytical tools by making the final translation step, from sets of operations to *algebras*.

Definition 3.2 An *algebra* is an ordered pair $\mathcal{A} = (A, F)$ 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 $\mathcal{A}$, and the operations from F are called *basic*. An algebra with a finite universe is referred to as a *finite algebra*.

To make the translation from sets of operations to algebras we simply note that any set of operations F on a fixed set A can be associated with the algebra (A, F). Hence, we will define what it means for an algebra to be tractable by considering the tractability of the basic operations.

Definition 3.3 An algebra $\mathcal{A} = (A, F)$ is said to be *tractable* if F is tractable.
An algebra $\mathcal{A} = (A, F)$ is said to be **NP**-*complete* if F is **NP**-complete.

Using Definition 3.3 we can now translate our original Tractable Relations Problem (Problem 2.9) into the following equivalent problem for algebras:

Problem 3.4 ("Tractable Algebras Problem") Characterise all tractable algebras.

Example 3.5 A *group* is an algebra with three basic operations: a binary multiplication operation, a unary converse operation and a constant unit operation. It follows from Theorem 33 of [11] that any finite group is tractable. □

Using Definition 3.3, we may reformulate Schaefer's Dichotomy Theorem [39] in yet another way, this time as a classification of two-element algebras with respect to tractability.

Corollary 3.6 (Schaefer's Dichotomy for Algebras) *A 2-element algebra $\mathcal{A}$ is* **NP**-*complete if all its basic operations are essentially unary. Otherwise $\mathcal{A}$ is tractable.*

The first advantage of using algebras instead of sets of operations is that we can make use of some standard constructions on algebras to obtain new results about the complexity of constraint languages. Another advantage is that finite algebras have been extensively studied, and a considerable body of structural theory has been developed [16, 29, 40]. We explore these ideas further in the remainder of the paper.

In our study it will be useful to describe an equivalence relation linking algebras that correspond to the same constraint language. As we noted earlier, the mappings Pol and Inv have the property that $\mathsf{Inv}(\mathsf{Pol}(\mathsf{Inv}(F))) = \mathsf{Inv}(F)$, and so we can extend a set of operations F to the set $\mathsf{Pol}(\mathsf{Inv}(F))$ without changing the associated invariant relations. The set $\mathsf{Pol}(\mathsf{Inv}(F))$ consists of all operations that can be obtained from the operations in F, together with the set of all projection operations, by forming arbitrary compositions of operations [**?**, 35]. The set of operations obtained in this way is usually referred to in universal algebra as the set of *term operations* over F [**?**], so we will make the following definition.

Definition 3.7 For any algebra $\mathcal{A} = (A, F)$, an operation f on A will be called a *term operation* of $\mathcal{A}$ if $f \in \mathsf{Pol}(\mathsf{Inv}(F))$.

The set of all term operations of $\mathcal{A}$ will be denoted $\mathsf{Term}(\mathcal{A})$.

Two algebras with the same universe are called *term equivalent* if they have the same set of term operations. Note that, for any algebra $\mathcal{A} = (A, F)$, $\mathsf{Inv}(F) = \mathsf{Inv}(\mathsf{Term}(\mathcal{A}))$, so two algebras are term equivalent if and only if they have the same set of associated invariant relations. It follows that we need to characterise tractable algebras only up to term equivalence.

4 Special classes of algebras

In this section we will show that, when studying the tractability of *finite* algebras we can restrict our attention to certain special classes of algebras.

Definition 4.1 We call an algebra surjective if all of its term operations are surjective[2].

It is easy to verify that a finite algebra is surjective if and only if its unary term operations are all surjective, and hence form a group of permutations.

It was shown in [17] that any unary polymorphism can be applied to a set of relations without changing the complexity of that set.

[2]Note that in [41] an algebra is said to be surjective if all of its basic operations are surjective. Such algebras can have non-surjective term operations.

Proposition 4.2 ([17]) *For any set of relations* Γ*, and any unary operation* $f \in \mathsf{Pol}(\Gamma)$*, let* $f(\Gamma)$ *be the set of relations* $\{f(\rho) \mid \rho \in \Gamma\}$*, where* $f(\rho) = \{f(\overline{a}) \mid \overline{a} \in \rho\}$*. The set* Γ *is tractable if and only if* $f(\Gamma)$ *is tractable, and* Γ *is* **NP***-complete if and only if* $f(\Gamma)$ *is* **NP***-complete.*

Any algebra $\mathcal{A} = (A, F)$ which is not surjective will have a unary term operation f which is not surjective, and hence has range U, where U is a proper subset of A. By applying this operation to all of the relations in $\mathsf{Inv}(F)$, as described in Proposition 4.2, we can obtain a set of relations over U without changing the tractability. The algebra corresponding to this new set of relations can be shown to be a *term induced* algebra of $\mathcal{A}$, which is defined as follows.

Definition 4.3 ([42]) Let $\mathcal{A} = (A, F)$ be an algebra, and U a non-empty subset of A. The term induced algebra $\mathcal{A}|_U$ is defined as $(U, \mathsf{Term}(\mathcal{A})|_U)$ where $\mathsf{Term}(\mathcal{A})|_U = \{g|_U : g \in \mathsf{Term}(\mathcal{A})$ and g preserves $U\}$.

By choosing a term operation f with a range U of *minimal* cardinality, we can ensure that the term induced algebra $\mathcal{A}|_U$ is surjective. Hence we have the following theorem.

Theorem 4.4 *For any finite algebra* $\mathcal{A}$*, there exists a set* U *such that*

- *the algebra* $\mathcal{A}|_U$ *is surjective; and*
- *the algebra* $\mathcal{A}$ *is tractable if and only if* $\mathcal{A}|_U$ *is tractable, and is* **NP***-complete if and only if* $\mathcal{A}|_U$ *is* **NP***-complete.*

Theorem 4.4 shows that we can restrict our attention to surjective algebras. The next theorem shows that we need only consider surjective algebras which have the additional property of being *idempotent*.

Definition 4.5 An operation f on A is called *idempotent* if it satisfies $f(x, \ldots, x) = x$ for all $x \in A$.

The *full idempotent reduct* of an algebra $\mathcal{A} = (A, F)$ is the algebra $(A, \mathsf{Term}_{id}(\mathcal{A}))$, where $\mathsf{Term}_{id}(\mathcal{A})$ consists of all idempotent operations from $\mathsf{Term}(\mathcal{A})$.

Note that an operation f on a set A is idempotent if and only if it preserves all the relations in the set $\Gamma_{\mathrm{CON}} = \{\{(a)\} \mid a \in A\}$, consisting of all unary one-element relations on A. Hence, $\mathsf{Inv}(\mathsf{Term}_{id}(\mathcal{A}))$ is the relational clone generated by $\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}}$.

To establish the next theorem we need an auxiliary lemma from [40].

Lemma 4.6 *Let* $\mathcal{A} = (\{a_1, a_2, \ldots, a_k\}, F)$ *be a finite algebra whose unary term operations form a permutation group* G*. Then the relation* $\rho_G \in \mathsf{Inv}(F)$*, where*

$$\rho_G = \{(g(a_1), \ldots, g(a_k)) \mid g \in G\}.$$

Proof: Follows from Proposition 1.3 in [40]. ∎

Theorem 4.7 *A finite surjective algebra $\mathcal{A}$ is tractable if and only if its full idempotent reduct $\mathcal{A}_0$ is tractable. Moreover, $\mathcal{A}$ is* **NP***-complete if and only if $\mathcal{A}_0$ is* **NP***-complete.*

Proof: Let $\mathcal{A} = (A, F)$ be a finite surjective algebra, and let $\mathcal{A}_0$ be the full idempotent reduct of $\mathcal{A}$.

As observed above, $\mathsf{Inv}(\mathsf{Term}(\mathcal{A}_0))$ is the relational clone generated by $\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}}$, where $\Gamma_{\mathrm{CON}} = \{\{(a)\} \mid a \in A\}$. By Corollary 2.13, it follows that $\mathcal{A}_0$ is tractable if and only if $\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}}$ is tractable, and $\mathcal{A}_0$ is **NP**-complete if and only if $\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}}$ is **NP**-complete. In the remainder of the proof we will show that $\mathbf{CSP}(\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}})$ is polynomial-time equivalent to $\mathbf{CSP}(\mathsf{Inv}(F))$.

Clearly, every instance of $\mathbf{CSP}(\mathsf{Inv}(F))$ may be considered as an instance of $\mathbf{CSP}(\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}})$, so there is a constant time reduction from $\mathbf{CSP}(\mathsf{Inv}(F))$ to $\mathbf{CSP}(\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}})$.

For the converse result, let $\mathcal{P} = (V, A, \mathcal{C})$ be an instance of $\mathbf{CSP}(\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}})$ and let $\mathcal{P}'$ be the problem instance $(V', A, \mathcal{C}')$, where $V' = V \cup \{v_a \mid a \in A\}$ (each of the variables v_a is a new variable not in V). To obtain the constraints $\mathcal{C}'$, take the original constraints $\mathcal{C}$ of $\mathcal{P}$, and replace each unary constraint of the form $\langle v, \{a\} \rangle$ in $\mathcal{C}$ with the constraint $\langle (v, v_a), =_A \rangle$, where $=_A$ is the binary equality relation on A. Finally, add the constraint $\langle (v_{a_1}, \ldots, v_{a_k}), \varrho_G \rangle$, where $a_1, a_2, \ldots, a_k$ are the elements of A (in some order) and ϱ_G is the relation defined in Lemma 4.6. Note that $\mathcal{P}'$ is an instance of $\mathbf{CSP}(\mathsf{Inv}(F))$, and this construction can be carried out in polynomial time.

We claim that if the problem $\mathcal{P}'$ has a solution ψ, then it has a solution ϕ such that $\phi(v_a) = a$ for all $a \in A$. To establish this claim, note that there is $f \in G$ such that $\psi(v_a) = f(a)$ for all $a \in A$. Since G is a group, the inverse operation $f^{-1} \in G$, and hence $\phi = f^{-1}\psi$ is a solution to $\mathcal{P}'$ with $f^{-1}\psi(v_a) = a$ for all $a \in A$.

Any solution ϕ satisfying this condition clearly satisfies all the constraints in $\mathcal{C}$, so the restriction of ϕ to V is a solution to $\mathcal{P}$. Conversely, if ψ is any solution to $\mathcal{P}$, then its extension ψ' to V' such that $\psi'(v_a) = a$, for all $a \in A$, is a solution to $\mathcal{P}'$. Hence this construction establishes a polynomial-time reduction from $\mathbf{CSP}(\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}})$ to $\mathbf{CSP}(\mathsf{Inv}(F))$. ∎

Theorem 4.7 can be restated in terms of constraint languages, as follows.

Corollary 4.8 *Let Γ be a constraint language over a finite set A, and let $\Gamma_{\mathrm{CON}} = \{\{(a)\} \mid a \in A\}$ be the set of all unary one-element relations on A.*

If all unary polymorphisms of Γ are permutations, then Γ is tractable if and only if $\Gamma \cup \Gamma_{\mathrm{CON}}$ is tractable, and Γ is **NP***-complete if and only if $\Gamma \cup \Gamma_{\mathrm{CON}}$ is* **NP***-complete.*

Corollary 4.8 has an interesting consequence connecting decision problems and search problems. In this paper we have formulated the constraint satisfaction problem as a decision problem (Definition 2.2), in which the question is to decide whether or not a solution exists. However, the corresponding search problem, in which the question is to *find* a solution, often arises in practice. We will now show that the tractable cases of these two forms of the problem coincide. (Note that the tractable cases of the search problem are those which belong to the complexity class **FP**.)

Corollary 4.9 *A decision problem* $\mathbf{CSP}(\Gamma)$ *is tractable if and only if the corresponding search problem can be solved in polynomial time.*

Proof: Obviously, tractability of the search problem implies tractability of the corresponding decision problem.

For the converse, let Γ be a tractable set of relations over a finite set A. By choosing a unary polymorphism f of Γ, whose image set U is minimal we can obtain a corresponding set of relations $\Gamma' = f(\Gamma)$ over U, such that every unary polymorphism of Γ' is a permutation. By Proposition 4.2, Γ' is also tractable.

Now consider any instance $\mathcal{P} = (V, A, \mathcal{C})$ of $\mathbf{CSP}(\Gamma)$. By the choice of Γ, we can decide in polynomial time in the size of $\mathcal{P}$ whether this instance has a solution. Assume that it has. Then the instance $\mathcal{P}' = (V, U, \mathcal{C}')$, obtained by replacing each constraint relation ρ with the corresponding relation $f(\rho)$, also has a solution. Furthermore, every solution of $\mathcal{P}'$ is also a solution to $\mathcal{P}$.

Since $\mathcal{P}'$ has a solution, it follows that for each $v \in V$ there must be some $a \in A$ for which we can add the constraint $\langle (v), \{(a)\} \rangle$ and still have a solvable instance. Hence, by considering each variable in turn, and each possible value $a \in A$ for that variable, we can add such a constraint to each variable in turn, and hence obtain a solution to $\mathcal{P}'$. To check for solvability for each possible value at each variable requires us to solve an instance of the decision problem $\mathbf{CSP}(\Gamma' \cup \Gamma_{\mathrm{CON}})$ at most $|V| \cdot |U|$ times, and hence can be completed in polynomial time in the size of $\mathcal{P}'$, by Corollary 4.8. ∎

5 Constructions on algebras

The results in this section link the complexity of a finite algebra with the complexity of its *subalgebras* and *homomorphic images* [**?**, 29]. In many cases, we can use these results to reduce the problem of analysing the complexity of an algebra to a similar problem involving an algebra with a smaller universe.

Definition 5.1 Let $\mathcal{A} = (A, F)$ be an algebra and B a subset of A such that, for any (n_i-ary) $f_i \in F$, and for any $b_1, \ldots, b_{n_i} \in B$, we have $f_i(b_1, \ldots, b_{n_i}) \in B$. Then the algebra $\mathcal{B} = (B, F|_B)$, where $F|_B$ consists of restrictions of operations f_i to B, is called a subalgebra of $\mathcal{A}$. If $B \neq A$ then $\mathcal{B}$ is said to be proper.

Theorem 5.2 *Let $\mathcal{A}$ be a finite algebra. Then*

(i) if $\mathcal{A}$ is tractable then so is every subalgebra of $\mathcal{A}$.

(ii) if $\mathcal{A}$ has an **NP***-complete subalgebra then $\mathcal{A}$ is* **NP***-complete itself.*

Proof: Let $\mathcal{B} = (B, F|_B)$ be a subalgebra of $\mathcal{A} = (A, F)$. It is easy to check that $\mathsf{Inv}(F|_B) \subseteq \mathsf{Inv}(F)$. Hence, $\mathbf{CSP}(\mathsf{Inv}(F|_B))$ can be reduced to $\mathbf{CSP}(\mathsf{Inv}(F))$ in constant time.

Now (i) and (ii) follow immediately from the existence of this reduction. ∎

Definition 5.3 Let $\mathcal{A}_1 = (A_1, F_1)$ and $\mathcal{A}_2 = (A_2, F_2)$ be such that $F_1 = \{f_i^1 \mid i \in I\}$ and $F_2 = \{f_i^2 \mid i \in I\}$, where both f_i^1 and f_i^2 are n_i-ary, for all $i \in I$.

A map $\varphi : A_1 \to A_2$ is called a *homomorphism* from $\mathcal{A}_1$ to $\mathcal{A}_2$ if

$$\varphi f_i^1(a_1, \ldots, a_{n_i}) = f_i^2(\varphi(a_1), \ldots, \varphi(a_{n_i}))$$

holds for all $i \in I$ and all $a_1, \ldots, a_{n_i} \in A_1$.

If the map φ is surjective, then $\mathcal{A}_2$ is said to be a *homomorphic image* of $\mathcal{A}_1$.

Theorem 5.4 *If a finite algebra $\mathcal{A}$ is tractable then so is every homomorphic image of $\mathcal{A}$.*

Proof: Let $\mathcal{B} = (B, F_B)$ be a homomorphic image of $\mathcal{A} = (A, F_A)$ and let φ be the corresponding homomorphism. We will show that, for any finite $\Gamma \subseteq \mathsf{Inv}(F_B)$, $\mathbf{CSP}(\Gamma)$ is linear-time reducible to $\mathbf{CSP}(\Gamma')$ for some finite $\Gamma' \subseteq \mathsf{Inv}(F_A)$.

For $\rho \in \mathsf{Inv}(F_B)$, set $\varphi^{-1}(\rho) = \{\overline{a} \mid \varphi(\overline{a}) \in \rho\}$ where φ acts component-wise. It is clear that $\varphi^{-1}(\rho)$ is a relation of the same arity as ρ. It can straightforwardly be checked that $\varphi^{-1}(\rho) \in \mathsf{Inv}(F_A)$. Let $\Gamma' = \{\varphi^{-1}(\rho) \mid \rho \in \Gamma\}$. Then Γ' is a finite subset of $\mathsf{Inv}(F_A)$.

Take an instance $\mathcal{P} = (V, B, \mathcal{C})$ of $\mathbf{CSP}(\Gamma)$ and construct an instance $\mathcal{P}' = (V, A, \mathcal{C}')$ of $\mathbf{CSP}(\Gamma')$ where $\mathcal{C}' = \{\langle s, \varphi^{-1}(\rho)\rangle \mid \langle s, \rho\rangle \in \mathcal{C}\}$.

If ψ is a solution of $\mathcal{P}'$ then $\varphi\psi$ is a solution of $\mathcal{P}$. Conversely, if ξ is a solution of $\mathcal{P}$ then any function $\psi : V \to A$ such that $\varphi\psi(v) = \xi(v)$ for any $v \in V$ is a solution of $\mathcal{P}'$. ∎

We now give two examples to illustrate the use of Theorems 5.2 and 5.4. The examples show that *both* of these results can be useful to establish the complexity of certain algebras by reducing the question to an algebra over a smaller domain.

Example 5.5 Reconsider Example 3.1. Let $\mathcal{A}$ be the idempotent algebra $(\{0, 1, 2\}, \circ)$ where $\circ$ is the binary operation defined by the following Cayley table[3]

$\circ$	0	1	2
0	0	1	1
1	1	1	0
2	2	2	2

[3] Note, we shall write $x \circ y$ instead of $\circ(x, y)$.

By using Theorem 5.4, we will show that $\mathcal{A}$ is **NP**-complete even though all of its proper subalgebras are tractable.

Notice that, as the equalities $0 \circ 2 = 1$, $1 \circ 2 = 0$, $0 \circ 1 = 1 \circ 0 = 1$ show, $\mathcal{A}$ has only one proper subalgebra having more than 1 element, the algebra $\mathcal{B} = (\{0,1\}; \circ|_{\{0,1\}})$. The operation $\circ|_{\{0,1\}}$ is a semilattice operation, which means that $\mathcal{B}$ is tractable, by Theorem 16 of [19] (see Example 2.21).

On the other hand, consider the algebra $\mathcal{C} = (C; *)$ where $C = \{a, b\}$ and $x * y = x$. It is easy to check that the mapping $\varphi : \{0,1,2\} \to C$ such that $\varphi(0) = \varphi(1) = a, \varphi(2) = b$, is a homomorphism from $\mathcal{A}$ onto $\mathcal{C}$. By Corollary 3.6, $\mathcal{C}$ is **NP**-complete. Hence, by Theorem 5.4, $\mathcal{A}$ is **NP**-complete. $\square$

Example 5.6 Consider the idempotent algebra $\mathcal{A} = (\{0,1,2\}, \circ)$ where $\circ$ is the binary operation defined by the following Cayley table

$\circ$	0	1	2
0	0	1	1
1	0	1	1
2	1	1	2

By using Theorem 5.2, we will show that $\mathcal{A}$ is **NP**-complete even though all of its smaller homomorphic images are tractable.

Since one-element algebras are certainly tractable, we need to consider only two-element homomorphic images $\mathcal{B}$ of $\mathcal{A}$. Let $\mathcal{B} = (\{a, b\}, *)$ where $*$ is a binary operation on $\{a, b\}$ and assume that φ is a homomorphism from $\mathcal{A}$ onto $\mathcal{B}$. Then we have $\varphi(x \circ y) = \varphi(x) * \varphi(y)$ for any $x, y \in \{0,1,2\}$.

<u>Case 1.</u> $\varphi(0) = \varphi(1) = a, \varphi(2) = b$.

We have

$$\begin{aligned} a * a &= \varphi(0) * \varphi(0) = \varphi(0 \circ 0) = \varphi(0) = a, \\ a * b &= \varphi(0) * \varphi(2) = \varphi(0 \circ 2) = \varphi(1) = a, \\ b * a &= \varphi(2) * \varphi(0) = \varphi(2 \circ 0) = \varphi(1) = a, \\ b * b &= \varphi(2) * \varphi(2) = \varphi(2 \circ 2) = \varphi(2) = b. \end{aligned}$$

It is easy to see that $*$ is a semilattice operation on $\{a, b\}$. Hence, by Definition 3.3 and Theorem 16 of [19] (see Example 2.21), the algebra $\mathcal{B}$ is tractable.

<u>Case 2.</u> $\varphi(0) \neq \varphi(1)$.

It follows that $\varphi(2) \in \{\varphi(0), \varphi(1)\}$. Suppose that $\varphi(2) = \varphi(0)$. We have

$$\varphi(0) = \varphi(0 \circ 0) = \varphi(0) * \varphi(0) = \varphi(0) * \varphi(2) = \varphi(0 \circ 2) = \varphi(1).$$

If $\varphi(2) = \varphi(1)$ then

$$\varphi(0) = \varphi(1 \circ 0) = \varphi(1) * \varphi(0) = \varphi(2) * \varphi(0) = \varphi(2 \circ 0) = \varphi(1).$$

Hence this second case is impossible, and we have shown that all smaller homomorphic images of $\mathcal{A}$ are tractable.

On the other hand, the algebra $\mathcal{A}$ has a subalgebra $\mathcal{A}' = (\{0,1\}, \circ)$ such that $x \circ y = y$ for all $x, y \in \{0,1\}$. By Corollary 3.6, $\mathcal{A}'$ is **NP**-complete. Hence, by Theorem 5.2, $\mathcal{A}$ is **NP**-complete. $\square$

6 Strictly simple surjective algebras

The results of the previous sections have established that a tractable algebra must have the property that all of its subalgebras and homomorphic images are tractable. Furthermore, we have shown that we need only consider surjective algebras.

Hence, a natural first question is whether we can classify the complexity of all surjective algebras which do not have any smaller (non-trivial) subalgebras or homomorphic images. Such algebras constitute a kind of 'base case for induction'.

Definition 6.1 A finite algebra is called *simple* if all of its smaller homomorphic images are one-element; and *strictly simple*[4] if it is simple and all of its proper subalgebras are one-element.

In this section we obtain a complete classification for all strictly simple surjective algebras[5]. We show that any algebra of this type is either tractable or **NP**-complete, and we give a complete characterisation of the tractable cases.

Note that all two-element surjective algebras are strictly simple, so this dichotomy result properly includes Schaefer's Dichotomy for Algebras over a 2-element domain (Corollary 3.6).

To obtain the result, we make use of the complete description of finite strictly simple surjective algebras obtained by Á. Szendrei [41]. To formulate Á.Szendrei's result we first need to introduce some notation.

Let G be a permutation group acting on a set A. By $\mathcal{R}(G)$ we denote the set of operations on A preserving each relation of the form $\{(a, g(a)) \mid a \in A\}$ where $g \in G$, and $\mathcal{F}(G)$ denotes the set of idempotent members of $\mathcal{R}(G)$.

A permutation group G acting on a set A is called *regular* if, for any $a, b \in A$, there exists $g \in G$ such that $g(a) = b$, and if each nonidentity member of G has no fixed point. G is called *primitive* if the algebra (A, G) is simple.

Let ${}_K\overline{A} = (A; +, K)$ be a finite dimensional vector space over a finite field K, $T(\overline{A})$ the group of translations $\{x + a \mid a \in A\}$, and $\mathsf{End}\ {}_K\overline{A}$ the endomorphism ring of ${}_K\overline{A}$. Then one can consider $\overline{A}$ as a module over $\mathsf{End}\ {}_K\overline{A}$. This module is denoted by ${}_{(\mathsf{End}\ {}_K\overline{A})}\overline{A}$.

Finally, let $\mathcal{F}_k^0$ denote the set of all operations preserving the relation

$$X_k^0 = \{(a_1, \ldots, a_k) \in A^k \mid a_i = 0 \text{ for at least one } i,\ 1 \le i \le k\}$$

where 0 is some fixed element of A, and let $\mathcal{F}_\omega^0 = \bigcap_{k=2}^{\infty} \mathcal{F}_k^0$.

Theorem 6.2 ([41]) *Let $\mathcal{A}$ be a finite strictly simple surjective algebra.*

[4] In some papers appearing before 1990 such algebras are called *plain*.

[5] Note that we are able to classify all strictly simple surjective algebras, and not just the idempotent ones. This gives a stronger result, because the full idempotent reduct of a strictly simple surjective algebra may not inherit the property of being strictly simple.

- *If $\mathcal{A}$ has no one-element subalgebras then $\mathcal{A}$ is term equivalent to one of the following algebras:*

  (a) $(A, \mathcal{R}(G))$ *for a regular permutation group G acting on A;*

  (b) $(A, \mathsf{Term}_{\mathrm{id}}(_{(\mathsf{End}\ _K\overline{A})}\overline{A}) \cup T(\overline{A}))$ *for some vector space* $_K\overline{A} = (A; +, K)$ *over a finite field K;*

  (c) (A, G) *for a primitive permutation group G on A.*

- *If $\mathcal{A}$ has one-element subalgebras, then $\mathcal{A}$ is idempotent and term equivalent to one of the following algebras:*

  (a°) $(A, \mathcal{F}(G))$ *for a permutation group G on A such that every nonidentity member of G has at most one fixed point;*

  (b°) $(A, \mathsf{Term}_{\mathrm{id}}(_{(\mathsf{End}\ _K\overline{A})}\overline{A}))$ *for some vector space $_K\overline{A}$ over a finite field K;*

  (d) $(A, \mathcal{F}(G) \cap \mathcal{F}_k^0)$ *for some k ($2 \leq k \leq \omega$), some element $0 \in A$ and some permutation group G acting on A such that 0 is the unique fixed point of every nonidentity member of G;*

  (e) (A, F) *where $|A| = 2$ and F contains a semilattice operation;*

  (f) a two-element algebra with an empty set of basic operations.

In the following theorem we determine all tractable finite strictly simple surjective algebras by analyzing each of the cases listed in Theorem 6.2.

Theorem 6.3 *A finite strictly simple surjective algebra is* **NP***-complete if all of its basic operations are essentially unary. Otherwise it is tractable.*

Proof: By Proposition 2.23, the algebras in cases (c) and (f) are **NP**-complete.

Let $\mathcal{A} = (A; F)$ be an algebra of type (a) or (a°). We claim that the dual discriminator operation $d(x, y, z)$ is a term operation of $\mathcal{A}$. To establish this claim, it is easy to verify that d preserves every relation of the form $\{(a, g(a)) \mid a \in A\}$ where g is a permutation on A. Since d is an idempotent operation, it belongs to both $\mathcal{R}(G)$ and $\mathcal{F}(G)$. Hence, any relation in $\mathsf{Inv}(F)$ is invariant under d. It follows from Theorem 5.7 of [20] that $\mathbf{CSP}(\mathsf{Inv}(F))$ is tractable (see Example 2.20). Hence, in this case $\mathcal{A}$ is tractable.

If $\mathcal{A}$ is an algebra of type (b) or (b°) then the affine operation $f(x, y, z) = x - y + z$ is a term operation of $\mathcal{A}$. Tractability of $\mathcal{A}$ follows from Theorem 33 of [11] (see also Corollary 5.17 of [20]).

Now let $\mathcal{A}$ be an algebra of type (d) corresponding to some $2 \leq k \leq \omega$. Consider the operation $f(x, y)$ defined as follows:

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

We will now verify that $f \in \mathcal{F}(G) \cap \mathcal{F}_k^0$.

First, we show that f preserves any relation of the form $g^\circ = \{(a, g(a)) \mid a \in A\}$ where $g \in G$. Let $\overline{a} = (a_1, a_2), \overline{b} = (b_1, b_2) \in g^\circ$. If $a_1 = b_1$ then

$a_2 = g(a_1) = g(b_1) = b_2$, and, by definition of f, we have that the pair $(a_1, g(a_1))$ corresponding to $f(\overline{a}, \overline{b})$ belongs to g°. If $a_1 \neq b_1$ then, since g is a permutation, $a_2 \neq b_2$ and the pair $(0,0) = f(\overline{a}, \overline{b})$ also belongs to g° because 0 is a fixed point of g. Hence $f \in \mathcal{F}(G)$.

Next we show that $f \in \mathcal{F}^0_k$. Let $\overline{a} = (a_1, \ldots, a_k), \overline{b} = (b_1, \ldots, b_k) \in X^0_k$, that is, each of $\overline{a}, \overline{b}$ contains 0. If, say $a_i = 0$ then $f(a_i, b_i) = 0$, that is, $f(\overline{a}, \overline{b})$ contains 0, and hence belongs to X^0_k.

We have shown that $f \in \mathcal{F}(G) \cap \mathcal{F}^0_k$, and hence any relation in $\mathsf{Inv}(F)$ is invariant under f. It is easy to check that f is a semilattice operation. By Theorem 16 of [19], this implies that $\mathbf{CSP}(\mathsf{Inv}(F))$ is tractable. Hence, in this case $\mathcal{A}$ is tractable.

In case (e), tractability of $\mathcal{A}$ follows from Corollary 3.6 ∎

7 Towards a general classification

Theorem 6.3 gives a straightforward necessary and sufficient condition for a finite strictly simple surjective algebra to be tractable. In this section we examine what can be said for more general finite algebras which do have (non-trivial) subalgebras and homomorphic images.

Theorems 5.2 and 5.4 establish two separate necessary conditions for any finite algebra to be tractable. We can combine these conditions by using the standard algebraic notion of a *factor*.

Definition 7.1 A homomorphic image of a subalgebra of an algebra $\mathcal{A}$ is called a *factor* of $\mathcal{A}$.

Corollary 7.2 *If $\mathcal{A}$ is a tractable finite algebra then so is every factor of $\mathcal{A}$.*

Theorems 5.2 and 5.4 also establish two separate sufficient conditions for any finite algebra to be **NP**-complete. Using Proposition 2.23, and the notion of a factor, we can obtain a single sufficient condition for NP-completeness.

Corollary 7.3 *A finite algebra $\mathcal{A}$ is* **NP***-complete if it has a factor $\mathcal{B}$ all of whose basic operations are essentially unary.*

The condition described in Corollary 7.3 is not a necessary condition, in general, as the next example shows.

Example 7.4 Consider the algebra $\mathcal{A} = (\{0,1,2,3\}, \{d, f, p\})$, where d is a binary operation and f, p are unary operations, defined by the following tables[6]

d	0	1	2	3
0	0	3	0	3
1	2	1	2	1
2	2	1	2	1
3	0	3	0	3

x	$f(x)$	$p(x)$
0	1	1
1	0	0
2	3	2
3	2	3

[6]This algebra is, in fact, the *matrix square* [29] of $(\{0,1\}; ^-)$ where $^-$ denotes the negation.

Let $\mathcal{B}$ be a subalgebra of $\mathcal{A}$. Then, since $f(0) = 1$, if $\mathcal{B}$ contains 0 it also contains 1. Analogously, if $1 \in \mathcal{B}$ then $0 \in \mathcal{B}$, and if $\mathcal{B}$ contains one of $2, 3$ then it contains the other. Finally, since $d(0, 1) = 3$, $\mathcal{B} \neq \{0, 1\}$. Similarly, $\mathcal{B} \neq \{2, 3\}$. Hence $\mathcal{A}$ has no proper subalgebras.

Now let ϕ be a non-injective homomorphism of $\mathcal{A}$. Suppose that $\phi(0) = \phi(1)$. Then

$$\begin{aligned} \phi(2) &= \phi(d(1,0)) = \phi(d(0,0)) = \phi(0) = \phi(1), \text{ and} \\ \phi(3) &= \phi(d(0,1)) = \phi(d(0,0)) = \phi(0) = \phi(1), \end{aligned}$$

i.e. the image of ϕ is 1-element. If $\phi(0) = \phi(2)$ then

$$\phi(0) = \phi(2) = \phi(p(2)) = \phi(p(0)) = \phi(1),$$

and we get the previous case. In all other cases a similar proof shows that the image of ϕ contains only 1 element.

We have shown that the only factor of $\mathcal{A}$ with at least 2 elements is $\mathcal{A}$ itself, and clearly not all the operations of $\mathcal{A}$ are essentially unary.

However, we will now show that $\mathcal{A}$ is **NP**-complete. (Note that this does not contradict Theorem 6.3 because $\mathcal{A}$ is not surjective). To establish this, note that the relation

$$\rho = \begin{pmatrix} 0\,0\,0\,3\,3\,3\,0\,0\,0\,3\,3\,3\,1\,1\,1\,2\,2\,2\,1\,1\,1\,2\,2\,2\,1\,1\,1\,2\,2\,2\,0\,0\,0\,3\,3\,3 \\ 0\,3\,3\,0\,0\,3\,1\,1\,2\,1\,2\,2\,0\,0\,3\,0\,3\,3\,1\,2\,2\,1\,1\,2\,0\,0\,3\,0\,3\,3\,1\,1\,2\,1\,2\,2 \\ 1\,1\,2\,1\,2\,2\,0\,3\,3\,0\,0\,3\,0\,3\,0\,3\,0\,3\,0\,0\,3\,0\,3\,3\,1\,2\,2\,1\,1\,2\,1\,2\,1\,2\,1\,2 \end{pmatrix}$$

is invariant under the operations d, f and p. However, if we set $h(x) = d(f(x), p(x))$, then $h(\rho) = N$ where N is the relation defined in Example 2.24, and hence $\{h(\rho)\}$ is **NP**-complete. By Proposition 4.2, it follows that $\{\rho\}$ is **NP**-complete, and hence $\mathcal{A}$ is **NP**-complete. $\square$

On the other hand, Theorem 6.3 shows that the condition described in Corollary 7.3 is both necessary and sufficient for a finite strictly simple surjective algebra to be **NP**-complete (assuming that $\mathbf{P} \neq \mathbf{NP}$). Furthermore, all previously known forms of **NP**-complete constraint satisfaction problems (see [33, 39]) can be shown to be **NP**-complete using Corollary 7.3. Hence, we conjecture that the condition described in Corollary 7.3 is both necessary and sufficient for **NP**-completeness for any finite *surjective* algebra. We state this conjecture for idempotent algebras, where the only essentially unary operations are projections.

Conjecture 7.5 ("Tractable Algebras Conjecture") *A finite idempotent algebra $\mathcal{A}$ is* **NP***-complete if it has a factor $\mathcal{B}$ all of whose operations are projections. Otherwise it is tractable.*

As shown in Sections 2–4, the problem of determining the complexity of an arbitrary constraint language can be reduced to an equivalent problem for a certain idempotent algebra associated with the language. Therefore, this conjecture, if true, would solve all the various forms of "Tractability Problem" mentioned above, including the original problem for arbitrary constraint languages, Problem 2.9.

All dichotomy results obtained by now confirm the conjecture. Schaefer's theorem [39], the dichotomy for GRAPH H-COLORING [15] and LIST GRAPH H-COLOURING [10] problem, the dichotomy for constraint problems on a three-element set [2] and for constraint problems admitting all unary relations [**?**] can be stated as particular cases of the criterion from Conjecture 7.5. !

If Conjecture 7.5 is true then it yields an effective procedure to determine whether a constraint language is tractable, as the next result indicates.

Corollary 7.6 *If Conjecture 7.5 is true, then a constraint language* Γ *over a finite set* A *is* **NP***-complete if and only if there is a binary or ternary relation* $\rho \in \langle\Gamma\rangle$ *such that* $\mathbf{CSP}(\{\rho\})$ *is* **NP***-complete.*

Proof: If Γ is **NP**-complete, then the algebra $\mathcal{A} = (A; \mathsf{Pol}(\Gamma))$ has a factor whose term operations are all essentially unary. Let this factor be a homomorphic image $\mathcal{B} = (B, F_B)$ of a subalgebra $\mathcal{A}' = (A'; F')$. If $\mathcal{B}$ is not surjective then we replace it with the surjective algebra $\mathcal{B}|_U$ where $U = f(B)$ for a certain unary term operation of $\mathcal{B}$.

Now if $|U| = 2$, then $\mathsf{Inv}(F_B|_U)$ contains the ternary relation N defined in Example 2.24, since this relation is preserved by all permutations. Similarly, if $|U| \geq 3$, then $\mathsf{Inv}(F_B|_U)$ contains the binary disequality relation $\neq_U$. So let σ denote N or $\neq_U$, whichever is contained in $\mathsf{Inv}(F_B|_U)$. If φ is the homomorphism from $\mathcal{A}'$ onto $\mathcal{B}$, then as was shown in the proof of Theorem 5.4, $\rho = \varphi^{-1}(f^{-1}(\sigma)) \in \mathsf{Inv}(F)$, and therefore $\rho \in \langle\Gamma\rangle$. ∎

The presence of such a relation can be detected by computing all at most $|A|$-ary polymorphisms of Γ [**?**], and therefore the tractability or **NP**-completeness of Γ can be recognized in polynomial time for any fixed finite set A.

Acknowledgements

The authors thank Victor Dalmau for reporting the result concerning idempotent reducts and for some comments simplifying the proof of Theorem 6.3, and David Cohen for drawing our attention to Corollary 4.9.

This research was partially supported by the UK EPSRC, under Grant Reference GR/M12926.

References

[1] W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence*, 35:401–413, 1988.

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

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

[4] M.C. Cooper, D.A. Cohen, and P.G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.

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

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

[7] V. Dalmau and J. Pearson. Set functions and width 1 problems. In *Proceedings 5th International Conference on Constraint Programming, CP'99*, volume 1713 of *Lecture Notes in Computer Science*, pages 159–173. Springer-Verlag, 1999.

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

[9] T. Drakengren and P Jonsson. A complete classification of tractability in Allen's algebra relative to subsets of basic relations. *Artificial Intelligence*, 106:205–219, 1998.

[10] T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19:487–505, 1999.

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

[12] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA., 1979.

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

[14] M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66(1):57–89, 1994.

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

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

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

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

[19] P.G. Jeavons, D.A. Cohen, and M. Gyssens. A unifying framework for tractable constraints. In *Proceedings 1st International Conference on Constraint Programming—CP'95 (Cassis, France, September 1995)*, volume 976 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, 1995.

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

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

[22] P.G. Jeavons and M.C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2):327–339, 1995.

[23] L. Kirousis. Fast parallel constraint satisfaction. *Artificial Intelligence*, 64:147–160, 1993.

[24] P.B. Ladkin and R.D. Maddux. On binary constraint problems. *Journal of the ACM*, 41:435–469, 1994.

[25] D. Lesaint, N. Azarmi, R. Laithwaite, and P. Walker. Engineering dynamic scheduler for Work Manager. *BT Technology Journal*, 16:16–29, 1998.

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

[27] A.K. Mackworth. Constraint satisfaction. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, volume 1, pages 285–293. Wiley Interscience, 1992.

[28] A.K. Mackworth and E.C. Freuder. The complexity of constraint satisfaction revisited. *Artificial Intelligence*, 59:57–62, 1993.

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

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

[31] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42:43–66, 1995.

[32] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[33] J.K. Pearson and P.G. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway, University of London, July 1997.

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

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

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

[37] L. Purvis and P. Jeavons. Constraint tractability theory and its application to the product development process for a constraint-based scheduler. In *Proceedings of 1st International Conference on The Practical Application of Constraint Technologies and Logic Programming - PACLP'99*, pages 63–79. Practical Applications Company, 1999.

[38] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *Artificial Intelligence*, 108:69–123, 1999.

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

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

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

[42] A. Szendrei. Term minimal algebras. *Algebra Universalis*, 32(4):439–477, 1994.

[43] P. van Beek and R. Dechter. On the minimality and decomposability of row-convex constraint networks. *Journal of the ACM*, 42:543–561, 1995.

[44] P. van Hentenryck, Y. Deville, and C-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.