SIAM J. COMPUT.
Vol. 34, No. 3, pp. 720–742

CLASSIFYING THE COMPLEXITY OF CONSTRAINTS USING FINITE ALGEBRAS*

ANDREI BULATOV†, PETER JEAVONS‡, AND ANDREI KROKHIN§

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 allowed forms of constraints 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 our results 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. We then classify all finite strictly simple surjective algebras with respect to tractability, obtaining a dichotomy theorem which generalizes Schaefer's dichotomy for the generalized satisfiability problem. Finally, we suggest a possible general algebraic criterion for distinguishing the tractable and intractable cases of the constraint satisfaction problem.

Key words. constraint satisfaction problem, universal algebra, dichotomy theorem

AMS subject classifications. 08A70, 68Q25, 68T99

DOI. 10.1137/S0097539700376676

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 [39, 43]. One common example of such a problem is the standard propositional satisfiability problem [21], 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 [22, 23, 34], with some notions of logic and group theory [1, 18, 20], and with universal algebra [31] have been investigated. There is an early survey of these results in [46].

Throughout the paper we assume that $\mathbf{P} \neq \mathbf{NP}$, and we call a problem *tractable* only if it belongs to **P**. The constraint satisfaction problem is known to be **NP**-hard in general [39, 43]. However, certain restrictions on the form of the constraints have been shown to ensure tractability [8, 10, 30, 32, 33, 56].

One fundamental open research problem in this area is to characterize exactly the forms of constraint relations that give rise to tractable problem classes. This problem is important from a theoretical perspective, as it helps to clarify the boundary

*Received by the editors August 15, 2000; accepted for publication (in revised form) October 19, 2004; published electronically April 19, 2005. This work was supported by the UK EPSRC, under grants GR/M12926 and GR/R29598.

http://www.siam.org/journals/sicomp/34-3/37667.html

†School of Computing Science, Simon Fraser University, Canada (abulatov@cs.sfu.ca).

‡Computing Laboratory, University of Oxford, Oxford, UK (Peter.Jeavons@comlab.ox.ac.uk).

§Department of Computer Science, University of Durham, Durham, UK (Andrei.Krokhin@durham.ac.uk).

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 that exploit the existence of diverse families of tractable constraints to provide more efficient solution techniques [38, 50].

The problem of characterizing the tractable cases was completely solved for the important special case of Boolean constraint satisfaction problems by Schaefer in 1978 [52]. Schaefer established that for Boolean constraint satisfaction problems (which he called "generalized 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 that are related to the Boolean constraint satisfaction problem [11].

Schaefer [52] raised the question of how the analysis of complexity could be extended to larger sets of possible values (that is, sets with more than two 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 sets. In particular, Feder and Vardi [20] 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 coauthors has shown that any individual tractable constraint class over a finite domain can be characterized using algebraic properties of relations [28, 29, 30, 32]. This approach was used by Bulatov to obtain a complete classification for the complexity of constraints on a three-element set [3]. Finally, we note that for temporal and spatial reasoning problems involving constraints over infinite sets of values, several tractable constraint classes have been identified [17, 44, 51], and a dichotomy theorem has been obtained for qualitative temporal reasoning problems over intervals expressed using Allen's interval algebra [35]. It has also been shown recently that the complexity of certain forms of constraint over infinite sets of values can be analyzed using algebraic properties [2].

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

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 [13, 14, 20, 28, 30]; we believe that the most fruitful of these is the one that uses the algebraic properties of constraints [28, 30].

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* [16, 48]. Hence the first step in the analysis is to note that it is sufficient to analyze 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 characterized by their *polymorphisms*, which are algebraic *operations* on the same

Structural properties of a finite universal algebra

$\Updownarrow$

Properties of polymorphisms

$\Updownarrow$

Properties of the corresponding relational clone

$\Updownarrow$

Complexity of a restricted constraint satisfaction problem

FIG. 1.1. *Translating questions about the complexity of different forms of constraints into questions about the properties of algebras.*

underlying set [27, 30]. 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 [3, 4, 6, 7, 12, 28].

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 1.1). We show that the language of finite algebras provides a number of very powerful new tools for analyzing the complexity of constraint problems. In particular, using this language allows us to suggest a simple criterion for 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 [25, 42, 53].

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

The paper is organized as follows. In section 2 we define the class of constraint 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 can 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. Finally, in section 7, we use the new algebraic terminology to state a conjecture about the general structure of tractable algebras, and provide examples to illustrate and support 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 *over A. The set of all finitary relations over A is denoted by R_A. A* constraint language *over A is a subset of R_A.*

The "constraint satisfaction problem" was introduced by Montanari [43] in 1974 and has been widely studied [15, 20, 36, 39, 40, 41]. In this paper we study a parameterized 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 over 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*, $\{C_1, \dots, C_q\}$.*
  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 over A, called the* constraint relation.

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

In this paper we shall consider only cases where the set A, specifying the 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 [21, 45] is specified by giving a formula in propositional logic consisting of a conjunction of clauses, each of which contains at most three literals, and asking whether there are values for the variables that 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, the values 0 and 1 represent the logical values FALSE and TRUE, and $\mathcal{C}$ is the set of constraints $\{\langle s_1, \rho_1 \rangle, \dots, \langle s_n, \rho_n \rangle\}$, 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 $\Gamma_{3\text{-SAT}}$ to be the constraint language over $\{0,1\}$ consisting of all relations expressible by 3-clauses, then any instance of 3-SATISFIABILITY can be expressed as an instance of $\mathbf{CSP}(\Gamma_{3\text{-SAT}})$ in this way, and vice versa. □

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 be expressed as the constraint satisfaction problem instance defined by $(V, \{0, 1\}, \mathcal{C})$, 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 over the set $\{0, 1\}$.

If we define Γ_{GU} to be the constraint language over $\{0, 1\}$ 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 colors so that adjacent vertices are assigned different colors [21, 45].

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

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

Example 2.6. Given a fixed graph H, an instance of H-COLORABILITY consists of a graph G. The question is whether there is a mapping from the vertices of G to the vertices of H, such that adjacent vertices in G are mapped to adjacent vertices in H [24]. (In the special case when H is the complete graph on q vertices, K_q, the H-COLORABILITY problem reduces to the GRAPH q-COLORABILITY problem described in Example 2.5.)

This problem corresponds to the problem $\mathbf{CSP}(\{E_H\})$, where E_H is the edge relation of H, that is, the binary relation consisting of all adjacent pairs of vertices from H. □

Many other examples of standard combinatorial problems expressed as constraint satisfaction problems can be found in [31]. For alternative formulations of the constraint satisfaction problem, such as the problem of deciding whether there is a homomorphism from one relational structure to another, see [20, 31, 34].

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 a tractable constraint language in a way that depends on finite subsets only.

DEFINITION 2.7. *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 $\Delta \subseteq \Gamma$.*

Example 2.8. It is well known (see [21, 45]) that the GRAPH q-COLORABILITY problem described in Example 2.5 is tractable when $q \leq 2$ and is **NP**-complete otherwise. Hence, 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 is **NP**-complete otherwise. □

Example 2.9. The complexity of the H-COLORABILITY problem for *undirected* graphs H was completely characterized in [24], where it was shown that if an undirected graph H is bipartite or has a loop, then the corresponding H-COLORABILITY problem is tractable; otherwise it is **NP**-complete.

Using this result, it follows from Example 2.6 that a constraint language $\Gamma = \{E\}$, consisting of a single symmetric binary relation E, is tractable if E is the edge relation of a bipartite graph, or a graph with a loop. Otherwise Γ is **NP**-complete. □

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 2.7, 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 2.10. 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 system of linear equations over A. Any relation from Γ_{LIN}, and therefore any instance of $\mathbf{CSP}(\Gamma_{\text{LIN}})$, can be represented by a system of linear equations over A. Indeed, if $\rho \in \Gamma_{\text{LIN}}$, 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 $b_1x_1 + \cdots + b_nx_n = b_0$, where $b_0 = b_1a_{01} + \cdots + b_na_{0n}$.

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

A constraint language over the set $A = \{0, 1\}$ is known as a *Boolean* constraint language. The complexity of $\mathbf{CSP}(\Gamma)$ has been investigated [52] for all Boolean constraint languages Γ, and the following complete classification has been obtained.

THEOREM 2.11 (see Schaefer [52]). *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 two 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 2.11 establishes that any Boolean constraint language can be classified as either tractable or **NP**-complete, and hence this result is known as Schaefer's dichotomy theorem.

A similar dichotomy theorem has been obtained for constraint languages over any set with three elements [3], using some of the algebraic methods described below. The classification problem for languages over larger finite sets is still open [20], and is the central topic of this paper.

PROBLEM 2.12 ("tractable relations problem"). *Characterize all tractable constraint languages over finite sets.*

Example 2.13. One of the first non-Boolean tractable constraint languages to be characterized was the set Γ_{ZOA} of "0/1/all" relations described in [10]. The set Γ_{ZOA} contains all relations over some fixed set A of the following forms:

(i) all unary relations;
(ii) all binary relations of the form $A_1 \times A_2$ for subsets A_1, A_2 of A;
(iii) 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;
(iv) 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 [10] 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. □

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 [11, 52], and in the analysis of temporal and spatial constraints [17, 44, 51]; it was introduced for the study of constraints over arbitrary finite sets in [27].

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 2.14 (see [48]). *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*

(i) *relations (predicates) from $\Gamma \cup \{=_A\}$ (where $=_A$ is the equality relation on the set A);*
(ii) *conjunction; and*
(iii) *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 pp-formulas over the relations in Γ together with the equality relation.

Example 2.15. Consider the Boolean constraint language $\Gamma = \{R_1, R_2\}$, where $R_1 = \{(0,1),(1,0),(1,1)\}$ and $R_2 = \{(0,0),(0,1),(1,0)\}$.

It is straightforward to check that every binary Boolean relation can be expressed by a pp-formula involving R_1 and R_2. For example, the relation $R_3 = \{(0,0),(1,0),(1,1)\}$ can be expressed by the formula $R_3 = \exists y R_1(x,y) \wedge R_2(y,z)$. Hence the relational clone generated by Γ, $\langle\Gamma\rangle$, includes all 16 binary Boolean relations.

In fact it can be shown that $\langle\Gamma\rangle$ consists of precisely those Boolean relations (of any arity) that can be expressed as a conjunction of unary or binary Boolean relations [49, 53]. ☐

There are a number of different but equivalent definitions of relational clones [16, 48], and a different definition was used in [27] to establish the following theorem. We give a proof here that uses Definition 2.14.

THEOREM 2.16 (see [27]). *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 $\Delta = \{\varrho_1, \dots, \varrho_k\}$ 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 $s = (v_1, \dots, v_l)$ and ρ is representable by the pp-formula

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

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

(i) add the auxiliary variables $u_1, \dots, u_m$ to V (renaming if necessary so that none of them occurs before);

(ii) add the constraints $\langle (w_1^1, \dots, w_{l_1}^1), \rho_1\rangle, \dots, \langle (w_1^n, \dots, w_{l_n}^n), \rho_n\rangle$ to $\mathcal{C}$;

(iii) 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 2.17. *A set of relations Γ is tractable if and only if the relational clone $\langle\Gamma\rangle$ is tractable.*

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

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

Example 2.18. Reconsider the tractable constraint language of 0/1/all relations, Γ_{ZOA}, defined in Example 2.13.

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 2.17 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 [33]. 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 analyze 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, it 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 [48].

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 [42] and [53].

DEFINITION 2.19. *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 2.20 (see [16, 48, 53]). *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

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

We remark that the operators Pol and Inv form a Galois correspondence between R_A and O_A (see Proposition 1.1.14 of [48]). Introductions to this correspondence can be found in [16, 47], and a comprehensive study in [48]. 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 2.21 (see [48]). *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 $F \subseteq O_A$. In particular, for any $\Gamma \subseteq R_A$, $\langle \Gamma \rangle = \mathsf{Inv}(\mathsf{Pol}(\Gamma))$.*

Using Proposition 2.21 together with Corollary 2.17, we can now translate our original problem of characterizing tractable sets of relations (Problem 2.12) 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 2.22. *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 2.12.

PROBLEM 2.23 ("tractable operations problem"). *Characterize 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 2.24. Recall the constraint language Γ_{ZOA} of 0/1/all relations which was defined in Example 2.13 and extended to $\langle \Gamma_{\mathrm{ZOA}} \rangle$ in Example 2.18.

It was shown in [30] 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 2.25. A binary operation $f(x,y)$ satisfying the following three conditions is said to be a semilattice operation:[1]

(i) $f(x, f(y,z)) = f(f(x,y),z)$ (associativity),
(ii) $f(x,y) = f(y,x)$ (commutativity),
(iii) $f(x,x) = x$ (idempotency).

Theorem 16 of [29] 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 2.26. *An operation $f : A^n \to A$ is called* essentially unary *if there exists a (nonconstant) unary operation $g : A \to A$ 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 nonunary.

PROPOSITION 2.27 (see Jeavons [27]). *For any finite set A and any $\Gamma \subseteq R_A$, if* $\mathsf{Pol}(\Gamma)$ *contains essentially unary operations only, then* $\mathbf{CSP}(\Gamma)$ *is* **NP***-complete.*

Example 2.28. 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 [49], and hence $\mathbf{CSP}(\{N\})$ is **NP**-complete, by Proposition 2.27.

We remark that the problem $\mathbf{CSP}(\{N\})$ was first shown to be **NP**-complete by Schaefer [52]; it corresponds to a restricted form of the NOT-ALL-EQUAL SATISFIABILITY problem [21, 45]. □

Boolean operations, that is, operations on $A = \{0,1\}$, have been very well studied [49, 53]. 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 [53]. Hence we may reformulate Schaefer's dichotomy theorem to obtain the following complete classification for Boolean operations.

COROLLARY 2.29 (Schaefer's dichotomy for operations). *A set of Boolean operations is tractable if it contains an essentially nonunary operation. Otherwise it is* **NP***-complete.*

[1] Note that in some earlier papers [27, 30, 46] the term *ACI operation* is used.

3. Algebras. We have shown in section 2 that the problem of analyzing the complexity of a constraint language can be translated into the problem of analyzing 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 sets 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, 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). $\square$

In this section we shall open 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 2.12) into the following equivalent problem for algebras.

PROBLEM 3.4 ("tractable algebras problem"). *Characterize all tractable algebras.*

Using Definition 3.3, we can reformulate Schaefer's dichotomy theorem [52] in yet another way, this time as a classification of the complexity of *algebras* defined on a two-element set.

COROLLARY 3.5 (Schaefer's dichotomy for algebras). *An algebra with a two-element universe is* **NP***-complete if all of its basic operations are essentially unary. Otherwise it 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 [25, 42, 53]. 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 [16, 48]. (If f is an n-ary operation on A, and $g_1, g_2, \ldots, g_n$ are k-ary operations on A, then the *composition* of f and $g_1, g_2, \ldots, g_n$ is the k-ary operation h on A defined by $h(a_1, a_2, \ldots, a_k) = f(g_1(a_1, \ldots, a_k), \ldots, g_n(a_1, \ldots, a_k))$.) The set of operations obtained in this way is usually referred to in universal algebra as the set of *term operations* over F [16], so we will make the following definition.

DEFINITION 3.6. *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 $\mathcal{A} = (A, F)$, we have $\mathsf{Inv}(F) = \mathsf{Inv}(\mathsf{Term}(\mathcal{A}))$, 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 characterize tractable algebras only up to term equivalence.

Example 3.7. A *group* is an algebra with three basic operations: a binary multiplication operation, a unary converse operation, and a constant unit operation (see [37]). A *coset* of a group is a relation which is invariant under the ternary term operation $t(x, y, z) = xy^{-1}z$. It is stated in Theorem 33 of [20] that any constraint language consisting of cosets of a finite group is tractable. Hence any finite group is tractable, and moreover, any finite algebra with the ternary term operation t is also tractable. □

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 [27] that any unary polymorphism can be applied to a set of relations without changing the complexity of that set.

PROPOSITION 4.2 (see [27]). *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 (see [55]). *Let $\mathcal{A} = (A, F)$ be an algebra, and let U be a nonempty 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 unary 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*

(i) *the algebra $\mathcal{A}|_U$ is surjective, and*

[2]Note that in [54] an algebra is said to be surjective if all of its basic operations are surjective. However, such algebras can have nonsurjective term operations, so our definition is more restrictive.

(ii) *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 for many purposes we need consider only those surjective algebras with 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 [53].

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* ρ_G, *defined by*

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

belongs to $\mathsf{Inv}(F)$.

Proof. Proposition 1.3 of [53] states that relations of the form ρ_G are preserved by all operations of the algebra $\mathcal{A}$ and hence belong to $\mathsf{Inv}(F)$. $\square$

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 the set $\mathsf{Inv}(F) \cup \Gamma_{\mathrm{CON}}$, where $\Gamma_{\mathrm{CON}} = \{\{(a)\} \mid a \in A\}$. By Corollary 2.17, 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 that 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, by the definition of ρ_G, there is $g \in G$ such that $\psi(v_a) = g(a)$ for all $a \in A$. Since G is a group, the inverse operation $g^{-1} \in G$, which means that it is a term operation of $\mathcal{A}$. This implies that every relation in $\mathsf{Inv}(F)$ is invariant under g^{-1}, so $\phi = g^{-1}\psi$ is also a solution to $\mathcal{P}'$, and has the property that $g^{-1}\psi(v_a) = a$ 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 $\mathbf{CSP}(\mathsf{Inv}(F) \cup \Gamma_{\text{CON}})$ 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 $\Gamma_{\text{CON}} = \{\{(a)\} \mid a \in A\}$ 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 $\Gamma \cup \Gamma_{\text{CON}}$ is tractable, and Γ is **NP***-complete if and only if $\Gamma \cup \Gamma_{\text{CON}}$ 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 $\mathcal{P}'$. Checking for solvability for each possible value at each variable requires us to solve an instance of the decision problem $\mathbf{CSP}(\Gamma' \cup \Gamma_{\text{CON}})$ 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* [9, 16, 42]. In many cases, we can use these results to reduce the problem of analyzing 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 $f \in F$ and for any $b_1, \dots, b_n \in B$, where n is the arity of f, we have $f(b_1, \dots, b_n) \in B$. Then the algebra $\mathcal{B} = (B, F|_B)$ is called a* subalgebra *of $\mathcal{A}$, where $F|_B$ consists of the restrictions of all operations in F to B. If $B \neq A$, then $\mathcal{B}$ is said to be a* proper *subalgebra.*

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

(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.*

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*

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

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

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

THEOREM 5.4. *Let $\mathcal{A}$ be a finite algebra.*

(i) *If $\mathcal{A}$ is tractable, then so is every homomorphic image of $\mathcal{A}$.*

(ii) *If $\mathcal{A}$ has an* **NP**-*complete homomorphic image, then $\mathcal{A}$ is* **NP**-*complete.*

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 componentwise. 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 (independently) to establish the complexity of certain algebras by reducing the question to an algebra over a smaller set.

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

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 one element, the algebra $\mathcal{B} = (\{0, 1\}, \circ|_{\{0,1\}})$. It is easy to check that $\circ|_{\{0,1\}}$ is a semilattice operation on $\{0, 1\}$. Hence, by Definition 3.3 and Theorem 16 of [29] (see Example 2.25), the algebra $\mathcal{B}$ is tractable.

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

On the other hand, consider the algebra $\mathcal{C} = (C, *)$, where $C = \{a, b\}$ and for all $x, y \in \{a, b\}$, $x * y = x$. 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 3.5, $\mathcal{C}$ is **NP**-complete. Hence, by Theorem 5.4, $\mathcal{A}$ is **NP**-complete. □

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 all $x, y \in \{0, 1, 2\}$.

Case 1. $\varphi(0) = \varphi(1) = a$, $\varphi(2) = b$. In this case 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 check that $*$ is a semilattice operation on $\{a, b\}$. Hence, by Definition 3.3 and Theorem 16 of [29] (see Example 2.25), the algebra $\mathcal{B}$ is tractable.

Case 2. $\varphi(0) \neq \varphi(1)$.

It follows that $\varphi(2) \in \{\varphi(0), \varphi(1)\}$. If $\varphi(2) = \varphi(0)$, then

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

Alternatively, 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 3.5, $\mathcal{A}'$ is **NP**-complete. Hence, by Theorem 5.2, $\mathcal{A}$ is **NP**-complete. □

6. Strictly simple surjective algebras. The results of the previous section have established that a tractable algebra must have the property that all of its subalgebras and homomorphic images are tractable. Hence, a natural first question is whether we can classify the complexity of all algebras which do not have any smaller (nontrivial) subalgebras or homomorphic images. Classifying all such algebras can be viewed as a possible "base case for induction" in the pursuit of a general classification.

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.*

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

By Theorem 4.4, it is sufficient to consider only surjective algebras. 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 characterization of the tractable cases. Such algebras include all surjective two-element algebras, as well as many algebras over larger universes, so this dichotomy result includes and generalizes Schaefer's dichotomy for algebras with a two-element universe (see Corollary 3.5 above).

To obtain the result, we make use of the complete description of finite strictly simple surjective algebras obtained by Szendrei [54]. To formulate Szendrei's result, we first need to introduce some further standard algebraic concepts and notation (for a general introduction to these algebraic concepts, see, for example, [37]).

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 $\{(a, g(a)) \mid a \in A\}$, for some $g \in G$. By $\mathcal{R}_{id}(G)$ we denote the set of idempotent operations in $\mathcal{R}(G)$.

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 $\overline{A} = (A, +)$ be a finite Abelian group, and let K be a finite field. The finite dimensional vector space on $\overline{A}$ over K will be denoted ${}_K\overline{A} = (A; +, K)$, the group of translations $\{x + a \mid a \in A\}$ will be denoted $T(\overline{A})$, and the endomorphism ring of ${}_K\overline{A}$ will be denoted $\mathsf{End}\ {}_K\overline{A}$. Note that one can consider $\overline{A}$ as a module over $\mathsf{End}\ {}_K\overline{A}$. This module will be 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 (see [54]). *Let $\mathcal{A}$ be a finite strictly simple surjective algebra.*

- *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°) *$(A, \mathcal{R}_{id}(G))$ 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) *$(A, \mathcal{R}_{id}(G) \cap \mathcal{F}_k^0)$ for some k $(2 \le k \le \omega)$, 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.

[5] Note that the full idempotent reduct of a strictly simple surjective algebra is not always strictly simple. Hence we obtain a slightly stronger result by classifying all strictly simple surjective algebras, rather than just the strictly simple idempotent algebras.

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

Proof. If $\mathcal{A}$ is an algebra of type (c) or (f), then all of its term operations are essentially unary. Hence $\mathsf{Pol}(\mathsf{Inv}(\mathsf{Term}(\mathcal{A})))$ contains essentially unary operations only, so $\mathcal{A}$ is **NP**-complete, by Proposition 2.27.

If $\mathcal{A} = (A, F)$ is an algebra of type (a) or (a°), then we claim that the dual discriminator operation $d(x, y, z)$ defined in Example 2.24 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 $\mathcal{R}_{id}(G)$. Hence, in cases (a) and (a°), every relation in $\mathsf{Inv}(F)$ is invariant under d. It follows from Theorem 5.7 of [30] that $\mathbf{CSP}(\mathsf{Inv}(F))$ is tractable (see Example 2.24). 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 $\mathcal{A}$ then follows from Theorem 33 of [20] (see Example 3.7).

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

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

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 $f(\overline{a}, \overline{b})$ is equal to $(a_1, g(a_1))$ and hence belongs to g°. If $a_1 \neq b_1$, then, since g is a permutation, $a_2 \neq b_2$ and the pair $f(\overline{a}, \overline{b})$ equals $(0, 0)$, which also belongs to g° because 0 is a fixed point of g. Hence $f \in \mathcal{R}_{id}(G)$.

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

We have shown that $f \in \mathcal{R}_{id}(G) \cap \mathcal{F}_k^0$, and hence f is a term operation of $\mathcal{A}$. It is easy to check that f is a semilattice operation. By Theorem 16 of [29], this implies that $\mathbf{CSP}(\mathsf{Inv}(F))$ is tractable (see Example 2.25). Hence, in this case $\mathcal{A}$ is tractable.

Finally, if $\mathcal{A}$ is an algebra of type (e), then tractability again follows from Theorem 16 of [29]. ∎

7. Toward a general classification. Theorem 6.3 gives a straightforward criterion to determine whether a finite strictly simple surjective algebra is tractable or **NP**-complete. In this section we examine what can be said about more general finite algebras.

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* [9].

DEFINITION 7.1. *A homomorphic image of a subalgebra of an algebra $\mathcal{A}$ is called a* factor *of $\mathcal{A}$. A factor whose universe contains only a single element is called a* trivial *factor.*

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 the notion of a factor, we can combine these results, together with Proposition 2.27, to 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 term operations are essentially unary.*

However, the condition described in Corollary 7.3 is not a necessary condition for an arbitrary algebra $\mathcal{A}$ to be **NP**-complete, 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

Since $f(0) = 1$, any subalgebra of $\mathcal{A}$ containing 0 also contains 1. Furthermore, since $f(1) = 0$, any subalgebra containing 1 also contains 0. Similarly, any subalgebra containing one of $2, 3$ also contains the other. Finally, since $d(0,1) = 3$ and $d(2,3) = 1$, it follows that the only subalgebra of $\mathcal{A}$ is $\mathcal{A}$ itself.

Now let ϕ be a homomorphism of $\mathcal{A}$. If $\phi(0) = \phi(1)$, then

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

so ϕ maps all the elements of $\mathcal{A}$ to a single element. Furthermore, if $\phi(0) = \phi(2)$, then

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

and we get the previous case. In all other cases a similar proof shows that if ϕ is not injective, then it maps all the elements of $\mathcal{A}$ to a single element.

Hence we have shown that the only nontrivial factor of $\mathcal{A}$ 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, consider the ternary relation ρ, consisting of 36 tuples, defined as follows (where tuples are written as columns):

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

It is straightforward to verify that this relation 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 2.28, 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. □

On the other hand, it was shown in Theorem 6.3 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 [46, 52]) 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

[6] This algebra is, in fact, the *matrix square* [42] of $(\{0,1\}; {}^-)$, where ${}^-$ denotes the negation.

finite *surjective* algebra. We state this conjecture for the special case of 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 nontrivial 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 the "tractability problem" mentioned above, including the original problem for arbitrary constraint languages, Problem 2.12.

The next examples show that Conjecture 7.5 is confirmed in a number of special cases, by existing dichotomy results. Moreover, we will show that in each case the existing dichotomy result can be obtained as a simple consequence of Conjecture 7.5.

Example 7.6. A dichotomy theorem for two-element algebras was given in Corollary 3.5. In this example we will show that this result is equivalent to Conjecture 7.5 restricted to two-element algebras.

Let $\mathcal{A}$ be a two-element algebra.

If $\mathcal{A}$ is idempotent, then Corollary 3.5 implies that either $\mathcal{A}$ is tractable, or else it is **NP**-complete and all of its operations are projections. Hence Corollary 3.5 implies that Conjecture 7.5 holds for any two-element algebra.

Conversely, we will now establish that Conjecture 7.5 implies Corollary 3.5. If $\mathcal{A}$ is not surjective, then it has a nonsurjective unary term operation, which must be constant, and hence $\mathcal{A}$ is tractable, by Theorem 4.4. On the other hand, if $\mathcal{A}$ is surjective, then by Theorem 4.7 we can consider its full idempotent reduct, $\mathcal{A}_0$. Assuming Conjecture 7.5 holds for any two-element algebra, we have that either $\mathcal{A}_0$ is tractable or else it is **NP**-complete and every operation of $\mathcal{A}_0$ is a projection. If every operation of $\mathcal{A}_0$ is a projection, then we claim that every operation of $\mathcal{A}$ must be essentially unary. To establish this claim, let f be any term operation of $\mathcal{A}$. Since $\mathcal{A}$ is surjective, the unary term operation $g(x) = f(x, x, \ldots, x)$ must be a permutation. Now let h be the composition of the inverse permutation g^{-1} and f. It is easy to check that h is an idempotent term operation of $\mathcal{A}$ and hence is an operation of $\mathcal{A}_0$. If h is a projection, then f must depend on only one of its arguments, which establishes the claim. Hence, we have shown that either $\mathcal{A}$ is tractable, or else every operation of $\mathcal{A}$ is essentially unary, which establishes that Conjecture 7.5 implies Corollary 3.5. □

Example 7.7. A dichotomy theorem for constraint languages on a three-element set was given as Theorem 4 of [3]. This result is stated in a very similar form to Conjecture 7.5, and is easily shown to be equivalent to this conjecture restricted to three-element algebras. □

Example 7.8. A constraint language containing all unary relations is called a *conservative* constraint language [5]. (One example of a problem with a conservative constraint language is the LIST H-COLORABILITY problem, defined in [19].)

It was shown in Theorem 4 of [5] that, for any conservative constraint language Γ on a finite set A, either Γ is tractable, or else there exists some two-element subset $B \subseteq A$, such that for every polymorphism f of Γ, $f|_B$ is a projection. In this example we will show that this result is equivalent to Conjecture 7.5 restricted to algebras whose operations preserve all unary relations.

Let $\mathcal{A} = (A, F)$ be any finite algebra such that the set of relations $\Gamma = \mathsf{Inv}(F)$ contains all unary relations. Since Γ contains every relation $\{(a)\}$, for each $a \in A$, the algebra $\mathcal{A}$ is idempotent. Furthermore, since Γ contains every relation $\{(a_1), (a_2)\}$, for each two-element subset $B = \{a_1, a_2\} \subseteq A$, it follows that each of the corresponding algebras $\mathcal{B} = (B, F|_B)$ is a subalgebra of $\mathcal{A}$.

Hence, by Theorem 4 of [5], either Γ is tractable, or else there exists some two-element subalgebra $\mathcal{B}$ of $\mathcal{A}$, all of whose operations are projections. Hence this result implies that Conjecture 7.5 holds for any algebra whose operations preserve all unary relations.

Conversely, to show that Conjecture 7.5 implies Theorem 4 of [5], assume that Conjecture 7.5 holds for the algebra $\mathcal{A}$. This implies that either Γ is tractable, or else there exists some nontrivial factor $\mathcal{C}$ of $\mathcal{A}$ all of whose operations are projections. In the latter case, by the definition of a factor, $\mathcal{C}$ must be the image of some subalgebra $\mathcal{B}$ of $\mathcal{A}$ under some homomorphism φ. Choose elements $a_1, a_2 \in \mathcal{B}$ such that their images $\varphi(a_1), \varphi(a_2)$ in $\mathcal{C}$ are distinct. Since every two-element subset of $\mathcal{A}$ is a subalgebra, for any $f \in F$ we have that $f|_{\{a_1,a_2\}}$ has range $\{a_1, a_2\}$. Furthermore, the corresponding operation f' of $\mathcal{C}$ is a projection, so we have that for any tuple $\overline{a}$ over $\{a_1, a_2\}$, the tuple $\varphi(f(\overline{a})) = f'(\varphi(\overline{a}))$ = the ith component of $\varphi(\overline{a})$ for some i. Hence $f|_{\{a_1,a_2\}}$ is also a projection, and we have shown that Conjecture 7.5 implies Theorem 4 of [5]. ∎

Finally, we note that if Conjecture 7.5 is true, then it yields an effective procedure to determine whether any finite constraint language is tractable or **NP**-complete, as the following result indicates.

PROPOSITION 7.9. *Let A be a finite set. If Conjecture* 7.5 *is true, then for any finite constraint language* Γ *over A, it is possible to determine in polynomial time in the size of* Γ *whether* Γ *is* **NP***-complete or tractable.*

Proof. First set $\Gamma' = f(\Gamma) \cup \Gamma_{\mathrm{CON}}$, where f is a unary polymorphism of Γ whose range $f(A) = U$ is minimal and $\Gamma_{\mathrm{CON}} = \{\{(a)\} \mid a \in U\}$. (Note that the number of possible unary operations depends only on $|A|$, so Γ' can be obtained in polynomial time in the size of Γ.) By Proposition 4.2 and Corollary 4.8, Γ is **NP**-complete if and only if Γ' is **NP**-complete, and Γ is tractable if and only if Γ' is tractable.

By Corollary 2.17 and Proposition 2.21, Γ is **NP**-complete if and only if the idempotent algebra $\mathcal{A} = (U, Pol(\Gamma'))$ is **NP**-complete, and Γ is tractable if and only if $\mathcal{A}$ is tractable. By Conjecture 7.5, $\mathcal{A}$ is **NP**-complete if it has a nontrivial factor $\mathcal{B}$ whose operations are all projections; otherwise it is tractable.

Assume first that $\mathcal{A}$ does have a nontrivial factor $\mathcal{B}$ whose operations are all projections, and let $\mathcal{B}$ be the homomorphic image of the subalgebra $\mathcal{A}'$ of $\mathcal{A}$ under the homomorphism φ. We may assume, without loss of generality, that the universe of $\mathcal{B}$ contains the set $\{0,1\}$. The ternary **NP**-complete relation N, defined in Example 2.28, is preserved by all operations of $\mathcal{B}$, so the ternary relation $R = \varphi^{-1}(N)$ is preserved by all operations of $\mathcal{A}'$ and hence by all operations of $\mathcal{A}$. By Proposition 2.21, this implies that $R \in \langle \Gamma' \rangle$.

Conversely, assume that $R = \varphi^{-1}(N) \in \langle \Gamma' \rangle$, where φ is an arbitrary unary function from some subset A' of U onto $\{0,1\}$. By Proposition 2.21, R is preserved by all operations of $\mathcal{A}$, and hence A' is the universe of a subalgebra $\mathcal{A}'$ of $\mathcal{A}$. Furthermore, the relation $\varphi^{-1}(=_{\{0,1\}}) = \exists z R(x,z,z) \wedge R(y,z,z) \in \langle \Gamma' \rangle$. Using standard algebraic results (see Theorem 1.16 of [42]), this implies that φ is a homomorphism from $\mathcal{A}'$ to a two-element factor $\mathcal{B}$ of $\mathcal{A}$ whose operations preserve N. Moreover, $\mathcal{B}$ is idempotent, so all its operations are projections.

It follows that Γ is **NP**-complete if there is a relation $R \in \langle \Gamma' \rangle$ where R is of the form $\varphi^{-1}(N)$ for some unary function φ from some subset of A onto $\{0,1\}$, and in all other cases Γ is tractable. By Proposition 1.1.19 of [48], the presence of any relation R in $\langle \Gamma' \rangle$ can be determined by checking whether R is preserved by all polymorphisms of Γ' of arity bounded by $|R|$ (see [26] for an explicit construction). Since the number of possible unary functions φ is less than $3^{|A|}$, and each $|R| \leq |A|^3$, this condition can be checked in polynomial time in the size of Γ, for any fixed finite set A. ∎

Acknowledgments. 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.

REFERENCES

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

[2] M. Bodirsky and J. Nešetřil, *Constraint satisfaction with countable homogeneous templates*, in Proceedings of Computer Science Logic and the 8th Kurt Gödel Colloquium, Lecture Notes in Comput. Sci. 2803, Springer-Verlag, Berlin, 2003, pp. 44–57.

[3] A. Bulatov, *A dichotomy theorem for constraints on a three-element set*, in Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS'02), Vancouver, BC, Canada, 2002, pp. 649–658.

[4] A. Bulatov, *Mal'tsev Constraints Are Tractable*, Tech. report PRG-RR-02-05, Computing Laboratory, University of Oxford, Oxford, UK, 2002.

[5] A. Bulatov, *Tractable conservative constraint satisfaction problems*, in Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS'03), Ottawa, ON, Canada, IEEE Press, 2003, pp. 321–330.

[6] A. Bulatov and P. Jeavons, *Tractable Constraints Closed under a Binary Operation*, Tech. report PRG-TR-12-00, Computing Laboratory, University of Oxford, Oxford, UK, 2002.

[7] A. Bulatov, A. Krokhin, and P. Jeavons, *The complexity of maximal constraint languages*, in Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC'01), 2001, pp. 667–674.

[8] D. Cohen, P. Jeavons, P. Jonsson, and M. Koubarakis, *Building tractable disjunctive constraints*, J. ACM, 47 (2000), pp. 826–853.

[9] P. Cohn, *Universal Algebra*, Mathematics and Its Applications 6, D. Reidel, Dordrecht, The Netherlands, Boston, 1981.

[10] M. Cooper, D. Cohen, and P. Jeavons, *Characterising tractable constraints*, Artificial Intelligence, 65 (1994), pp. 347–361.

[11] N. Creignou, S. Khanna, and M. Sudan, *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM Monogr. Discrete Math. Appl. 7, SIAM, Philadelphia, 2001.

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

[13] V. Dalmau, *Constraint satisfaction problems in non-deterministic logarithmic space*, in Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02), Lecture Notes in Comput. Sci. 2380, Springer-Verlag, Berlin, 2002, pp. 414–425.

[14] V. Dalmau and J. Pearson, *Set functions and width* 1 *problems*, in Proceedings of the 5th International Conference on Constraint Programming (CP'99), Lecture Notes in Comput. Sci. 1713, Springer-Verlag, Berlin, 1999, pp. 159–173.

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

[16] K. Denecke and S. Wismath, *Universal Algebra and Applications in Theoretical Computer Science*, Chapman and Hall/CRC Press, Boca Raton, FL, 2002.

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

[18] T. Feder, *Constraint satisfaction on finite groups with near subgroups*, submitted; available online from http://theory.stanford.edu/~tomas/near.ps.

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

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

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

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

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

[24] P. Hell and J. Nešetřil, *On the complexity of H-coloring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.

[25] D. Hobby and R. McKenzie, *The Structure of Finite Algebras*, Contemp. Math. 76, AMS, Providence, RI, 1988.
[26] P. Jeavons, *Constructing constraints*, in Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Comput. Sci. 1520, Springer-Verlag, London, 1998, pp. 2–16.
[27] P. Jeavons, *On the algebraic structure of combinatorial problems*, Theoret. Comput. Sci., 200 (1998), pp. 185–204.
[28] P. Jeavons, D. Cohen, and M. Cooper, *Constraints, consistency and closure*, Artificial Intelligence, 101 (1998), pp. 251–265.
[29] P. Jeavons, D. Cohen, and M. Gyssens, *A unifying framework for tractable constraints*, in Proceedings of the 1st International Conference on Constraint Programming (CP'95), Lecture Notes in Comput. Sci. 976, Springer-Verlag, 1995, pp. 276–291.
[30] P. Jeavons, D. Cohen, and M. Gyssens, *Closure properties of constraints*, J. ACM, 44 (1997), pp. 527–548.
[31] P. Jeavons, D. Cohen, and J. Pearson, *Constraints and universal algebra*, Ann. Math. Artificial Intelligence, 24 (1998), pp. 51–67.
[32] P. Jeavons and M. Cooper, *Tractable constraints on ordered domains*, Artificial Intelligence, 79 (1995), pp. 327–339.
[33] L. Kirousis, *Fast parallel constraint satisfaction*, Artificial Intelligence, 64 (1993), pp. 147–160.
[34] P. Kolaitis and M. Vardi, *Conjunctive-query containment and constraint satisfaction*, J. Comput. System Sci., 61 (2000), pp. 302–332.
[35] A. Krokhin, P. Jeavons, and P. Jonsson, *Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra*, J. ACM, 50 (2003), pp. 591–640.
[36] P. Ladkin and R. Maddux, *On binary constraint problems*, J. ACM, 41 (1994), pp. 435–469.
[37] S. Lang, *Algebra*, Grad. Texts in Math., Springer-Verlag, New York, 2002.
[38] D. Lesaint, N. Azarmi, R. Laithwaite, and P. Walker, *Engineering dynamic scheduler for work manager*, BT Technology J., 16 (1998), pp. 16–29.
[39] A. Mackworth, *Consistency in networks of relations*, Artificial Intelligence, 8 (1977), pp. 99–118.
[40] A. Mackworth, *Constraint satisfaction*, in Encyclopedia of Artificial Intelligence, Vol. 1, S. Shapiro, ed., Wiley Interscience, New York, 1992, pp. 285–293.
[41] A. Mackworth and E. Freuder, *The complexity of constraint satisfaction revisited*, Artificial Intelligence, 59 (1993), pp. 57–62.
[42] R. McKenzie, G. McNulty, and W. Taylor, *Algebras, Lattices and Varieties*, Vol. I, Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1987.
[43] U. Montanari, *Networks of constraints: Fundamental properties and applications to picture processing*, Inform. Sci., 7 (1974), pp. 95–132.
[44] B. Nebel and H.-J. Bürckert, *Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra*, J. ACM, 42 (1995), pp. 43–66.
[45] C. Papadimitriou, *Computational Complexity*, Addison–Wesley, Reading, MA, 1994.
[46] J. Pearson and P. Jeavons, *A Survey of Tractable Constraint Satisfaction Problems*, Tech. report CSD-TR-97-15, Royal Holloway, University of London, London, 1997.
[47] N. Pippenger, *Theories of Computability*, Cambridge University Press, Cambridge, UK, 1997.
[48] R. Pöschel and L. Kalužnin, *Funktionen- und Relationenalgebren*, DVW, Berlin, 1979.
[49] E. Post, *The Two-Valued Iterative Systems of Mathematical Logic*, Annals Mathematics Studies 5, Princeton University Press, Princeton, NJ, 1941.
[50] L. Purvis and P. Jeavons, *Constraint tractability theory and its application to the product development process for a constraint-based scheduler*, in Proceedings of the 1st International Conference on the Practical Application of Constraint Technologies and Logic Programming (PACLP'99), The Practical Application Company, Blackpool, UK, 1999, pp. 63–79.
[51] J. Renz and B. Nebel, *On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus*, Artificial Intelligence, 108 (1999), pp. 69–123.
[52] T. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78), 1978, pp. 216–226.
[53] A. Szendrei, *Clones in Universal Algebra*, Séminaire de Mathematiques Supérieures 99, University of Montreal Press, Montreal, QC, Canada, 1986.
[54] A. Szendrei, *Simple surjective algebras having no proper subalgebras*, J. Austral. Math. Soc. Ser. A, 48 (1990), pp. 434–454.
[55] A. Szendrei, *Term minimal algebras*, Algebra Universalis, 32 (1994), pp. 439–477.
[56] P. van Beek and R. Dechter, *On the minimality and decomposability of row-convex constraint networks*, J. ACM, 42 (1995), pp. 543–561.