TRACTABILITY AND LEARNABILITY ARISING FROM ALGEBRAS WITH FEW SUBPOWERS

PAWEŁ IDZIAK[†], PETAR MARKOVIĆ[‡], RALPH MCKENZIE[§], MATTHEW VALERIOTE[¶], AND ROSS WILLARD[‖]

Abstract. A constraint language Γ on a finite set A has been called *polynomially expressive* if the number of n-ary relations expressible by $\exists\wedge$-atomic formulas over Γ is bounded by $\exp(O(n^k))$ for some constant k. It has recently been discovered that this property is characterized by the existence of a $k+1$-ary polymorphism satisfying certain identities; such polymorphisms are called *k-edge operations* and include Mal'cev and near-unanimity operations as special cases.

We prove that if Γ is any constraint language which, for some $k > 1$, has a k-edge operation as a polymorphism, then the constraint satisfaction problem for $\langle\Gamma\rangle$ (the closure of Γ under $\exists\wedge$-atomic expressibility) is globally tractable. We also show that the set of relations definable over Γ using quantified generalized formulas is polynomially exactly learnable using improper equivalence queries.

Key words. constraint satisfaction, learnability, complexity, polymorphism, Mal'cev, near unanimity, polynomially expressive, few subpowers

AMS subject classifications. 08A70, 68Q25, 69T99

1. Introduction. A wide variety of combinatorial problems can be expressed within the framework of the Constraint Satisfaction Problem (CSP). Given an instance of the CSP, the aim is to determine if there is an assignment of values to the variables of the instance that satisfies all of its constraints. While the class of all CSPs forms an **NP**-complete class of problems, there are many naturally defined subclasses that are tractable (i.e., lie in the class **P**). A main research goal is to identify the subclasses of the CSP that are tractable.

One common way to define a subclass of the CSP is to restrict the relations that appear in the constraints of an instance to a specified set of relations over some fixed domain, called a constraint language. The Dichotomy Conjecture of Feder and Vardi [19] states that for any constraint language Γ, the corresponding subclass of the CSP, denoted CSP(Γ), is either **NP**-complete or tractable. The Dichotomy Conjecture has been verified in a number of special cases, most notably over domains of size 2 [28] and size 3 [7]. Over larger domains, a number of general results have been obtained [2, 3, 6, 10, 13, 17, 25].

In this paper we introduce a condition which, when satisfied by a constraint language Γ, guarantees that the subclass CSP(Γ) is tractable. The polynomial-time algorithm that we use to establish tractability is essentially that presented by Dalmau in [17] and our result generalizes his and also generalizes earlier results of Feder and Vardi, Bulatov, and Jeavons, Cohen and Cooper [19, 9, 24]. The class of constraint languages that satisfy our condition is quite large and includes some well known constraint languages such as those that have Mal'cev or near unanimity polymorphisms.

A polymorphism of a set of relations Γ over some set A is a finitary operation on A that preserves all of the relations in Γ. Work of Jeavons and his co-authors has shown that the

[†]Theoretical Computer Science Department, Jagiellonian University, Poland (idziak@tcs.uj.edu.pl). Research supported by Polish MNiSW grant no. N206-2106-33.

[‡]Department of Mathematics and Informatics, University of Novi Sad, Serbia (pera@dmi.ns.ac.rs). Research supported by grant no. 144011 of the Ministry of Science and Technology of Serbia.

[§]Department of Mathematics, Vanderbilt University, USA (ralph.n.mckenzie@vanderbilt.edu).

[¶]Department of Mathematics and Statistics, McMaster University, Canada (matt@math.mcmaster.ca). Research supported by the Natural Sciences and Engineering Research Council of Canada.

[‖]Department of Pure Mathematics, University of Waterloo, Canada (rdwillar@uwaterloo.ca). Research supported by the Natural Sciences and Engineering Research Council of Canada.

tractability of a constraint language is determined by the set of its polymorphisms and many tractability results can be expressed in terms of the existence of certain kinds of polymorphisms [11]. A benefit of focusing on polymorphisms is that it allows the introduction of ideas and techniques from universal algebra into the study of tractability and in this paper we adopt this algebraic approach.

Our main result is that if a constraint language Γ has a k-edge operation (see Definition 3.3) as a polymorphism, then Dalmau's algorithm can be used, essentially unchanged, to solve in polynomial time all instances of CSP(Γ). This algorithm can be regarded as a natural generalization of the familiar Gaussian elimination algorithm for solving systems of linear equations since it makes essential use of the fact that, in this case, all solution sets of instances of CSP(Γ) have small generating sets (akin to bases of vector spaces) when considered as universes of particular algebras.

On the other hand, we argue that if a constraint language Γ fails to have a k-edge operation as a polymorphism then no "robust Gaussian-like" algorithm can solve all instances of CSP(Γ) in polynomial-time. So, in some sense, our result sharply determines the scope of algorithms like Dalmau's to quickly settle instances of the CSP.

In Section 5 we establish that if Γ is a set of relations over a finite set that is invariant under a k-edge operation then the set of relations defined by quantified generalized formulas over Γ is exactly learnable, in polynomial time, via an algorithm that makes improper equivalence queries. This extends the result of Bulatov, Chen, and Dalmau in [8] and earlier results of Dalmau and Jeavons [18].

This paper is an extended version of [22].

2. Preliminaries. DEFINITION 2.1. *An* instance *of the constraint satisfaction problem is a triple $P = (V, A, \mathcal{C})$ with*

- *V a non-empty, finite set (called the set of* variables*),*
- *A a non-empty, finite set (called the* domain*),*
- *$\mathcal{C}$ a set of* constraints *$\{C_1, \ldots, C_q\}$ where each C_i is a pair $(\vec{s}_i, R_i)$ with*
  - *$\vec{s}_i$ a tuple of variables of length m_i, called the* scope *of C_i, and*
  - *R_i an m_i-ary relation over A, called the* constraint relation *of C_i.*

Given an instance P of the CSP we wish to answer the following question:

Is there a solution *to P, i.e., is there a function $f : V \to A$ such that for each $1 \leq i \leq q$, the m_i-tuple $f(\vec{s}_i) \in R_i$?*

The full CSP is **NP**-complete (see [24]), but by restricting the nature of the constraint relations that are allowed to appear in an instance, it is possible to find natural subclasses of the CSP that are tractable.

DEFINITION 2.2. *Let A be a domain and Γ a set of finitary relations over A. CSP(Γ) denotes the collection of all instances of the CSP with domain A and with constraint relations coming from Γ. Γ is called the* constraint language *of the class CSP(Γ).*

DEFINITION 2.3. *Let Γ be a constraint language. We say that Γ is* tractable *(or more precisely, is* locally tractable*) if for every finite subset Γ' of Γ the class CSP(Γ') lies in* **P***. If CSP(Γ) itself is in* **P** *then we say that Γ is* globally tractable*. Γ is said to be* **NP**-complete *if for some finite subset Γ' of Γ, the class CSP(Γ') is* **NP***-complete.*

A key problem in this area is to classify the (globally) tractable constraint languages. Note that in this paper we will assume that $\mathbf{P} \neq \mathbf{NP}$. Feder and Vardi [19] conjecture that every finite constraint language is either tractable or is **NP**-complete.

The natural duality between sets of relations (constraint languages) over a set A and sets of operations (algebras) on A has been studied by algebraists for some time. Jeavons and his co-authors [23] have shown how this link between constraint languages and algebras can be

used to transfer questions about tractability into equivalent questions about algebras. In the remainder of this section we present a concise overview of this connection.

DEFINITION 2.4. *Let A be a non-empty set, Γ a set of finitary relations on A, F a set of finitary operations on A, R an n-ary relation on A, and f an m-ary operation on A.*

1. *We say that R is* invariant *under f and that f is a* polymorphism *of R if for all $\vec{a}_i \in R$, for $1 \leq i \leq m$, the n-tuple $f(\vec{a}_1, \ldots, \vec{a}_m)$, i.e. whose j-th coordinate is $f(\vec{a}_1(j), \ldots, \vec{a}_m(j))$, belongs to R.*
2. $\mathrm{Pol}(\Gamma)$ *denotes the set of functions on A that are polymorphisms of all the relations in Γ.*
3. $\mathrm{Inv}(F)$ *denotes the set of all finitary relations on A that are invariant under all operations in F.*
4. $\langle\Gamma\rangle$ *denotes* $\mathrm{Inv}(\mathrm{Pol}(\Gamma))$*, the* relational clone *on A generated by Γ.*

We note that given a set of relations Γ over a finite set A, the relational clone generated by Γ is equal to the set of relations over A definable from Γ using primitive-positive formulas (or conjunctive queries) (see [23]).

THEOREM 2.5. ([23]) *Let Γ be a constraint language on a finite set. If Γ is tractable then so is $\langle\Gamma\rangle$. If $\langle\Gamma\rangle$ is* **NP***-complete then so is Γ.*

We refer the reader to [12] or [27] for the basics of universal algebra, in particular to the notions of a cartesian power and a subuniverse of an algebra. By an algebra we mean the following:

DEFINITION 2.6. *An* algebra **A** *is a pair (A, F) where A is a non-empty set (called the* universe *or* domain*) and F is a (possibly infinite) collection of finitary operations on A. The operations in F are called the* basic operations *of* **A**. *A* term operation *of an algebra* **A** *is a finitary operation on A that can be obtained by repeated compositions of the basic operations of* **A**. *An algebra is* finite *if its universe is finite.*

DEFINITION 2.7. *Let $\mathbf{B} = (B, F)$ be a finite algebra and Γ a constraint language over A.*

1. $\mathbf{A}_\Gamma$ *denotes the algebra $(A, \mathrm{Pol}(\Gamma))$ and $\Gamma_\mathbf{B}$ denotes the constraint language $\mathrm{Inv}(F)$.*
2. *We call the algebra* **B** tractable, globally tractable, *or* **NP**-complete *if the constraint language $\Gamma_\mathbf{B}$ is.*

Note that if Γ is a constraint language, then $\Gamma_{(\mathbf{A}_\Gamma)} = \mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \langle\Gamma\rangle$. Hence in algebraic terms, Theorem 2.5 states that a constraint language Γ is tractable (or **NP**-complete) if and only if the algebra $\mathbf{A}_\Gamma$ is. Thus, the problem of characterizing the tractable constraint languages can be reduced to the special case of characterizing the tractable finite algebras.

This is the starting point of the so-called "algebraic method" for attempting to verify the Dichotomy Conjecture of Feder and Vardi. Using algebraic insights made possible by this method, Bulatov, Jeavons and Krokhin [11] have conjectured a precise characterization of which constraint languages should be tractable. Great strides have been made recently in partially confirming the latter conjecture [7, 25, 13, 3, 2]; see the excellent survey article [21, Section 3] for more details. The present paper may be viewed as another step towards verification of both conjectures.

3. Algebras with few subpowers. Let Γ be a constraint language on a set A. Note that (i) for any instance $P = (V, A, C)$ of CSP(Γ), its solution set can be naturally identified with an n-ary member of $\langle\Gamma\rangle$ where $n = |V|$, and (ii) the n-ary members of $\langle\Gamma\rangle$ are precisely the universes of the subalgebras of $(\mathbf{A}_\Gamma)^n$. Since Dalmau's algorithm mentioned in the introduction requires that these subalgebras of powers of $\mathbf{A}_\Gamma$ have algebraic generating sets of small (polynomial in n) size, it is natural to ask when *all* subalgebras of powers of $\mathbf{A}_\Gamma$ have small generating sets. This leads to the following concepts:

DEFINITION 3.1. *Let* **A** *be a finite algebra.*

1. *A* subpower *of* $\mathbf{A}$ *is a subalgebra of* $\mathbf{A}^n$ *for some* $n \geq 1$.

2. *For* $n \geq 1$, *define* $g_{\mathbf{A}}(n)$ *to be the least integer* t *such that every subalgebra of* $\mathbf{A}^n$ *has a generating set of size at most* t.

3. *For* $n \geq 1$, *define* $s_{\mathbf{A}}(n)$ *to be logarithm, base 2, of the cardinality of the set of all subalgebras of* $\mathbf{A}^n$.

4. *(Cf. [16])* $\mathbf{A}$ *has* polynomially generated subpowers *if the function* $g_{\mathbf{A}}$ *is bounded above by a polynomial.*

5. $\mathbf{A}$ *has* few subpowers *[22] if the function* $s_{\mathbf{A}}$ *is bounded above by a polynomial.*

We note that a constraint language Γ is polynomially expressive as defined in [14] if and only if $\mathbf{A}_\Gamma$ has few subpowers.

The following lemma is an immediate consequence of [4, Proposition 1.2].

LEMMA 3.2. *A finite algebra* $\mathbf{A}$ *has polynomially generated subpowers if and only if it has few subpowers.*

Thus our interest in algebras whose subpowers have small generating sets leads naturally to the notion of algebras having few subpowers. Finite algebras with few subpowers are thoroughly studied and characterized in [4]. The remainder of this section summarizes key definitions, examples and results from that paper that are needed here.

DEFINITION 3.3. *Let* $k \geq 2$. *A* k-edge operation *on a set* A *is a* $k+1$*-ary operation* $e(\bar{x})$ *on* A *that universally satisfies the* k *identities*

$$\begin{aligned}
e(x,x,y,y,y,\ldots,y,y) &\approx y\\
e(x,y,x,y,y,\ldots,y,y) &\approx y\\
e(y,y,y,x,y,\ldots,y,y) &\approx y\\
e(y,y,y,y,x,\ldots,y,y) &\approx y\\
&\vdots\\
e(y,y,y,y,y,\ldots,x,y) &\approx y\\
e(y,y,y,y,y,\ldots,y,x) &\approx y.
\end{aligned}$$

(Note the "shepherd's crook" shape of the occurrences of x *on the left-hand side of these identities.)*

Examples. Let A be a set.

1. A *Mal'cev operation* on A is a ternary operation $p(x,y,z)$ that satisfies the identities $p(y,x,x) \approx p(x,x,y) \approx y$. It is clear that a ternary operation $p(x,y,z)$ on A is Mal'cev if and only if $p(y,x,z)$ is a 2-edge operation on A.

2. A k-ary *near unanimity operation* on A is a k-ary operation $t(\bar{x})$ that satisfies the identities

$$t(x,y,\ldots,y) \approx t(y,x,y,\ldots,y) \approx t(y,y,x,y\ldots,y) \approx \cdots \approx t(y,\ldots,y,x) \approx y$$

If t is a k-ary near unanimity operation then the $k+1$-ary operation $t(x_2,x_3,\ldots,x_{k+1})$ is a k-edge operation.

3. A k-ary operation g on A is a *generalized majority-minority* (*gmm*) *operation* ([8, 17]) if for all $a, b \in A$, we have that either for all $x, y \in \{a,b\}$,

$$g(x,y,\ldots,y) = g(y,x,y,\ldots,y) = g(y,y,x,y\ldots,y) = \cdots = g(y,\ldots,y,x) = y$$

or for all $x, y \in \{a,b\}$,

$$g(y,x,\ldots,x) = g(x,\ldots,x,y) = y.$$

From [4, Theorem 4.7] we know that any algebra with a k-ary gmm term operation also has a k-edge term operation.

THEOREM 3.4. [4, Corollary 3.11] *Let* $\mathbf{A}$ *be a finite algebra. Then* $\mathbf{A}$ *has few subpowers if and only if for some* $k \geq 2$, $\mathbf{A}$ *has a* k*-edge term operation. In this case,* $s_{\mathbf{A}}(n)$ *is bounded above by a polynomial of degree* k*.*

We note that in [4] it is also established that if a finite algebra $\mathbf{A}$ fails to have few subpowers then the function $s_{\mathbf{A}}$ is bounded below by an exponential function. This extends the result of Chen for two element algebras found in [14].

One direction of the proof of Theorem 3.4 establishes that in the presence of a k-edge term operation, all subalgebras of finite powers of a finite algebra $\mathbf{A}$ have rather small generating sets; in fact, the function $g_{\mathbf{A}}$ is in $O(n^{k-1})$. The proof utilizes three auxiliary term operations that are derivable from a k-edge term operation.

LEMMA 3.5 ([4]). *Let* $\mathbf{A}$ *be a finite algebra with a* k*-edge term operation* e*. Then the algebra* (A, e) *has term operations* $d(x, y)$*,* $p(x, y, z)$ *and* $s(x_1, x_2, \ldots, x_k)$ (*which are also term operations of* $\mathbf{A}$) *satisfying*

$$\begin{aligned} p(x,y,y) &\approx x \\ p(x,x,y) &\approx d(x,y) \\ d(x,d(x,y)) &\approx d(x,y) \\ s(y,x,x,x,\ldots,x,x) &\approx d(x,y) \\ s(x,y,x,x,\ldots,x,x) &\approx x \\ s(x,x,y,x,\ldots,x,x) &\approx x \\ &\vdots \\ s(x,x,x,x,\ldots,x,y) &\approx x. \end{aligned}$$

The following two definitions are generalizations of those given in [8, 17] in the context of generalized majority-minority operations.

DEFINITION 3.6. *Suppose* $\mathbf{A}$ *is an algebra with* k*-edge term operation* e *and term operations* d*,* p*,* s *as in Lemma 3.5. A pair* $(a, b) \in A^2$ *is a* minority *pair if* $d(a, b) = b$*.*

By an *index (of rank* n*)* we mean a triple $I = (i, a, b)$ where $1 \leq i \leq n$ and $a, b \in A$. It is said to be a *minority* index if (a, b) is a minority pair. If $S \subseteq A^n$, we say that I is *witnessed* in S if there exist $f, g \in S$ satisfying

- $f(j) = g(j)$ for all $j < i$.
- $f(i) = a$ and $g(i) = b$.

In this case we call f, g *witnesses* to I.

DEFINITION 3.7. *Fix* $k > 1$*, let* $\mathbf{A}$ *be an algebra with* k*-edge term operation* e *and term operations* d*,* p*,* s *as in Lemma 3.5, and suppose* $R \subseteq B \subseteq A^n$ *for some* $n > 0$*.*

1. *The* signature *of* B*, denoted by* Sig_B*, is the set of all minority indices of rank* n *that are witnessed in* B*.*
2. *We say that* R *is a* representation *(more precisely, a* k-representation*) of* B *if*
   - *For all* $T \subseteq [n] := \{1, 2, \ldots, n\}$ *with* $|T| < k$*, the projections of* B *and* R *to* A^T *are identical.*
   - $Sig_R = Sig_B$*.*
3. $\langle R \rangle_e$ *denotes the closure of* R *under the operation* e *(or equivalently the smallest* n*-ary relation on* A *that contains* R *and is invariant under* e*).*

LEMMA 3.8. *Let* $\mathbf{A}$ *be a finite algebra with* k*-edge term operation* e *and term operations* d*,* p*,* s *as in Lemma 3.5. Suppose* $\mathbf{B}$ *is a subalgebra of* $\mathbf{A}^n$ *with universe* B*,* R *is a represen-*

tation of B, and $I = (i, a, b)$ is a minority index witnessed in B. Then for all $f \in \langle R\rangle_e$ with $f(i) = a$, there exists $g \in \langle R\rangle_e$ such that f, g witness I.

Proof. Because I is minority and witnessed in B, we can choose $f^*, g^* \in R$ witnessing I. Define

$$g = p(f, f^*, g^*)$$

and note that $g \in \langle R\rangle_e$ as p can be expressed as a repeated composition of e. Because $p(x, y, y) \approx x$ we have $f(j) = g(j)$ for all $j < i$. At coordinate i,

$$g(i) = p(a, a, b) \;=\; d(a, b) \;=\; b$$

where the last equality holds because (a, b) is minority. □

The proof of the following theorem may be found in [4, Corollary 3.9].

THEOREM 3.9. *Suppose* **A** *is a finite algebra with k-edge term operation e and term operations d, p, s as in Lemma 3.5. If* **B** *is a subalgebra of* $\mathbf{A}^n$ *and R is a representation of* **B**, *then $\langle R\rangle_e = B$.*

As observed in [8] and [17], every subset B of A^n has a representation whose size is bounded above by a polynomial in n. In fact, if we set $m = \min(k-1, n)$ then B has a representation of size at most $2|\mathrm{Sig}_B| + \sum_{T\subseteq[n],|T|=m} |\mathrm{proj}_T(B)|$. We will call a representation with this property a compact representation of B. A straightforward computation (noted in Lemma 4 of [8]) establishes that every subset of A^n has a compact representation of size bounded above by a $(k-1)$-degree polynomial in n. This explains our earlier claim that the function $g_{\mathbf{A}}$ is in $O(n^{k-1})$ when **A** has a k-edge term operation. We remark that if, conversely, the function $g_{\mathbf{A}}$ is in $O(n^{k-1})$, then it is a consequence of results in [4] that **A** has a $k+1$-edge term operation.

Note that given a subset R of A^n, R will be a compact representation of some subset of A^n if and only if it is a compact representation of itself. This can be determined by comparing the size of R with $2|\mathrm{Sig}_R| + \sum_{T\subseteq[n],|T|=m} |\mathrm{proj}_T(R)|$, where $m = \min(k-1, n)$, a calculation that can be carried out in time bounded by a polynomial in n and the size of R.

4. Tractability. In this section we argue that, with slight modifications, the polynomial-time algorithm presented in [17] for algebras having a gmm term operation works for finite algebras having an edge term operation. Thus we claim the following result:

THEOREM 4.1. *Let* **A** *be a finite algebra with few subpowers. The constraint language* $\Gamma_{\mathbf{A}}$ *consisting of all subuniverses of finite cartesian powers of* **A** *is globally tractable.*

COROLLARY 4.2. *Any constraint language over a finite set A that has, for some $k > 1$, a k-edge operation as a polymorphism is globally tractable.*

Note that this settles a conjecture posed by Chen in [14] and Conjecture 1 from Dalmau's thesis [16], namely that any algebra that has polynomially generated subpowers is tractable.

For the remainder of this section, we assume that **A** is a finite algebra of the form $\langle A, \varphi(x_1, \ldots, x_{k+1})\rangle$, where φ is a k-edge operation. Note that in order to prove Theorem 4.1 it will suffice to consider algebras of this form.

For $P = (\{v_1, \ldots, v_n\}, A, \{C_1, \ldots, C_m\})$ an instance of CSP($\Gamma_{\mathbf{A}}$), and for $0 \le l \le m$, let $P_l = (\{v_1, \ldots, v_n\}, A, \{C_1, \ldots, C_l\})$, be the instance of CSP($\Gamma_{\mathbf{A}}$) obtained from P by only using the first l constraints of P and let R_l denote the set of solutions of P_l.

In essence, the Dalmau Algorithm starts off with a compact representation of R_0 ($= A^n$) and then recursively constructs compact representations for R_l, for $0 < l \le m$. At the end of this recursion, we have R_m, a compact representation for the set of solutions of P, and so P will have a solution if and only if R_m is non-empty. Of course, the algorithm can be easily modified so that it outputs a solution of P if one exists.

Algorithm `Dalmau`$((\{v_1, \ldots, v_n\}, A, \{C_1, \ldots, C_m\}))$
Step 1 **set** R'_0 some compact representation of A^n
Step 2 **for each** $l \in \{0, \ldots, m-1\}$ **do**
(let C_{l+1} **be** $((v_{i_1}, \ldots, v_{i_{k_{l+1}}}), S_{l+1})$**)**
Step 2.1 **set** $R'_{l+1} := \texttt{Next}(R'_l, i_1, \ldots, i_{k_{l+1}}, S_{l+1})$
end for each
Step 3 **if** $R'_m \neq \emptyset$ **return yes**
Step 4 **return no**

The `Dalmau` algorithm makes use of a number of procedures that are defined and analyzed in [17]. We now present a description of these procedures, pointing out any necessary modifications, along with estimates of their run-times. Unless otherwise noted, justifications for the correctness and bounds on run-times can be found in Section 4 of [17].

- `Nonempty`$(R', i_1, \ldots, i_j, S)$ receives as input a compact representation R' of a subuniverse R of $\mathbf{A}^n$ for some n, a sequence of elements $i_1, \ldots, i_j$ from $[n]$, and a subuniverse S of $\mathbf{A}^j$. If there is some element $\mathbf{t} \in R$ such that $\mathrm{proj}_{\{i_1,\ldots,i_j\}}(\mathbf{t}) \in S$ then `Nonempty` outputs some member of R with this property and if not, returns the answer "no". The running time of `Nonempty` can be bounded by $O(((n|A|)^k + |\mathrm{proj}_{\{i_1,\ldots,i_j\}}(R)|)^{k+2}|S|n)$.
- `Fix-values`$(R', a_1, \ldots, a_m)$ receives as input a compact representation R' of a subuniverse R of $\mathbf{A}^n$ for some n and a sequence of elements $a_1, \ldots, a_m$ from A and outputs a compact representation of the subuniverse

$$\{\mathbf{t} \in R : \mathrm{proj}_{[m]}(\mathbf{t}) = (a_1, a_2, \ldots, a_m)\}.$$

  The only change in the definition of `Fix-values` found in Section 4.2 of [17] is that Step 2.2.1.2 has been removed and Step 2.2.1.1 changed to
  set $\mathbf{t}_5 := p(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)$
  where $p(x, y, z)$ is the operation derived from the k-edge operation φ using Lemma 3.5. See Figure 4.1 for a presentation of the modified `Fix-values`. The proof of the correctness of this modified version of `Fix-values` is similar to the proof found in Section 4.2 of [17] but makes use of our Lemma 3.8. The running time of `Fix-values` can be bounded by $O((n|A|)^{(k+1)(k+2)})$.
- `Next`$(R', i_1, \ldots, i_j, S)$ receives as input a compact representation R' of a subuniverse R of $\mathbf{A}^n$ for some n, a sequence of elements $i_1, \ldots, i_j$ from $[n]$, and a subuniverse S of $\mathbf{A}^j$. It outputs a compact representation of the subuniverse

$$R^* = \{\mathbf{t} \in R : \mathrm{proj}_{\{i_1,\ldots,i_j\}}(\mathbf{t}) \in S\}.$$

  Note that `Next` makes use of a similar procedure, called `Next-beta`, but which has a potentially worse running time. As noted in Section 4.3 of [17], the running time of each call to `Next` in line 2.1 of the `Dalmau` Algorithm is bounded by $O((n|A||S|)^{(k+2)^2})$.

COROLLARY 4.3. *The algorithm* `Dalmau` *correctly decides if an instance P of CSP($\Gamma_{\mathbf{A}}$) has a solution in time $O(m(n|A||S^*|)^{(k+2)^2})$, where n is the number of variables of P, m is the number of constraints of P, and S^* is the largest constraint relation occurring in P.*

To conclude this section, we describe a sense in which Corollary 4.3 is the most general "robust Gaussian-like" tractable algorithm for CSP. Clearly both the usual Gaussian elimination over a finite field and the Dalmau algorithm in [17] share the property that solution sets to instances of a system of "constraints" in n variables have "small generating sets" with respect

Algorithm `Fix-values`$(R', a_1, \ldots, a_m)$
Step 1 **set** $j := 0$; $U_j := R'$
Step 2 **while** $j < m$ **do**
Step 2.1 **set** $U_{j+1} := \emptyset$
Step 2.2 **for each** $(i, a, b) \in [n] \times A^2$, with (a, b) a minority pair, **do**
Step 2.2.1 **if** `Nonempty`$(U_j, j+1, i, \{(a_{j+1}, a)\}) \neq$ "no" and
$\exists \mathbf{t_2}, \mathbf{t_3} \in U_j$ witnessing (i, a, b) and
$i > j + 1$ **then**
(let $\mathbf{t_1}$ be the tuple returned by `Nonempty`$(U_j, j+1, i, \{a_{j+1}, a\})$)
Step 2.2.1.1 **set** $\mathbf{t_5} := p(\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3})$
Step 2.2.1.2 **set** $U_{j+1} := U_{j+1} \cup \{\mathbf{t_1}, \mathbf{t_5}\}$
end for each
Step 2.3 **for each** $k' \in [k-1]$
for each $l_1, \ldots, l_{k'} \in [n]$ with $l_1 < l_2 < \cdots < l_{k'}$
for each $d_1, \ldots, d_{k'} \in A$ **do**
Step 2.3.1 **if** `Nonempty`$(U_j, l_1, \ldots, l_{k'}, j+1, \{(d_1, \ldots, d_{k'}, a_{j+1})\}) \neq$ "no"
then (let $\mathbf{t_6}$ be the tuple returned by
`Nonempty`$(U_j, l_1, \ldots, l_{k'}, j+1, \{(d_1, \ldots, d_{k'}, a_{j+1})\})$)
set $U_{j+1} := U_{j+1} \cup \{\mathbf{t_6}\}$
end for each
Step 2.4 **set** $j := j + 1$
end while
Step 3 **return** U_m

FIG. 4.1. *The* `Fix-values` *Algorithm*

to an appropriate algebraic structure, where "small" means "bounded by a polynomial in n." We have seen that if Γ is a constraint language and Γ has a k-edge polymorphism for some k, then the Dalmau algorithm can be applied to CSP(Γ) with these properties holding true. What is more, the Dalmau algorithm applies globally to CSP(Γ') where $\Gamma' = \langle\Gamma\rangle = \mathrm{Inv}(\mathrm{Pol}(\Gamma))$.

Suppose, conversely, that Γ is a constraint language satisfying $\Gamma = \langle\Gamma\rangle$ and that some algorithm with these properties applies globally to solve CSP(Γ) in polynomial time. If we put $\mathbf{A} = \mathbf{A}_\Gamma$ and note that every subpower of $\mathbf{A}$ is a member of Γ and hence can be defined by a single constraint from Γ, it follows immediately from the assumed properties that $\mathbf{A}$ has polynomially generated subpowers. Hence Γ has a k-edge polymorphism by Lemma 3.2 and Theorem 3.4.

5. Learnability. In this section we show how to modify the proof of Bulatov, Chen, and Dalmau found in [8] to show that when f is a k-edge operation on a finite set A, there is an algorithm that exactly learns, in polynomial time, the set $\mathrm{Inv}(f)$, encoded by compact representations.

First we give a brief overview of the learning model that is used. More details may be found in Angluin and Kharitonov [1], or in [18, 8]. We fix a finite set A and define X to be the set of all finitary tuples over A. A *concept* c is simply a subset of X along with some sort of encoding of it, while a *concept class* is just a set of concepts.

A *learning algorithm* for a concept class C is a procedure that, by making specific kinds of queries to an oracle, eventually produces an encoding of a given target concept t from C. The model that we adopt is called the *exact model with equivalence queries*. Learning algorithms in this model are allowed to provide the oracle with a hypothetical encoding h

of the target concept t and the oracle either confirms that h indeed encodes t or it returns a counterexample from the symmetric difference of t and the concept coded by h. If the hypothesis h codes a concept that does not belong to the class C then the query made to the oracle is said to be *improper*.

A learning algorithm is said to *learn* a concept class C if for every target concept t from C, it halts with an encoding of t. The algorithm *runs in polynomial time* if its run-time can be bounded by a polynomial in the size of the encoding of the target concepts and the largest counterexample returned by the oracle. A concept class C is *polynomially learnable with equivalence queries* if there is a learning algorithm that learns C and that runs in polynomial time. If there is a polynomial time learning algorithm that only makes proper equivalence queries to the oracle then C is said to be *polynomially learnable with proper equivalence queries*; otherwise, C is said to be *polynomially learnable with improper equivalence queries*.

A related notion is that of being *polynomially evaluable* ([18]). A concept class C has this property if there is a polynomial time algorithm that, on input the code of a concept c from C and a tuple $\mathbf{x}$ from X, determines if $\mathbf{x}$ is in c or not.

For the remainder of this section, let A be a finite set, $f(x_1, \ldots, x_{k+1})$ a k-edge operation on A, and d, p, and s the operation derived from f as in Lemma 3.5. We let $\mathrm{Inv}(f)$ denote the set of all finitary relations over A that are invariant under f. As in Definition 3.7, for $R \subseteq A^n$, $\langle R \rangle_f$ denotes the smallest relation invariant under f that contains R (or equivalently, the subuniverse of $(A, f)^n$ generated by R).

For $\mathrm{Inv}(f)$ to be considered as a concept class, we need to specify an encoding for each of its members and for convenience, rather than using the notion of a signature developed in Section 4 of [8], we will use compact representations. In what follows, we could just as easily adopt the signature formalism from [8].

Recall that every n-ary relation R over A has a compact representation (relative to f, d, p and s) and that if R is invariant under f then every one of its compact representations generates R as a subuniverse of $(A, f)^n$. Thus, we may use compact representations to code members of $\mathrm{Inv}(f)$.

An example presented in Section 4 of [8] shows that not all compact representations of relations over A are necessarily compact representations of relations invariant under f. This necessitates expanding our concept class to include arbitrary relations over A, encoded by compact representations via a variation of the `Fix-values` routine from the previous section. The following Lemma establishes an interpolation property that we use to modify `Fix-values`.

LEMMA 5.1. *Suppose $R \subseteq A^n$ is a compact representation, $i \in [n]$, $\mathbf{a} \in A^i$, $\mathbf{b} \in R$ satisfying $a_j = b_j$ for all $1 \leq j < i$, and $c_i = d(a_i, b_i)$. Suppose further that*

1. *For each $I \subseteq [i]$ with $|I| = \min(k-1, i)$ and $i \in I$ we have an element $\mathbf{c}^I \in R$ with $\mathrm{proj}_I(\mathbf{c}^I) = \mathrm{proj}_I(\mathbf{a})$.*
2. *If $c_i \neq a_i$, then (i, a_i, c_i) is witnessed in R, say by $\mathbf{u}, \mathbf{v}$.*

Then $\{\mathbf{t} \in \langle R \rangle_f : t_j = a_j \text{ for } 1 \leq j \leq i\} \neq \emptyset$.

Proof. For notational convenience, if $c_i = a_i$ then let $\mathbf{u} = \mathbf{v} = \mathbf{c}^{[k-2] \cup \{i\}}$. We claim that the algorithm `Interpolate` found in Figure 5.1 returns an element in $\{\mathbf{t} \in \langle R \rangle_f : t_j = a_j \text{ for } 1 \leq j \leq i\}$. For $i \geq k$, we prove by induction on $j = 0, 1, \ldots, i-k+1$ that if $I \subseteq \{j+1, j+2, \ldots, i\}$ with $|I| = k-1$ and $i \in I$, then $c_t^{j,I} = a_t$ for all $t \in [j] \cup I$.

The base of the induction, when $j = 0$, follows since $\mathrm{proj}_I(\mathbf{c}^I) = \mathrm{proj}_I(\mathbf{a})$ and $\mathbf{c}^{0,I} = \mathbf{c}^I$. Assume that $j > 0$ and that the $\mathbf{c}^{j-1,L}$ have the desired property for all $L \subseteq \{j, j+1, \ldots, i\}$ with $|L| = k-1$ and $i \in L$. Straightforward calculations show that the elements $\mathbf{d}$ and $\mathbf{c}^{j,I}$ produced in Steps 3.1.3 and 3.1.5 of `Interpolate` are such that $d_l = a_l$ for $l \in [j] \cup (I \setminus \{i\})$, $d_i = c_i$ and that $\mathrm{proj}_{[j] \cup I}(\mathbf{c}^{j,I}) = \mathrm{proj}_{[j] \cup I}(\mathbf{a})$, as required. □

Algorithm `Interpolate`$(i, \mathbf{b}, (\mathbf{c}^I)_I, \mathbf{u}, \mathbf{v})$
Step 1 **if** $i < k$ **then return** $\mathbf{c}^{[i]}$
Step 2 **for** $I \subseteq [i]$ with $|I| = k-1$ and $i \in I$
Step 2.1 **set** $\mathbf{c}^{0,I} := \mathbf{c}^I$
next I
Step 3 **for** $j = 1, 2, \ldots, i-k+1$
Step 3.1 **for** $I \subseteq \{j+1, \ldots, i\}$ with $|I| = k-1$ and $i \in I$
Step 3.1.1 **enumerate** $I = \{\ell_1, \ell_2, \ldots, \ell_{k-2}, i\}$ with $\ell_1 < \ell_2 < \cdots < \ell_{k-2} < i$
Step 3.1.2 **for** $t = 1, 2, \ldots, k-2$
Step 3.1.2.1 **set** $J_t := (I \setminus \{\ell_t\}) \cup \{j\}$
next t
Step 3.1.3 **set** $\mathbf{d} := s(\mathbf{b}, \mathbf{c}^{j-1,I}, \mathbf{c}^{j-1,J_1}, \mathbf{c}^{j-1,J_2}, \ldots, \mathbf{c}^{j-1,J_{k-2}})$
Step 3.1.4 **set** $\mathbf{e} := p(\mathbf{c}^{j-1,I}, \mathbf{u}, \mathbf{v})$
Step 3.1.5 **set** $\mathbf{c}^{j,I} := f(\mathbf{e}, \mathbf{d}, \mathbf{c}^{j-1,I}, \mathbf{c}^{j-1,J_1}, \mathbf{c}^{j-1,J_2}, \ldots, \mathbf{c}^{j-1,J_{k-2}})$
next I
next j
Step 4 **set** $j := i-k+1$
Step 5 **return** $\mathbf{c}^{j,\{j+1,\ldots,i\}}$

FIG. 5.1. *The* `Interpolate` *Algorithm*

Algorithm `New-Fix-values`$(R, a_1, \ldots, a_i)$
Step 1 **if** $i = 0$ **then return** R
Step 2 **set** $U :=$ `Fix-values`$(R, a_1, \ldots, a_i)$
Step 3 **if** $U \neq \emptyset$ **then return** U
Step 4 **if** there exists $\mathbf{b} \in$ `New-Fix-values`$(R, a_1, \ldots, a_{i-1})$ with $c_i := d(a_i, b_i)$,
and for every $I \subseteq [i]$ with $|I| = k-1$ and $i \in I$ there exists
$\mathbf{c}^I \in R$ with $\mathrm{proj}_I(\mathbf{c}^I) = \mathrm{proj}_I(\mathbf{a})$, **and** there exist $\mathbf{u}, \mathbf{v} \in R$ witnessing
(i, a_i, c_i) (permitting $a_i = c_i$, in which case $\mathbf{u} = \mathbf{v} = \mathbf{c}^{[k-2] \cup \{i\}}$), **then**
Step 4.1 **choose** such $\mathbf{b}, (\mathbf{c}^I)_I, \mathbf{u}, \mathbf{v}$
Step 4.2 **return** $\{$`Interpolate`$(i, \mathbf{b}, (\mathbf{c}^I)_I, \mathbf{u}, \mathbf{v})\}$
else
Step 4.3 **return** $\emptyset$
endif

FIG. 5.2. *The* `New-Fix-values` *Algorithm*

We note that the run-time of `Interpolate` is $O(n^k)$ and hence is polynomial in n. Using the `Interpolate` algorithm we define the algorithm `New-Fix-values` as in Figure 5.2. Using `New-Fix-values`, we can define a relation (concept) from a given compact representation as follows:

> Let R be a compact representation of some n-ary relation over A. The concept *encoded* by R is defined to be the set of all $(a_1, \ldots, a_n) \in A^n$ such that `New-Fix-values`$(R, a_1, \ldots, a_n)$ returns a non-empty set.

While the output of `New-Fix-values` may be hard to pin down when its input is a compact representation of no relation that is invariant under f, we can nevertheless state some useful properties of it.

PROPOSITION 5.2. *Let R' be a compact representation of some n-ary relation R over*

Step 1 **set** $R' := \emptyset$
Step 2 **while** $EQ(R') =$ "no" **do**
let $\mathbf{a} = (a_1, \ldots, a_n)$ be the counterexample returned by EQ
Step 2.1 **if** there is some $k' < k$ and $I = \{i_1, \ldots, i_{k'}\} \subseteq [n]$ with $\mathrm{proj}_I(\mathbf{a}) \notin \mathrm{proj}_I(R')$
Step 2.1.1 **then set** $R' := R' \cup \{\mathbf{a}\}$
Step 2.2 **else**
Step 2.2.1 **compute** the least i such that `New-Fix-values`$(R', a_1, \ldots, a_i)$ returns $\emptyset$
Step 2.2.2 **select** $\mathbf{b} = (a_1, \ldots, a_{i-1}, b_i, \ldots, b_n)$ from the set returned by `New-Fix-values`$(R', a_1, \ldots, a_{i-1})$
Step 2.2.3 **set** $R' := R' \cup \{\mathbf{a}, d(\mathbf{a}, \mathbf{b})\}$
endif
endwhile
Step 3 **return** R'

FIG. 5.3. *The Learning Algorithm*

A and let $\mathbf{a} = (a_1, \ldots, a_n) \in A^n$.

1. *The concept encoded by R' is contained in $\langle R' \rangle_f$.*
2. *If R is invariant under f then $\mathbf{a} \in R$ if and only if* `New-Fix-values`$(R', \mathbf{a})$ *is non-empty. Thus, if R is invariant under f then the concept encoded by R' is equal to R.*

Proof. A careful inspection of the definitions of `Fix-values` and `New-Fix-values` should convince the reader of the correctness of statement 1. If R is invariant under f then by design, `Fix-values`$(R', a_1, \ldots, a_n)$ (and hence `New-Fix-values`$(R', a_1, \ldots, a_n)$) returns a compact representation of the subuniverse of R that consists of all elements $\mathbf{b}$ of R with $b_i = a_i$ for $1 \leq i \leq n$. Of course, this subuniverse is either empty (iff $\mathbf{a} \notin R$) or $\{\mathbf{a}\}$ and so $\mathbf{a} \in R$ if and only if `New-Fix-values` returns a non-empty set (which must be equal to $\{\mathbf{a}\}$). So, if R is invariant under f then the concept encoded by R' is equal to R. □

THEOREM 5.3. *The concept class of relations over A, encoded by compact representations, is polynomially evaluable.*

Proof. Let $R \subseteq A^n$ be a compact representation. Since the run-times of `Fix-values` and `Interpolate` can be bounded by some polynomial in n it is not hard to see that the run-time of `New-Fix-values`$(R, a_1, \ldots, a_n)$ can also be bounded by a polynomial in n, for any $\mathbf{a} \in A^n$. From this it follows that membership in the relation encoded by R can be determined in time bounded by a polynomial in n. □

In Figure 5.3 we present a learning algorithm for the concept class of all finitary relations of A invariant under f, coded by compact representations. We note that this algorithm is essentially the same as that found in [8]. In Step 2 of the algorithm, $EQ(R')$ represents the call to the oracle with hypothesis R'. If R' does not code the target concept then EQ returns a tuple from the symmetric difference of the target concept and the concept encoded by R'. It can be shown inductively that R' is always a subset of the target concept and so EQ will always return a tuple that lies in the target concept. Since in general, compact representations do not code relations invariant under f, this algorithm makes improper equivalence queries. We have:

THEOREM 5.4. *Let A be a finite set and f a k-edge operation on A. Then there exists an algorithm that exactly learns* $\mathrm{Inv}(f)$*, encoded by compact representations, with improper equivalence queries and whose run-time is bounded by a polynomial in the arity of the target relation.*

Proof. Let n be the arity of the target concept T. We will argue by induction that at any point in the execution of the learning algorithm the set R' is a compact representation of some n-ary relation contained in T and that for each pass through the while-loop, the size of R' increases by one or two. Since the size of any compact representation of an n-ary relation over A can be bounded by a polynomial in n, it follows that the run-time of the learning algorithm can be bounded by a polynomial in the arity of the target concept and that the output of the algorithm will encode the target concept.

Initially, $R' = \emptyset$, a compact representation of the empty relation. Assume that at some later stage of the execution of the learning algorithm we have that R' is a compact representation. In Step 2, if the oracle returns the answer "yes" then the algorithm has successfully learned a code for the target concept. If, instead, the oracle returns an n-tuple $\mathbf{a} = (a_1, \ldots, a_n)$ then $\mathbf{a} \in T$ (but not in the relation coded by R') and R' will be increased in one of two ways.

If some projection of $\mathbf{a}$ onto a subset I of coordinates of size $< k$ is not contained in $\mathrm{proj}_I(R')$ then $\mathbf{a}$ is added to R' and control passes again to Step 2. The new R' is still a compact representation since we have simply added an n-tuple to R' that witnesses that $\mathrm{proj}_I(\mathbf{a}) \in \mathrm{proj}_I(T)$.

If on the other hand, all projections of $\mathbf{a}$ onto small sets of coordinates are witnessed by members of R' then control passes to Step 2.2 and we compute the smallest $i \leq n$ such that `New-Fix-values`$(R', a_1, \ldots, a_i)$ returns the empty set. Such an i exists and $i > 0$ since `New-Fix-values`$(R', a_1, \ldots, a_n)$ returns the empty set and `New-Fix-values`(R') returns R'.

Since `New-Fix-values`$(R', a_1, \ldots, a_i)$ returns the empty set then one of the conditions in the **if** statement in Step 4 of the execution of `New-Fix-values`$(R', a_1, \ldots, a_i)$ must fail. Note that since we are executing Step 2.2 of the learning algorithm at this point, it must be the case that for every $I \subseteq [i]$ with $|I| = k - 1$ and $i \in I$ there exists $\mathbf{c}^I \in R$ with $\mathrm{proj}_I(\mathbf{c}^I) = \mathrm{proj}_I(\mathbf{a})$. By choice of i we know that `New-Fix-values`$(R', a_1, \ldots, a_{i-1})$ returns a non-empty set and for any $\mathbf{b}$ in this set, the element $c_i = d(a_i, b_i)$ is not equal to a_i and the minority index (i, a_i, c_i) is not witnessed in R. Thus for any selection of $\mathbf{b}$ in Step 2.2.2 of the learning algorithm we have that the minority index $(i, a_i, d(a_i, b_i))$ is not witnessed in R' and hence, in Step 2.2.3, the set $R' \cup \{\mathbf{a}, d(\mathbf{a}, \mathbf{b})\}$ is a compact representation that is strictly bigger than R' and that is contained in T. □

DEFINITION 5.5. *Let Γ be a set of finitary relations over the finite set A. The set of quantified generalized formulas over the basis Γ, denoted by $\forall\exists$-Form(Γ), is the smallest set of first-order formulas over the variables $\{x_1, x_2, \ldots\}$ that is closed under conjunction and universal and existential quantification over the x_i and that contains, for each $R \in \Gamma$ and sequence of variables $\mathbf{y}$, the formula $R(\mathbf{y})$ (where the arity of R = the length of $\mathbf{y}$).*

Using the usual semantics of first-order logic, each member Φ of $\forall\exists$-Form(Γ) defines a relation R_Φ on A. With this in mind, we also use $\forall\exists$-Form(Γ) to denote the concept class of all relations of the form R_Φ, for $\Phi \in \forall\exists$-Form(Γ), coded by Φ.

COROLLARY 5.6. *Let A be a set and f a k-edge operation on A. If Γ is a subset of $\mathrm{Inv}(f)$ then $\forall\exists$-Form(Γ) is polynomially exactly learnable with improper equivalence queries.*

Proof. Lemma 1 from [8] establishes that R_Φ is a member of $\mathrm{Inv}(f)$ for any $\Phi \in \forall\exists$-Form(Γ). From this, the result follows from the previous theorem. □

We note that Theorem 5.4 and Corollary 5.6 extend results found in [18] and [8]. They also settle one direction of a conjecture found in [16] and [14].

6. Conclusion. The study of algebras with few subpowers has led to a number of surprising algebraic results and has helped to further work on the Dichotomy Conjecture for

constraint languages. As noted earlier, our results delimit the scope of Dalmau-like algorithms for solving instances of the CSP and so new algorithms will need to be devised to deal with finite algebras that support non-trivial term operations more general than k-edge terms.

Given a finite algebra (or constraint language) there is a fairly straightforward, but inefficient algorithm to determine if it has a Mal'cev term operation (or Mal'cev polymorphism). In [20] a polynomial time algorithm to test for a Mal'cev term operation is presented that works for idempotent algebras. It is apparently much more difficult to test for the presence of a near unanimity term operation, but at least we know from a recent result of Maróti [26] that this question is decidable. Maróti's algorithm has been successfully modified by Jonah Horowitz at McMaster University to decide whether a finite algebra has an edge operation.

Question 1: Is there a polynomial time algorithm to test whether a finite algebra has a near-unanimity term operation? Has an edge term operation? Are these problems easier when restricted to idempotent algebras?

On the relational side, L. Barto has recently announced a positive solution to a conjecture of L. Zádori, by proving that a finite constraint language has a near-unanimity polymorphism if and only if it has a collection of ternary polymorphisms called *Jónsson operations*. (For the definition, see e.g. [4, Theorem 4.3].) As there are just a fixed finite number of ternary operations on a fixed finite domain, Barto's result gives an algorithm to determine whether a finite constraint language has a near-unanimity polymorphism. Inspired by Zadori's conjecture, Bova, Chen and Valeriote [5] have conjectured that a finite constraint language has an edge polymorphism if and only if it has a collection of 4-ary polymorphisms called *Day operations*. (For the definition, see e.g. [4, Theorem 4.2].) Their conjecture, if true, implies a positive answer to the following question.

Question 2: Is there an algorithm to determine, given a finite constraint language, whether it has a k-edge polymorphism for some $k > 1$?

A more direct approach to learnability than the one in Section 5 would be to use generating sets as codes for subpowers of a finite algebra, rather than compact representations. Theorem 5.4 and Corollary 5.6 would have much cleaner proofs if the following question has an affirmative answer.

Question 3: Let $\mathbf{A}$ be a finite algebra with few subpowers. Is there a polynomial time algorithm that takes as input a subset R and an n-tuple $\mathbf{a}$ from A^n for some n, and determines if $\mathbf{a}$ is in the subuniverse of $\mathbf{A}^n$ generated by R? An algebra with this property is said to be *polynomially evaluable* (see [18]).

REFERENCES

[1] D. Angluin and M. Kharitonov. When won't membership queries help? *J. Comput. System Sci.*, 50(2):336–355, 1995. 23rd Symposium on the Theory of Computing (New Orleans, LA, 1991).

[2] L. Barto and M. Kozik Constraint satisfaction problems of bounded width. To appear in FOCS'09.

[3] L. Barto, M. Kozik, and T. Niven The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). *SIAM J. Comput.*. 38:1782–1802, 2009.

[4] J. Berman, P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Varieties with few subalgebras of powers. to appear in *Trans. Amer. Math. Soc.*

[5] S. Bova, H. Chen, and M. Valeriote. Hardness results for the equivalence and isomorphism of primitive positive formulas. In preparation, 2009.

[6] A. Bulatov. Tractable conservative constraint satisfaction problems. In P. G. Kolaitis, editor, *Proceedings of the Eighteenth Annual IEEE Symp. on Logic in Computer Science, LICS 2003*, pages 321–330. IEEE Computer Society Press, June 2003.

[7] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120 (electronic), 2006.

[8] A. Bulatov, H. Chen, and V. Dalmau. Learning intersection-closed classes with signatures. to appear in Theoretical Computer Science.
[9] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27 (electronic), 2006.
[10] A. Bulatov and P. Jeavons. Algebraic structures in combinatorial problems. submitted for publication.
[11] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742 (electronic), 2005.
[12] S. Burris and H. P. Sankappanavar. *A course in universal algebra*, volume 78 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1981.
[13] C. Carvalho, V. Dalmau, P. Marković, and M. Maróti. CD(4) has bounded width. *Algebra Universalis* 60:293–307, 2009.
[14] H. Chen. The expressive rate of constraints. *Ann. Math. Artif. Intell.*, 44(4):341–352, 2005.
[15] H. Chen. Quantified constraint satisfaction and the polynomially generated powers property. 35th Colloquium on Automata, Languages and Programming (ICALP), Reykjavik, 2008.
[16] V. Dalmau. *Computational complexity of problems over generalized formulas*. PhD thesis, Universitat Politécnica de Catalunya, 2000.
[17] V. Dalmau. Generalized majority-minority operations are tractable. In P. Panangaden, editor, *Proceedings of the Twentieth Annual IEEE Symp. on Logic in Computer Science, LICS 2005*, pages 438–447. IEEE Computer Society Press, June 2005.
[18] V. Dalmau and P. Jeavons. Learnability of quantified formulas. *Theoret. Comput. Sci.*, 306(1-3):485–511, 2003.
[19] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104 (electronic), 1999.
[20] R. S. Freese and M. A. Valeriote. On the complexity of some Maltsev conditions. *Internat. J. Algebra Comput.* 19(1):41–77, 2009.
[21] P. Hell and J. Nešetřil. Colouring, constraint satisfaction, and complexity. *Comput. Sci. Rev.*, 2:143–263, 2008.
[22] P. Idziak, P. Marković, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. 22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007), pp. 213–224, 2007.
[23] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoret. Comput. Sci.*, 200(1-2):185–204, 1998.
[24] P. Jeavons, D. Cohen, and M. C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.
[25] E. Kiss and M. Valeriote. On tractability and congruence distributivity. *Logical Methods in Comput. Sci.*, 3(2:6), 20 pp. (electronic), 2007.
[26] M. Maróti. The existence of a near-unanimity term in a finite algebra is decidable. *J. Symbolic Logic* 74(3):1001-1014, 2009.
[27] R. McKenzie, G. McNulty, and W. Taylor. *Algebras, Lattices, Varieties Volume 1*. Wadsworth and Brooks/Cole, Monterey, California, 1987.
[28] T. J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978.