2002

Computational complexity of generators and nongenerators in algebra

Clifford Bergman
Iowa State University, cbergman@iastate.edu

Giora Slutzki
Iowa State University, slutzki@iastate.edu

Computational complexity of generators and nongenerators in algebra

Abstract

We discuss the computational complexity of several prob- lems concerning subsets of an algebraic structure that generate the structure. We show that the problem of determining whether a given subset X generates an algebra A is P-complete, while determining the size of the smallest generating set is NP-complete. We also consider several questions related to the Frattini subuniverse, Φ(A), of an algebra A. We show that the membership problem for Φ(A) is co-NP-complete, while the membership problems for Φ(Φ(A)), Φ(Φ(Φ(A))),... all lie in the class P (NP).

Disciplines

Algebra | Numerical Analysis and Scientific Computing

Comments

This is a manuscript of an article published as Bergman, Clifford, and Giora Slutzki. "Computational complexity of generators and nongenerators in algebra." *International Journal of Algebra and Computation* 12, no. 05 (2002): 719-735. doi: 10.1142/S0218196702001127. Posted with permission.

COMPUTATIONAL COMPLEXITY OF GENERATORS AND NONGENERATORS IN ALGEBRA

CLIFFORD BERGMAN AND GIORA SLUTZKI

ABSTRACT. We discuss the computational complexity of several problems concerning subsets of an algebraic structure that generate the structure. We show that the problem of determining whether a given subset X generates an algebra $\mathbf{A}$ is $\mathbf{P}$-complete, while determining the size of the smallest generating set is $\mathbf{NP}$-complete. We also consider several questions related to the Frattini subuniverse, $\Phi(\mathbf{A})$, of an algebra $\mathbf{A}$. We show that the membership problem for $\Phi(\mathbf{A})$ is co-$\mathbf{NP}$-complete, while the membership problems for $\Phi(\Phi(\mathbf{A}))$, $\Phi(\Phi(\Phi(\mathbf{A})))$,... all lie in the class $\mathbf{P}_{\|}(\mathbf{NP})$.

In the analysis of any algebraic structure, determining those subsets that generate the structure frequently plays a key role. This is evident in linear algebra for example, where the discussion of bases and spanning sets forms a central element of the subject. The same is true in other branches of algebra such as group and lattice theory.

Knowledge of the generating subsets of an algebra gives us information on its subalgebras, homomorphic images, automorphism group etc. As computer algebra systems become commonplace in the toolboxes of scientists (see for example GAP [11], and the "algebra calculator" [21]), questions of efficiency in the determination of generating subsets arise. In this paper we address this fundamental issue by providing completeness results for several variants of the basic question: does the subset X generate the algebra $\mathbf{A}$? In particular, we consider the question of minimal generating sets and the existence of a generating set of a given cardinality.

In addition to questions such as these, it is sometimes of interest to ask about the role an individual element can play in the generation of an algebra. Schmid [28] suggests classifying elements as *generators, nongenerators* and *irreducibles*. To explain these, we need the notion of the Frattini subuniverse of an algebra.

The Frattini subgroup has played an important role in group theory since it was first considered by Giovanni Frattini in 1885 [10]. It was observed quite some time ago that most of the basic properties of the Frattini subgroup hold more generally in any algebraic structure. Numerous papers have discussed Frattini sublattices. For a sample, see [1, 2, 9, 27]. There are

2000 *Mathematics Subject Classification.* 68Q17, 08A30, 20D25.

Key words and phrases. subalgebra, subalgebra generation, Frattini.

also papers studying the concept in the context of Moufang loops [17], closure algebras [30], Stone algebras [8], semilattices [22] and Lie algebras [3]. The study within the general framework of universal algebra was apparently initiated by Pasini [26] and continued in [5, 28].

The standard way to define the Frattini subuniverse is as the intersection of all maximal subuniverses (Definition 3.3). However Frattini's original approach was along the following lines. Let $\mathbf{A} = \langle A, F \rangle$ be an algebra and $X \subseteq A$. An element $g \in A$ is a *generator of* $\mathbf{A}$ *relative to* X if $X \cup \{g\}$ generates the entire algebra $\mathbf{A}$, while X (by itself) does not; g is called a *relative generator* if it is a generator relative to some subset X. The element g is a *nongenerator of* $\mathbf{A}$ if it is not a relative generator, i.e., it can be dropped from any set generating $\mathbf{A}$. It is not hard to show that the set of all nongenerators of $\mathbf{A}$ is equal to the Frattini subuniverse of $\mathbf{A}$. Following these same ideas further, we define an *irreducible element of* $\mathbf{A}$ to be one that must occur in every set that generates $\mathbf{A}$.

Using these notions Schmid, in [28], suggests classifying the elements of an algebra into three categories depending on the manner in which they help generate $\mathbf{A}$: **nongenerators,** elements that can be omitted from every generating set; **relative generators,** elements that play an essential role in at least one generating set; and **irreducibles,** elements that must be included in every generating set. Note that the first two of these are complements of each other and that every irreducible element is also a relative generator.

Our original goal was to study the computational complexity of various problems related to this classification. However, it quickly became apparent that before these problems could be tackled, we needed to address the complexity of two much more fundamental questions involving subalgebra generation.

- Does a given subset generate a given algebra?
- What is the size of the smallest generating set of a given (finite) algebra?

We show that these problems are complete for **P** and **NP** respectively (Theorems 2.2 and 2.6). Returning to our original questions, we prove that the problem of deciding, given an algebra $\mathbf{A}$ and element $a \in A$, whether a is a nongenerator is co-**NP**-complete; while that of deciding whether a is a relative generator is **NP**-complete (Theorem 3.5). The problem of deciding whether a is irreducible will be easily seen to be solved in deterministic logspace. In addition we consider the complexity of several other problems of interest in universal algebra: Does a given subset form a basis for a given algebra? Does a given algebra have a proper subalgebra? Does it have a proper, non-trivial subalgebra? All of these are proved complete for **P** in Theorems 2.3 and 2.4.

We begin with a brief review of the relevant background material from both universal algebra and complexity theory. Section 2 contains the general results on generating subalgebras. The third section considers the problem

of classifying an element as a relative generator, a nongenerator or an irreducible. In the final section we briefly discuss some additional problems that arise from the repeated application of the Frattini subuniverse construction. Interestingly, these problems seem to lead us into the second level of the polynomial-time hierarchy.

1. Background Material

We provide here only the barest summary of the notions we need from universal algebra and complexity theory. For more details on universal algebra, the reader should consult any of [6, 13, 23], and for computational complexity, [16, 25, 29]. Also, the first two sections of our paper [4] contain a more extensive discussion of both of these topics.

Universal algebra. For a nonnegative integer n, an *n-ary operation* on a set A is a function $f\colon A^n \to A$. The integer n is called the *rank* of f. An *algebra* is a pair $\mathbf{A} = \langle A, F\rangle$, in which A is a nonempty set, and F is a set of operations on A. The set A is called the *universe* and F the set of *basic operations* of the algebra $\mathbf{A}$. If F is finite, the algebra is said to be of *finite similarity type.*

A *subuniverse* of an algebra $\mathbf{A}$ is simply a subset of A closed under all of the basic operations of $\mathbf{A}$. We denote the collection of subuniverses of $\mathbf{A}$ by $\mathrm{Sub}(\mathbf{A})$. One easily sees that $\mathrm{Sub}(\mathbf{A})$ is always closed under arbitrary nonempty intersections. Obviously A itself is always a subuniverse. For any subset X of A we define

$$\mathrm{Sg}^{\mathbf{A}}(X) = \bigcap \{\, U \in \mathrm{Sub}(\mathbf{A}) : X \subseteq U \,\}$$

called the *subuniverse of* $\mathbf{A}$ *generated by* X. If it happens that $\mathrm{Sg}^{\mathbf{A}}(X) = A$, then we say that X is a *generating set for* $\mathbf{A}$.

For example, a group can be considered to be an algebra $\mathbf{G} = \langle G, \cdot, {}^{-1}, e\rangle$ with one binary, one unary and one nullary operation. Construed this way, the subuniverses of $\mathbf{G}$ are precisely what are normally considered to be the subgroups.

Notice that the smallest element of $\mathrm{Sub}(\mathbf{A})$ is equal to $\mathrm{Sg}^{\mathbf{A}}(\varnothing)$. This subuniverse will be empty precisely when the algebra $\mathbf{A}$ contains no nullary basic operations. A *subalgebra* of $\mathbf{A}$ is an algebra $\mathbf{B}$ of the same similarity type as $\mathbf{A}$ whose universe is a subuniverse of $\mathbf{A}$ and whose basic operations are obtained from those of $\mathbf{A}$ by restriction to B. Every subuniverse of $\mathbf{A}$ gives rise to a subalgebra *except* for the empty subuniverse (if it exists).

Computational complexity. The formal definitions of complexity theory are usually given in terms of *languages,* i.e., sets of finite strings over some fixed alphabet. Associated with each language L is a decision problem: Given a string x, decide whether $x \in L$. The amount of time or space required by a Turing machine to perform this computation generally depends on the length of the input string x. The language L is said to be computable

in *polynomial time* if there is a polynomial p such that some deterministic Turing machine can decide whether an input string x of length s lies in L in time $O\big(p(s)\big)$. The set of all languages computable in polynomial time is denoted **P**.

The set **NP** consists of those languages computable by a *nondeterministic* Turing machine whose running time lies in $O\big(p(s)\big)$, for inputs of length s. We say that such a problem is computable in *nondeterministic polynomial time.*

If L is a language over an alphabet Σ, then we denote by $(L)^{\mathrm{c}}$ the complement of L, i.e., the set of all finite strings over Σ that are not elements of L. For any complexity class **X**, the class co-**X** is equal to $\{\,(L)^{\mathrm{c}} : L \in \mathbf{X}\,\}$. It is inherent in the definition of determinism that **P** = co-**P**. However the classes **NP** and co-**NP** are believed to be different, although no proof of this is known.

Of course in practice, we prefer to couch our discussion in terms of "real" problems, rather than languages. But we always tacitly assume that there is some reasonable encoding of the instances of the problem into finite strings. In this way, we can identify our mathematical problems with formal languages, and we describe our problems as certain subsets of the set of all appropriate instances. It is common to consider a problem to be computationally feasible just in case it lies in **P**.

Given two problems A and B, we say that A is *log-space reducible* to B ($\mathrm{A} \leq_{\log} \mathrm{B}$) if there is a function f, computable in (deterministic) log-space, such that for every instance x of A, $x \in \mathrm{A} \iff f(x) \in \mathrm{B}$. B is said to be *hard for* **NP** if every member of **NP** is log-space reducible to B, and B is *complete for* **NP** if it is both hard for **NP** and a member of **NP**. It is easy to see that '$\leq_{\log}$' is reflexive and transitive. Thus, if B is known to be **NP**-complete and if $\mathrm{B} \leq_{\log} \mathrm{A} \in \mathbf{NP}$, then A is **NP**-complete as well. Similar definitions apply to **P**-hardness and **P**-completeness.

Since every deterministic Turing machine can also be considered nondeterministic, the inclusion $\mathbf{P} \subseteq \mathbf{NP}$ holds. It is generally believed, although still unproved, that the inclusion is proper. It follows from this conviction that a proof that a problem B is complete for **NP** is viewed as strong evidence that B does not belong to **P**.

We make the following assumptions regarding the format of an input instance to the problems considered in this paper. All algebras are finite and of finite similarity type. The underlying set of an algebra can be assumed to be $\{0, 1, \ldots, n-1\}$ for some positive integer n, and, in fact, this set can be represented in the input by its cardinality. This requires only $\log n$ bits of storage. Each operation of an algebra can be represented as a table of values. Thus, a k-ary operation will be represented as a k-dimensional array, with both the indices and entries coming from $\{0, \ldots, n-1\}$. An array such as this occupies $n^k \cdot \log n$ bits in the input stream.

2. Subalgebra generation

The natural starting point for any discussion of subalgebra generation is with the problem of determining the subuniverse of an algebra generated by a given subset. In order to discuss its complexity, we formulate it as a decision problem in the following way.

Problem: Gen-SubAlg
Instance: $\langle \mathbf{A}, X, a \rangle$ in which $\mathbf{A}$ is a finite algebra of finite similarity type, X is a subset of A, and a is an element of A.
Question: Is $a \in \mathrm{Sg}^{\mathbf{A}}(X)$?

In the literature, this problem is often referred to as Gen. In [18] Jones and Laaser proved the following theorem.

Theorem 2.1 (Jones and Laaser, 1977)**.** Gen-SubAlg *is complete for* $\mathbf{P}$.

Since the class $\mathbf{P}$ is closed under complements, we note that the complementary problem $(\text{Gen-SubAlg})^{\mathrm{c}}$ is also complete for $\mathbf{P}$. We now modify Gen-SubAlg to ask about generating sets.

Problem: GenSet
Instance: $\langle \mathbf{A}, X \rangle$, where $\mathbf{A}$ is a finite algebra of finite similarity type and $X \subseteq A$.
Question: Does X generate $\mathbf{A}$?

Theorem 2.2. GenSet *is complete for* $\mathbf{P}$.

Proof. We can verify that $\langle \mathbf{A}, X \rangle$ is a 'yes' instance of GenSet by checking, for each $a \in A$, that $\langle \mathbf{A}, X, a \rangle$ is a 'yes' instance of Gen-SubAlg. Since each call to Gen-SubAlg can be completed in polynomial time, and since we have only polynomially-many such calls, our algorithm for GenSet will also run in polynomial time.

To demonstrate completeness, we shall prove that Gen-SubAlg $\leq_{\log}$ GenSet. Let $\langle \mathbf{A}, X, a \rangle$ be an instance of Gen-SubAlg. For each $b \in A$ we define a new unary operation f_b on A by

$$f_b(x) = \begin{cases} b & \text{if } x = a \\ x & \text{otherwise.} \end{cases}$$

Let $\mathbf{B}$ be the algebra obtained from $\mathbf{A}$ by adding the operations $\{ f_b : b \in A \}$. Note that the universes of $\mathbf{A}$ and $\mathbf{B}$ are equal.

It is clear that this reduction requires only log-space. To complete the proof we must check that

$$\langle \mathbf{A}, X, a \rangle \in \text{Gen-SubAlg} \iff \langle \mathbf{B}, X \rangle \in \text{GenSet}. \tag{1}$$

Suppose first that $\langle \mathbf{A}, X, a \rangle \in$ Gen-SubAlg. Since $\mathbf{B}$ is an expansion of $\mathbf{A}$, the fact that $a \in \mathrm{Sg}^{\mathbf{A}}(X)$ implies $a \in \mathrm{Sg}^{\mathbf{B}}(X)$. But, for every $b \in A$, $b = f_b(a) \in \mathrm{Sg}^{\mathbf{B}}(a) \subseteq \mathrm{Sg}^{\mathbf{B}}(X)$. Thus $\langle \mathbf{B}, X \rangle \in$ GenSet.

Conversely, let $Y = \mathrm{Sg}^{\mathbf{A}}(X)$ and assume that $a \notin Y$. Then Y is a subalgebra of $\mathbf{B}$ since it is closed under the operations of $\mathbf{A}$ and for each $b \in B$ and $y \in Y$, $f_b(y) = y$. Hence, $\mathrm{Sg}^{\mathbf{B}}(X) \subseteq Y \neq A$, in other words $\langle \mathbf{B}, X \rangle \notin$ GenSet. □

In some algebraic contexts (for example, in linear algebra) it is of interest to consider generating sets that are minimal under inclusion. To make this precise, let us define a subset X of an algebra $\mathbf{A}$ to be *independent* if for every $x \in X$, $x \notin \mathrm{Sg}^{\mathbf{A}}(X - \{x\})$. By a *basis* of $\mathbf{A}$ we mean an independent, generating subset of A. Each of these gives rise to a corresponding decision problem.

Problem: IndSet, Basis
Instance: $\langle \mathbf{A}, X \rangle$, in which $\mathbf{A}$ is a finite algebra of finite similarity type, and $X \subseteq A$.
Question: (For IndSet) Is X an independent subset of $\mathbf{A}$?
Question: (For Basis) Is X a basis for $\mathbf{A}$?

Theorem 2.3. *Both* IndSet *and* Basis *are complete for* $\mathbf{P}$.

Proof. Let $\langle \mathbf{A}, X \rangle$ be an instance of either of these problems. For each $x \in X$, the test $x \in \mathrm{Sg}^{\mathbf{A}}(X - \{x\})$ can be computed in polynomial time, according to Theorem 2.1. To check the independence of X requires at most $|A|$ many such tests, which can therefore be done in polynomial time. Since Basis is the conjunction of the problems GenSet and IndSet, it too lies in $\mathbf{P}$.

To show completeness, we use the following construction. Let $\mathbf{A} = \langle A, f_1, f_2, \ldots, f_q \rangle$ be an algebra. We define an algebra $\mathbf{A}^{(2)} = \langle B, g_1, \ldots, g_q \rangle$ of the same similarity type as $\mathbf{A}$ as follows. The universe B of $\mathbf{A}^{(2)}$ is $A \times \{0, 1\}$. For clarity, we shall denote an element $(x, i) \in B$ by x^i. For each $j = 1, \ldots, q$, we define the operation g_j by

$$g_j(x_1^{i_1}, x_2^{i_2}, \ldots, x_r^{i_r}) = f_j(x_1, x_2, \ldots, x_r)^1.$$

To show the $\mathbf{P}$-completeness of IndSet, we shall reduce (Gen-SubAlg)$^{\mathrm{c}}$ to IndSet. Let $\langle \mathbf{A}, X, a \rangle$ be an instance of Gen-SubAlg. Set $Y = (X \times \{0\}) \cup \{a^1\}$ and consider it as a subset of $\mathbf{A}^{(2)}$. It is easy to see that

$$a \notin \mathrm{Sg}^{\mathbf{A}}(X) \iff \langle \mathbf{A}^{(2)}, Y \rangle \in \text{IndSet},$$

thus (Gen-SubAlg)$^{\mathrm{c}} \leq_{\log}$ IndSet.

Using a similar construction, we can reduce GenSet to Basis. Let $\langle \mathbf{A}, X \rangle$ be an instance of GenSet. Let $\mathbf{B}'$ be the subalgebra of $\mathbf{A}^{(2)}$ with universe $(X \times \{0\}) \cup (A \times \{1\})$, and take Y to be $X \times \{0\}$. Notice that Y is an independent set. Then we obviously have

$$\langle A, X \rangle \in \text{GenSet} \iff \langle \mathbf{B}', Y \rangle \in \text{Basis}.$$

Therefore GenSet $\leq_{\log}$ Basis. □

In universal algebra, it is often important to determine whether an algebra has any proper subalgebras. For example, a necessary condition for an algebra to be primal (see [20]) is that it have no proper subalgebras. Sometimes, we are only interested in the presence or absence of proper *nontrivial* subalgebras. This suggests the following.

Problem: PROPER-SUB, PROPER-NONTRIV-SUB
Instance: $\mathbf{A}$, a finite algebra of finite similarity type.
Question: (For PROPER-SUB) Does $\mathbf{A}$ have a proper subalgebra?
Question: (For PROPER-NONTRIV-SUB) Does $\mathbf{A}$ have a proper, nontrivial subalgebra?

Theorem 2.4. PROPER-SUB *and* PROPER-NONTRIV-SUB *are both complete for* $\mathbf{P}$.

Proof. It is easy to see that $\mathbf{A} \in$ PROPER-SUB if and only if for some $a \in A$, $\langle \mathbf{A}, \{a\} \rangle \notin$ GENSET. Similarly, $\mathbf{A} \in$ PROPER-NONTRIV-SUB if and only if for some pair $\{a, b\}$ of distinct elements of A, $\langle \mathbf{A}, \{a, b\} \rangle \notin$ GENSET. From these two observations it follows that both PROPER-SUB and PROPER-NONTRIV-SUB lie in $\mathbf{P}$.

For completeness, we first reduce $(\text{GENSET})^c$ to PROPER-SUB. Let $\langle \mathbf{A}, X \rangle$ be an instance of GENSET. Add to $\mathbf{A}$ a nullary operation for each element of X. Call the resulting algebra $\mathbf{B}$. Then we have

$$\mathrm{Sg}^{\mathbf{A}}(X) \neq A \iff \mathbf{B} \text{ has a proper subalgebra.}$$

From the remark following Theorem 2.1, we conclude that PROPER-SUB is complete for $\mathbf{P}$. Finally, we can reduce PROPER-SUB to PROPER-NONTRIV-SUB as follows. Let $\mathbf{A} = \langle A, f_0, \ldots, f_{r-1} \rangle$ be an algebra. Choose a new element $\diamondsuit$ and let $A(\diamondsuit) = A \cup \{\diamondsuit\}$. For each $i < r$ define f_i' on $A(\diamondsuit)$ by

$$f_i'(x_1, \ldots, x_k) = \begin{cases} \diamondsuit & \text{if } \diamondsuit \in \{x_1, \ldots, x_k\} \\ f_i(x_1, \ldots, x_k) & \text{otherwise.} \end{cases}$$

Now define the algebra $\mathbf{A}(\diamondsuit) = \langle A(\diamondsuit), f_0', \ldots, f_{r-1}', \diamondsuit \rangle$. (Note the addition of a nullary operation symbol for $\diamondsuit$.) It is easy to see that the subuniverses of $\mathbf{A}(\diamondsuit)$ are precisely the sets $B \cup \{\diamondsuit\}$ for $B \in \mathrm{Sub}(\mathbf{A})$. Thus $\mathbf{A}$ has a proper subalgebra if and only if $\mathbf{A}(\diamondsuit)$ has a proper nontrivial subalgebra. We conclude that PROPER-SUB $\leq_{\log}$ PROPER-NONTRIV-SUB. □

We now turn to the the second fundamental problem on our list, determining, for a given integer k, whether an algebra has a generating set of cardinality k.

Problem: k-GEN
Instance: $\langle \mathbf{A}, k \rangle$, in which $\mathbf{A}$ is a finite algebra of finite similarity type, and k is a natural number.
Question: Is there a subset X of A with $|X| \leq k$ and $\mathrm{Sg}^{\mathbf{A}}(X) = A$?

We shall prove that k-GEN is **NP**-complete by reduction from the well-known problem Exact 3-Cover.

Problem: X3C
Instance: $\langle Y, C\rangle$, in which Y is a set of cardinality $3q$ and C is a set of 3-element subsets of Y.
Question: Does C contain a subset D that forms a partition (i.e., an exact 3-cover) of Y?

Theorem 2.5 (Karp, see [12, pg. 53])**.** X3C *is* **NP***-complete.*

Theorem 2.6. k-GEN *is complete for* **NP***.*

Proof. Certainly, we can guess a subset X of cardinality k and test, in polynomial time, whether $\langle \mathbf{A}, X\rangle \in$ GENSET. Thus k-GEN lies in **NP**. To prove completeness, we shall show that X3C $\leq_{\log} k$-GEN.

Let $\langle Y, C\rangle$ be an instance of X3C, with $|Y| = 3q$. Without loss of generality, assume that Y and C are disjoint. We describe a pair $\langle \mathbf{A}, k\rangle$ such that

$$\langle Y, C\rangle \in \text{X3C} \iff \langle \mathbf{A}, k\rangle \in k\text{-GEN}. \tag{2}$$

We take $\mathbf{A}$ to have universe $Y \cup C$ and operations f_1, f_2, f_3, g of ranks $1, 1, 1, 3$ respectively, and set $k = q$. Enumerate the elements of C as $\mathbf{c}_1, \ldots, \mathbf{c}_m$ and for each $i \leq m$, enumerate $\mathbf{c}_i = \{c_{i1}, c_{i2}, c_{i3}\}$. We define the operations as follows.

$$\begin{matrix} f_j(\mathbf{c}_i) = c_{ij} & \text{for } i \leq m, \\ f_j(y) = y & \text{for } y \in Y, \end{matrix} \quad \text{for } j = 1, 2, 3;$$

$$g(u, v, w) = \begin{cases} \mathbf{c}_i & \text{if } \{u, v, w\} = \mathbf{c}_i, \\ u & \text{otherwise.} \end{cases}$$

To verify the equivalence in (2) we show that the generating subsets of $\mathbf{A}$ of cardinality at most q are precisely the exact 3-covers of Y contained in C. Suppose first that D is an exact 3-cover. Obviously $|D| = q$. By applying the operations f_1, f_2, f_3 to the elements of D, we obtain all elements of Y. Then applying g to the elements of Y yields every element of C. Thus D is a generating set of cardinality q.

Conversely, let X be a subset of A and let B be the subuniverse of $\mathbf{A}$ generated by X. Then for any point y of Y, $y \in B$ if and only if either $y \in X$ or for some $\mathbf{c} \in X$, $y \in \mathbf{c}$. It follows that

$$|Y \cap B| \leq |Y \cap X| + 3\,|C \cap X|\,. \tag{3}$$

Now let X be a generating set of $\mathbf{A}$ of cardinality at most q. Then $|Y \cap B| = |Y| = 3q$ and it follows from (3) that $|Y \cap X| = 0$ and $|C \cap X| = q$, in other words, X is an exact 3-cover.

Finally, let us indicate why this reduction can be accomplished in log-space. The algorithm must output the operation tables for the f_j's and g. Each "row" in each table can be determined by a single pass through the

input, searching either for the specification of $\mathbf{c}_i$ (to compute $f_j(\mathbf{c}_i)$) or for some subset $\mathbf{c}$ that matches $\{u, v, w\}$ to compute the value of g. □

3. Classification of elements

We now turn to the problems of classifying an element of an algebra as a nongenerator, a relative generator, or an irreducible.

Definition 3.1. Let $\mathbf{A}$ be an algebra, X a subset of A and a an element of A. We call a a *relative generator of* $\mathbf{A}$ *with respect to* X if $a \notin \mathrm{Sg}^{\mathbf{A}}(X)$ and $\mathrm{Sg}^{\mathbf{A}}(X \cup \{a\}) = A$.

Associated with this notion is a natural decision problem.

Problem: Rel-Gen-wrt
Instance: $\langle \mathbf{A}, X, a \rangle$, in which $\mathbf{A}$ is a finite algebra of finite type, $X \subseteq A$ and $a \in A$.
Question: Is a a relative generator of $\mathbf{A}$ with respect to X?

We wish to emphasize that this problem does not ask whether a is a relative generator of $\mathbf{A}$, but only whether it is a relative generator of $\mathbf{A}$ *with respect to* X.

Proposition 3.2. Rel-Gen-wrt *is complete for* $\mathbf{P}$.

Proof. The condition $\langle \mathbf{A}, X, a \rangle \in$ Rel-Gen-wrt is equivalent to the conjunction of the conditions $\langle \mathbf{A}, X, a \rangle \notin$ Gen-SubAlg and $\langle \mathbf{A}, X \cup \{a\} \rangle \in$ GenSet. Each of these latter two can be computed in polynomial time, so Rel-Gen-wrt $\in \mathbf{P}$.

For the converse, we can reduce GenSet to Rel-Gen-wrt as follows. Let $\langle \mathbf{A}, X \rangle$ be an instance of GenSet. Recall the definition of the algebra $\mathbf{A}(\diamondsuit)$ given in Theorem 2.4. Let $\mathbf{B}$ be the same algebra as $\mathbf{A}(\diamondsuit)$, except that this time we do not include the nullary operation symbol whose value is $\diamondsuit$.

Notice that $\mathbf{A}$ is a subalgebra of $\mathbf{B}$ containing X. It follows that $\diamondsuit \notin \mathrm{Sg}^{\mathbf{B}}(X)$. It is easy to see that $\mathrm{Sg}^{\mathbf{A}}(X) = A$ if and only if $\mathrm{Sg}^{\mathbf{B}}(X \cup \{\diamondsuit\}) = B$, that is, X generates $\mathbf{A}$ if and only if $\diamondsuit$ is a generator of $\mathbf{B}$ relative to X. □

As we mentioned in the introduction, an element a of an algebra $\mathbf{A}$ is called a *relative generator* of $\mathbf{A}$ if it is a generator relative to some subset X as in Definition 3.1. If a is not a relative generator, then we call it a *nongenerator* of $\mathbf{A}$. The original motivation for studying nongenerators comes from the Frattini subuniverse. Although Frattini worked only with groups, the idea generalizes naturally to any algebraic structure.

Definition 3.3. Let $\mathbf{A}$ be an algebra. The *Frattini subuniverse* of $\mathbf{A}$, denoted $\Phi(\mathbf{A})$ is the intersection of all maximal proper subuniverses of $\mathbf{A}$.

It is possible for $\Phi(\mathbf{A})$ to be empty. This is why we call it a subuniverse. However, it is common to refer to $\Phi(\mathbf{A})$ as the Frattini subalgebra of $\mathbf{A}$. In any case, as the intersection of subuniverses, $\Phi(\mathbf{A})$ is always a subuniverse of $\mathbf{A}$. Note that being the intersection of *all* maximal proper subuniverses,

$\Phi(\mathbf{A})$ is preserved by every automorphism of $\mathbf{A}$. In the specific case that $\mathbf{A}$ is a group, it follows that $\Phi(\mathbf{A})$ will be a normal subgroup. This helps to explain why the Frattini subgroup has played an important role in group theory. (In fact, the Frattini subgroup of a finite group is nilpotent, but this does not seem to have an analog for general algebras.)

The next proposition describes the relationship between nongenerators and the Frattini subuniverse. It is not difficult to prove, but does require Zorn's lemma for infinite algebras. A proof can be found in [19].

Proposition 3.4. *For any algebra* $\mathbf{A}$*,* $\Phi(\mathbf{A})$ *is exactly the set of all nongenerators of* $\mathbf{A}$*.*

We now return to our original classification problem. Given an algebra $\mathbf{A}$ and an element d, is d a relative generator or is it a nongenerator of $\mathbf{A}$?

Problem: Rel-Gen
Instance: $\langle \mathbf{A}, d \rangle$, where $\mathbf{A}$ is a finite algebra of finite similarity type, and d is an element of A.
Question: Is d a relative generator of $\mathbf{A}$?

Theorem 3.5. Rel-Gen *is complete for* **NP**.

Proof. It is easy to see that Rel-Gen lies in **NP**. After all, one can nondeterministically choose a subset X and verify that $\langle \mathbf{A}, X, d \rangle \in$ Rel-Gen-wrt. To prove completeness, we shall reduce X3C to Rel-Gen and apply Theorem 2.5.

Let $\langle Y, C \rangle$ be an instance of X3C, where Y is a set of cardinality q and $C = \{c_0, c_1, \ldots, c_{m-1}\}$. We construct an algebra $\mathbf{A}$ as follows. $A = C \cup \{g\}$ for a new element g. We endow A with basic operations $f, h_0, h_1, \ldots, h_{m-1}$ given by

$$f(x,y) = \begin{cases} g & \text{if } x, y \in C \text{ and } x \cap y \neq \varnothing, \\ x & \text{otherwise;} \end{cases}$$

$$h_i(x,y,z,w) = \begin{cases} c_i & \text{if } x = g,\ y, z, w \in C \text{ and } c_i \subseteq y \cup z \cup w, \\ x & \text{otherwise.} \end{cases}$$

We claim that $\langle Y, C \rangle$ has an exact subcover if and only if g is a generator of $\mathbf{A}$ relative to some set X. Suppose D is an exact subcover. Let $X = D$. Then X is a proper subuniverse of $\mathbf{A}$ (all operations reduce to the first projection). Since D covers Y, $X \cup \{g\}$ generates all of A. Thus g is a generator relative to X.

Conversely, let g be a generator relative to a subset X. Let $D = \mathrm{Sg}^{\mathbf{A}}(X)$. Then $g \notin D$. Therefore, D must consist of pairwise-disjoint elements of C. On the other hand, since $D \cup \{g\}$ generates A, for each $i < n$, either $c_i \in D$ or there are $c, c', c'' \in D$ such that $c_i = h_i(g, c, c', c'')$, which certainly implies that $c_i \subseteq \bigcup D$. Hence D is an exact cover. $\square$

The membership problem for the Frattini subuniverse is precisely the complement of the problem Rel-Gen. Thus, if we define Φ-Mem to be (Rel-Gen)^c we have the following corollary.

Corollary 3.6. Φ-Mem *is* co-**NP**-*complete.*

We include one further problem involving the notion of a relative generator.

Problem: k-Rel-Gen
Instance: $\langle \mathbf{A}, a, k\rangle$, in which $\mathbf{A}$ is a finite algebra of finite type, $a \in A$ and k is a natural number.
Question: Does there exist a subset X of A such that $|X| \leq k$ and a is a generator of $\mathbf{A}$ relative to X?

Proposition 3.7. k-Rel-Gen *is* **NP**-*complete.*

Proof. Since we can guess an X and check (in polynomial time) the condition $\langle \mathbf{A}, X, a\rangle \in$ Rel-Gen-wrt, we see that k-Rel-Gen $\in$ **NP**. We can prove that k-Gen $\leq_{\log}$ k-Rel-Gen using the same reduction that was used in Proposition 3.2. □

Let us briefly discuss irreducible elements. Recall that an element a of an algebra $\mathbf{A}$ is irreducible if $\mathrm{Sg}^{\mathbf{A}}(X) = A$ implies $a \in X$, for every subset X. This is obviously quite a strong condition on an element. As an example, the free generators of a free lattice are irreducible, but the free generators of a free group are not. The following proposition is easy to verify (or see [5, Prop. 1.2]).

Proposition 3.8. *Let* $\mathbf{A}$ *be an algebra and* a *an element of* A*. The following are equivalent.*

(1) a *is irreducible in* $\mathbf{A}$.
(2) $A - \{a\}$ *is a subuniverse of* $\mathbf{A}$.
(3) *For every basic operation* f *and* $b_1, \ldots, b_k \in A$, $f(b_1, \ldots, b_k) = a$ *implies that* $a \in \{b_1, \ldots, b_k\}$.

Based on this proposition, it is easy to develop an algorithm that will take as input an algebra $\mathbf{A}$ and an element a and determine whether a is irreducible. Simply check each "row" in each operation table to verify that condition (3) of 3.8 holds. Such a procedure can clearly be accomplished in deterministic log-space. We state this as a proposition.

Problem: Irr
Instance: $\langle \mathbf{A}, a\rangle$ in which $\mathbf{A}$ is a finite algebra of finite type and $a \in A$.
Question: Is a an irreducible element of $\mathbf{A}$?

Proposition 3.9. *The problem* Irr *lies in deterministic log-space.*

4. Iterating the Frattini construction

Let us define, for an algebra $\mathbf{A}$,

$$C(\mathbf{A}) = \bigcap \mathrm{Sub}(\mathbf{A})$$

the smallest subuniverse of $\mathbf{A}$. Note that $C(\mathbf{A})$ is equal to the subuniverse generated by those elements of the algebra that are the value of some nullary basic operation. In particular, $C(\mathbf{A}) = \varnothing$ if and only if $\mathbf{A}$ has no nullary basic operations. Note also, that for any subalgebra $\mathbf{B}$ of $\mathbf{A}$, $C(\mathbf{B}) = C(\mathbf{A})$.

It is possible to iterate the Frattini construction and obtain a descending chain of subuniverses. More precisely, we define

$$\begin{aligned} \Phi^0(\mathbf{A}) &= A \\ \Phi^{n+1}(\mathbf{A}) &= \Phi(\Phi^n(\mathbf{A})) \qquad \text{for any natural number } n. \end{aligned}$$

(What we have written is not technically correct, since $\Phi(\mathbf{A})$ is defined to be a sub*universe* of $\mathbf{A}$ rather than a sub*algebra*. But it should be clear what we have in mind. If, for some n, $\Phi^n(\mathbf{A})$ is empty, then $\Phi^{n+1}(\mathbf{A})$ will be taken to be empty as well.)

Proposition 4.1. *Let* $\mathbf{A}$ *be an algebra and assume that* $\mathrm{Sub}(\mathbf{A})$ *is a lattice of finite height. Then for some natural number* n, $\Phi^n(\mathbf{A}) = C(\mathbf{A})$. *In particular,* $\Phi(\mathbf{A}) = A$ *if and only if* $C(\mathbf{A}) = A$.

Proof. By induction on the height of $\mathrm{Sub}(\mathbf{A})$. The height is 0 exactly when $\mathrm{Sub}(\mathbf{A}) = \{A\}$. Equivalently, $C(\mathbf{A}) = A = \Phi(\mathbf{A})$. On the other hand, if the height of $\mathrm{Sub}(\mathbf{A})$ is positive, then $\mathbf{A}$ must have at least one proper subalgebra (namely $C(\mathbf{A})$). Since $\mathrm{Sub}(\mathbf{A})$ has finite height, $\mathbf{A}$ must have at least one maximal proper subuniverse. Hence $C(\mathbf{A}) \subseteq \Phi(\mathbf{A}) \subsetneq A$. If $C(\mathbf{A}) = \Phi(\mathbf{A})$ then we are done. If not, then we can apply the induction hypothesis to obtain an integer n such that $\Phi^n(\Phi(\mathbf{A})) = C(\Phi(\mathbf{A})) = C(\mathbf{A})$. □

Of course, any finite algebra satisfies the conditions of the above proposition. Since the idea of iterating the Frattini construction appears to be new, it is not clear whether these concepts have further applications in algebra. Except for Proposition 4.1, we have not pursued these ideas. However, from the point of view of compuational complexity, the membership problem for these derived subuniverses may be of interest since it seems to take us further into the polynomial-time hierarchy, see equations (4) below for the definitions and [24, 25, 29, 33] for a complete discussion of the polynomial-time hierarchy.

Problem: Φ^n-Mem (n a fixed positive integer)
Instance: $\langle \mathbf{A}, a \rangle$ in which $\mathbf{A}$ is a finite algebra of finite similarity type and $a \in A$.
Question: Is $a \in \Phi^n(\mathbf{A})$?

Not suprisingly, the complexity of Φ^n-MEM increases with n, as we now show.

Lemma 4.2. *For every positive integer n, Φ^n-MEM $\leq_{\log}$ Φ^{n+1}-MEM.*

Proof. Once again we utilize the construction of $\mathbf{A}(\diamondsuit)$ given in Theorem 2.4. However this time we omit the nullary operation with value $\diamondsuit$ and instead add, for each $a \in A$ a unary operation h_a given by

$$h_a(x) = \begin{cases} a & \text{if } x = \diamondsuit, \\ x & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathrm{Sub}\big(\mathbf{A}(\diamondsuit)\big) = \mathrm{Sub}(\mathbf{A}) \cup \{A(\diamondsuit)\}$. It follows from Definition 3.3 that $\Phi\big(\mathbf{A}(\diamondsuit)\big) = A$. In this way we obtain a reduction of Φ^n-MEM to Φ^{n+1}-MEM using the equivalence

$$\langle \mathbf{A}, a\rangle \in \Phi^n\text{-MEM} \iff \langle \mathbf{A}(\diamondsuit), a\rangle \in \Phi^{n+1}\text{-MEM}.$$

□

In order to discuss the complexity of Φ^n-MEM, we require the notion of an oracle Turing machine. For any problem L, we denote by $\mathbf{P}(\mathrm{L})$ the set of problems solvable in polynomial time by a deterministic Turing machine with an oracle for L. There is an analogous set, $\mathbf{NP}(\mathrm{L})$, defined in terms of nondeterministic oracle Turing machines. Informally, one can imagine a computer program that is allowed to make calls to a subroutine which computes the answer to the decision problem L. Each subroutine call counts as one instruction step. According to the usual definition, the calls to the oracle are permitted to be *adaptive,* that is, the Turing machine's second query to the oracle may be based on the result of the first query. See any of [12, 16, 25, 29] for a full discussion of oracle Turing machines.

For a set $\mathbf{C}$ of languages, define $\mathbf{P}(\mathbf{C}) = \bigcup_{\mathrm{L}\in\mathbf{C}} \mathbf{P}(\mathrm{L})$ and $\mathbf{NP}(\mathbf{C}) = \bigcup_{\mathrm{L}\in\mathbf{C}} \mathbf{NP}(\mathrm{L})$. The *polynomial-time hierarchy* is defined recursively by the formulas (for all $n > 0$)

$$\begin{gathered} \Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathbf{P} \\ \Delta_{n+1}^p = \mathbf{P}(\Sigma_n^p), \quad \Sigma_{n+1}^p = \mathbf{NP}(\Sigma_n^p), \quad \Pi_{n+1}^p = \text{co-}\Sigma_{n+1}^p. \end{gathered} \tag{4}$$

Note that $\Delta_1^p = \mathbf{P}$, $\Sigma_1^p = \mathbf{NP}$ and $\Pi_1^p = \text{co-}\mathbf{NP}$. It is easy to show that Φ^n-MEM $\in \Delta_2^p$. We now discuss an improvement on that upper bound.

Let us tighten the restrictions on our oracle Turing machines by requiring that all of the queries to the oracle be made in parallel. In other words, the formulation of every query must be finished before the results of any of the queries are known. The class $\mathbf{P}_{\|}(\mathrm{L})$ denotes the set of problems solvable by a deterministic Turing machine utilizing the oracle L in such a nonadaptive manner. We write $\mathrm{L}' \leq_{\mathrm{tt}} \mathrm{L}$ to indicate that $\mathrm{L}' \in \mathbf{P}_{\|}(\mathrm{L})$. It is not hard to show that '$\leq_{\mathrm{tt}}$' is transitive. Thus, if $\mathrm{L}' \leq_{\mathrm{tt}} \mathrm{L}$, then $\mathbf{P}_{\|}(\mathrm{L}') \subseteq \mathbf{P}_{\|}(\mathrm{L})$. We denote by $\mathbf{P}_{\|}(\mathbf{NP})$ the class $\bigcup_{L\in\mathbf{NP}} \mathbf{P}_{\|}(\mathrm{L})$. It is easy

to see that $\mathbf{NP} \cup \text{co-}\mathbf{NP} \subseteq \mathbf{P}_{\|}(\mathbf{NP}) \subseteq \Delta_2^p$. See Papadimitriou [25, chap. 17] for a more complete discussion of the class $\mathbf{P}_{\|}(\mathbf{NP})$.

The class $\mathbf{P}_{\|}(\mathbf{NP})$ has several alternate characterizations and notations: $\mathbf{P}_{\log n}(\mathbf{NP})$, $\mathbf{P}_{\text{tt}}(\mathbf{NP})$ and $\Theta_2(\mathbf{P})$, among others. The equivalence of $\mathbf{P}_{\|}(\mathbf{NP})$ with each of these is proved by Buss and Hay in [7] (see also [14, 31, 32]), who also provided the following very useful characterization: A problem L lies in $\mathbf{P}_{\|}(\mathbf{NP})$ if and only if for some fixed natural number n, a deterministic Turing machine can solve L by making n rounds of parallel queries to an oracle for some **NP**-complete problem. Here, the queries in the later rounds may depend on answers to queries in earlier rounds; i.e., intuitively $\mathbf{P}_{\|}(\mathbf{NP})$ allows a finite, but fixed, amount of adaptivity.

As an application, given an algebra $\mathbf{A}$, one can use one round of parallel queries to Rel-Gen to determine all of the members of $\Phi(\mathbf{A})$. Once we know the Frattini subalgebra, we can use a second round of queries to determine the members of $\Phi^2(\mathbf{A})$. A third round can then be used to determine $\Phi^3(\mathbf{A})$, etc. This proves the following proposition.

Proposition 4.3. *For every positive integer n, Φ^n-Mem $\in \mathbf{P}_{\|}(\mathbf{NP})$.*

Using methods similar to those of Buss and Hay, J. Hitchcock [15] proved that for every $n \geq 2$, Φ^n-Mem $\leq_{\text{tt}}$ Φ^2-Mem. Put another way, Φ^n-Mem $\in \mathbf{P}_{\|}(\Phi^2\text{-Mem})$. Coupled with Lemma 4.2, it follows that all of the problems Φ^2-Mem, Φ^3-Mem, ... are equivalent under truth-table reductions (i.e., the relation '$\leq_{\text{tt}}$'), indeed, they are all truth-table equvialent to any **NP**-complete problem.

The definition of Φ^n implies that for any finite algebra $\mathbf{A}$ we have a strictly descending chain

$$A = \Phi^0(\mathbf{A}) \supset \Phi^1(\mathbf{A}) \supset \Phi^2(\mathbf{A}) \supset \cdots \supset C(\mathbf{A}).$$

That the sequence always terminates at $C(\mathbf{A})$ is the content of Proposition 4.1. The smallest n such that $\Phi^n(\mathbf{A}) = C(\mathbf{A})$, which we call the *Frattini index* of $\mathbf{A}$, is an invariant of $\mathbf{A}$. This notion provides another approach to the complexity of the iterated Frattini construction.

Problem: Φ-Index
Instance: $\langle \mathbf{A}, n \rangle$, in which $\mathbf{A}$ is a finite algebra of finite similarity type and n is a positive integer.
Question: Is $\Phi^{n-1}(\mathbf{A}) \supset \Phi^n(\mathbf{A}) = C(\mathbf{A})$?

The precise complexity of Φ-Index is unknown, however we do have the following upper bound.

Proposition 4.4. Φ-Index $\in \mathbf{P}_{\|}(\mathbf{NP})$.

Proof. Let $\langle \mathbf{A}, n \rangle$ be an instance of Φ-Index. We can, in polynomial-time, determine all members of A that are the values of a nullary operation. Then

by Theorem 2.1, the members of $C(\mathbf{A})$ can be determined in polynomial-time. Observe that

$$(5) \quad \langle \mathbf{A}, n\rangle \in \Phi\text{-INDEX} \iff \bigwedge_{a\in A-C(\mathbf{A})} a \notin \Phi^n(\mathbf{A}) \wedge \bigvee_{a\in A-C(\mathbf{A})} a \in \Phi^{n-1}(\mathbf{A}).$$

We can use a single round of parallel queries to oracles for Φ^{n-1}-MEM and Φ^n-MEM to determine whether the right-hand side of equivalence (5) holds. The result then follows Proposition 4.3 and the transitivity of '$\leq_{\mathrm{tt}}$'. □

Finding the exact computational complexity of Φ^n-MEM and Φ-INDEX seems to us to be an intriguing problem left open in this paper. We formulate it as a conjecture.

Conjecture. For any $n > 1$, both Φ^n-MEM and Φ-INDEX are complete for $\mathbf{P}_{\|}(\mathbf{NP})$.

We would like to thank John Hitchcock for help in clarifying the complexity-theoretic concepts in Section 4.

REFERENCES

1. M. Adams, P. Dwinger, and J. Schmid, *Maximal sublattices of finite distributive lattices*, Algebra Universalis **36** (1996), no. 4, 488–504.
2. M. Adams, R. Freese, J. Nation, and J. Schmid, *Maximal sublattices and Frattini sublattices of bounded lattices*, J. Austral. Math. Soc. Ser. A **63** (1997), no. 1, 110–127.
3. Yu. A. Bakhturin, *Identities in Lie algebras (Russian)*, "Nauka", Moscow, 1985.
4. C. Bergman and G. Slutzki, *Complexity of some problems concerning varieties and quasivarieties of algebras*, SIAM J. Comput. **30** (2000), no. 2, 359–382.
5. J. Berman and G. H. Bordalo, *Irreducible elements and uniquely generated algebras*, preprint, 1999.
6. S. Burris and H. P. Sankappanavar, *A course in universal algebra*, Springer-Verlag, New York, 1981.
7. S. R. Buss and L. Hay, *On truth-table reducibility to SAT*, Inform. and Comput. **91** (1991), no. 1, 86–102.
8. C. Chen, K. Koh, and K. Teo, *On the length of the lattice of subalgebras of a finite Stone algebra*, Bull. Malaysian Math. Soc. **5** (1982), 101–104.
9. C. C. Chen and K. M. Koh, *An algorithm for determining $F(L)$ in finite distributive lattices*, Algebra Universalis **8** (1978), no. 2, 151–158.
10. G. Frattini, *Intorno alla generazione dei gruppi di operazioni*, Rend. Atti. Accad. Lincei **4** (1885), no. 1, 281–285, 455–457.
11. The GAP Group, Aachen, St Andrews, *GAP – Groups, Algorithms, and Programming, Version 4.2*, 2000, (`http://www-gap.dcs.st-and.ac.uk/~gap`).
12. M. Garey and D. Johnson, *Computers and intractability—a guide to the theory of NP-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.
13. G. Grätzer, *Universal algebra*, second ed., Springer-Verlag, New York, 1979.
14. L. A. Hemachandra, *The strong exponential hierarchy collapses*, J. Comput. System Sci. **39** (1989), 299–322.
15. J. Hitchcock, private communication.
16. J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, MA, 1979.

17. T. Hsu, *Moufang loops of class* 2 *and cubic forms*, Math. Proc. Cambridge Philos. Soc. **128** (2000), no. 2, 197–222.
18. N. D. Jones and W. T. Laaser, *Complete problems for deterministic polynomial time*, Theoret. Comput. Sci. **3** (1977), 105–117.
19. B. Jónsson, *Topics in universal algebra*, Lecture Notes in Math., vol. 250, Springer-Verlag, Berlin, Heidelberg, New York, 1972.
20. K. Kaarli and A. Pixley, *Polynomial completeness in universal algebra*, Chapman & Hall/CRC Press, 2000.
21. E. Kiss, *Algebra calculator program*, 2000, (`http://www.cs.elte.hu/~ewkiss/software/uaprog/uaprog.html`).
22. K. Koh, *On the Frattini sub-semilattice of a semilattice*, Nanta Math. **5** (1971), no. 1, 22–33.
23. R. McKenzie, G. McNulty, and W. Taylor, *Algebras, lattices, varieties*, vol. I, Wadsworth & Brooks/Cole, Belmont, CA, 1987.
24. A. Meyer and L. Stockmeyer, *The equivalence problem for regular expressions with squaring requires exponential time*, Proc. 13th Ann. IEEE Symp. on Switching and Automata Theory, 1972.
25. C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, MA, 1994.
26. A. Pasini, *On the Frattini subalgebra* $F(A)$ *of an algebra* A, Boll. Un. Mat. Ital. (4) **12** (1975), no. 1-2, 37–40.
27. Ch. Ryter and J. Schmid, *Deciding Frattini is NP-complete*, Order **11** (1994), no. 3, 257–279.
28. J. Schmid, *Nongenerators, genuine generators and irreducibles*, Houston J. Math. **25** (1999), no. 3, 405–416.
29. M. Sipser, *Introduction to the theory of computation*, PWS Publishing Company, Boston, MA, 1997.
30. L. Vrancken-Mawet, *Sous-algèbres de Frattini d'algèbres de fermeture*, Bull. Soc. Math. Belg. Sér. B **39** (1987), no. 1, 33–45.
31. K. W. Wagner, *On restricting the access to an NP-oracle*, Automata, languages and programming (Tampere, 1988), Springer, Berlin, 1988, pp. 682–696.
32. ———, *Bounded query classes*, SIAM J. Comput. **19** (1990), no. 5, 833–846.
33. C. Wrathall, *Complete sets and the polynomial-time hierarchy*, Theoret. Comput. Sci. **3** (1976), no. 1, 23–33 (1977).

Dept. of Mathematics, Iowa State University, Ames, Iowa 50011, USA
E-mail address: `cbergman@iastate.edu`

Dept. of Computer Science, Iowa State University, Ames, Iowa 50011, USA
E-mail address: `slutzki@cs.iastate.edu`