SIAM J. COMPUT.
Vol. 30, No. 2, pp. 359–382

COMPLEXITY OF SOME PROBLEMS CONCERNING VARIETIES AND QUASI-VARIETIES OF ALGEBRAS*

CLIFFORD BERGMAN† AND GIORA SLUTZKI‡

Abstract. In this paper we consider the complexity of several problems involving finite algebraic structures. Given finite algebras **A** and **B**, these problems ask the following. (1) Do **A** and **B** satisfy precisely the same identities? (2) Do they satisfy the same quasi-identities? (3) Do **A** and **B** have the same set of term operations?

In addition to the general case in which we allow arbitrary (finite) algebras, we consider each of these problems under the restrictions that all operations are unary and that **A** and **B** have cardinality two. We briefly discuss the relationship of these problems to algebraic specification theory.

Key words. variety, quasi-variety, clone, term-equivalence, computational complexity, logarithmic space, polynomial space, hyperexponential time, nondeterminism

AMS subject classifications. Primary, 68Q25; Secondary, 08B15, 08C15, 08A40, 68Q65

PII. S0097539798345944

1. Introduction.

There are several relationships between mathematical structures that might be considered "fundamental." First and foremost is certainly the isomorphism relation. Questions about isomorphic structures occur throughout mathematics and apply to universal algebras, topological spaces, graphs, partially ordered sets, etc. Many other relationships are more specialized. For example, given two graphs **G** and **H**, one may wish to know whether **H** is a subgraph of **G** or perhaps a minor of **G**.

Properly formulated, questions about these relationships give rise to complexity questions. Generally speaking, we must impose some sort of finiteness assumption on the structures in question so that notions of computational complexity make sense. The complexity of various isomorphism problems has received a great deal of attention. The graph isomorphism problem has been intensively studied, partly because its exact relationship to the classes **P** and **NP** is still unknown, and partly because it provides a paradigm for other problems of unknown complexity status. In this case, both graphs are assumed to have finitely many vertices and finitely many edges. With a similar formulation, the isomorphism problem for algebras has the same complexity as does graph isomorphism. More generally, Kozen [17] showed that the isomorphism problem for finitely presented algebras has this same complexity. See [4, 16, 19] for further discussion and references on the isomorphism problem.

In this paper we consider the complexity of three relationships that arise from considerations in universal algebra. Any algebraic structure satisfies certain identities and fails to satisfy others. Roughly speaking, an identity is an equality between two expressions built from the operations of the algebra. Examples of identities are the associative law (which involves one binary operation) and DeMorgan's law (two binary

*Received by the editors October 19, 1998; accepted for publication (in revised form) October 26, 1999; published electronically June 3, 2000. A preliminary version of this paper appeared in *The* 16th *Symposium on Theoretical Computer Science (STACS* '99*),* Lecture Notes in Comput. Sci. 1563, Springer-Verlag, Berlin, 1999, pp. 163–172.

http://www.siam.org/journals/sicomp/30-2/34594.html

†Department of Mathematics, Iowa State University, 400 Carver Hall, Ames, IA 50011-5454 (cbergman@iastate.edu).

‡Department of Computer Science, Iowa State University, 226 Atanasoff Hall, Ames, IA 50011-5454 (slutzki@cs.iastate.edu).

operations and one unary operation). Identities are one of the primary organizing tools in algebra.

Given two algebras $\mathbf{A}$ and $\mathbf{B}$, we may ask whether they satisfy precisely the same set of identities. Notice that this is a far weaker notion than isomorphism. For example, any algebra satisfies the same identities as each of its direct powers. Nevertheless, if $\mathbf{A}$ and $\mathbf{B}$ satisfy the same identities, then they will be constrained to behave in a similar way. One of our problems, called VAR-EQUIV, is this: Given two finite algebras of the same finite similarity type, determine whether they satisfy the same identities.

This problem has implications for several areas of computer science. *Formal algebraic specifications* are expressions in a language which describe the properties and input-output behavior that a software system must exhibit, without putting any restrictions on the way in which these properties are implemented. This *abstraction* makes formal specifications extremely useful in the process of developing software systems where it serves as a reference point for users, implementers, testers, and writers of instruction manuals. Formal specifications have been applied successfully in deployment of sophisticated software systems; see [33], especially the references there.

Mathematically, formal algebraic specifications are firmly grounded on algebraic concepts, especially ideas, notions, and methods from universal algebra [6]. The relationship between implementation and equational specification corresponds, in algebraic terms, to the relationship between an algebra and a set of identities satisfied by the algebra. Thus, two algebras that satisfy the same identities correspond to a pair of implementations with precisely the same specifications. The computational complexity of these problems, in the universal algebraic framework, is thus quite relevant to the body of research in formal specification theory, and to the construction of supporting tools such as theorem provers and model checkers.

Generalizing the notion of identity, we arrive at a quasi-identity. We shall leave a precise definition for section 2, but crudely speaking, a quasi-identity involves a conjunction of identities and an implication. An example is the left-cancellation law (for, say, a semigroup). In direct analogy with the previous problem we can ask for the complexity of the following. Given two finite algebras of the same finite similarity type, determine whether they satisfy exactly the same quasi-identities. This notion too extends to algebraic specification theory, since "conditional specifications" take the form of quasi-identities.

Our third problem involves the term operations of an algebra. Although an algebra may be endowed with only finitely many basic operations, we can construct many more by composing the basic ones in various combinations. These are called the term operations of the algebra. Two algebras (presumably of different similarity types) are called term-equivalent if they have the same universe and exactly the same set of term operations. In universal algebra, term-equivalent algebras are considered the same "for all practical purposes." The problem we call TERM-EQUIV is that of determining whether two finite algebras are term-equivalent. Returning once again to the realm of specification theory, in this problem we are asking whether a pair of implementations for two entirely different specifications has the property that it exhibits the same input-output behavior. See [25], where this notion is called the "behavioral equivalence of specifications."

Each of these three problems makes sense for arbitrary finite algebras with an arbitrary (but finite) set of basic operations. In addition to this most general formu-

lation, we consider, for each of the three problems, two more restricted settings that, experience tells us, may result in different complexities (see Table 4.1). The first is to require that all basic operations on our algebras be unary. In the second, we only consider algebras of cardinality two.

Unary algebras constitute, from the standpoint of similarity type, the simplest sort of algebraic structure. The set of available term operations is quite small, and the free algebras in the generated variety have a simple structure. Furthermore, (finite) unary algebras capture the algebraic aspects of deterministic (finite-state) automata. (Here, the universe of the algebra corresponds to the set of states, and the similarity type corresponds to the input alphabet of the automata.)

Algebras of cardinality two are, of course, the smallest nontrivial algebras. These are the "Boolean" algebras, and they play an important role in the study of Boolean functions and circuits; see [23] and [32]. For us, the clue that the complexity of our problems will be lower when restricted to two-element algebras comes from the lattice of clones, called Post's lattice [24, 23]. Over a set of cardinality two, the lattice of clones is highly structured and quite manageable. Most of the nice properties of the lattice seem to disappear over larger base sets. One can hope that a good understanding of Post's lattice will lead to the design of more efficient algorithms. Algorithm 7.3 is an example of such an improvement over the "obvious" approach.

The first two sections of this paper are devoted to a development of the necessary background in both universal algebra and complexity theory, for the benefit of those unfamiliar with the basic notions in these fields. In section 3, we formally state the problems we will discuss and outline the major results. Then one section is devoted to each of the three main problems under consideration.

2. Universal algebraic preliminaries. Our primary reference for definitions and basic facts of universal algebra is [20]. Other good references are [5], especially for the material on quasi-varieties, and [10]. Although a bit dated, Taylor's survey in [31] is particularly readable. However, for the benefit of those readers unfamiliar with this material, we give an informal summary of the most important concepts that we will need in this paper.

Let A be a nonempty set and k a nonnegative integer. A k-ary *operation on* A is a function from A^k to A. The integer k is called the *rank* of the operation. Note that if $k = 0$, then $A^k = \{\varnothing\}$, so that a nullary operation is effectively an element of A.

Let F be a set of symbols (called *operation symbols*), and let ρ be a function from F to the nonnegative integers. An *algebra of similarity type* ρ is a structure $\mathbf{A} = \langle A, F^{\mathbf{A}} \rangle$ in which A is a nonempty set and $F^{\mathbf{A}} = \langle f^{\mathbf{A}} : f \in F \rangle$, where each $f^{\mathbf{A}}$ is an operation on A of rank $\rho(f)$. The members of $F^{\mathbf{A}}$ are called the *basic operations* of $\mathbf{A}$, and the set A is called the *universe,* or *underlying set* of $\mathbf{A}$. We will often leave off the superscript $\mathbf{A}$ when no confusion will result. The notation "$\mathbf{A} \sim \mathbf{B}$" will indicate that $\mathbf{A}$ and $\mathbf{B}$ have the same similarity type.

Suppose that $\mathbf{A} = \langle A, F^{\mathbf{A}} \rangle$ and $\mathbf{B} = \langle B, F^{\mathbf{B}} \rangle$ are two algebras of similarity type ρ. A function $\psi\colon B \to A$ is called a *homomorphism* if, for every $f \in F$ and $b_1, \ldots, b_{\rho(f)} \in B$, we have $\psi f^{\mathbf{B}}(b_1, \ldots, b_{\rho(f)}) = f^{\mathbf{A}}(\psi b_1, \ldots, \psi b_{\rho(f)})$. Injective homomorphisms are often called *embeddings.* The algebras $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic,* denoted $\mathbf{A} \cong \mathbf{B}$, if there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ that is a bijection. $\mathbf{B}$ is called a *subalgebra* of $\mathbf{A}$ if $B \subseteq A$ and the inclusion map is a homomorphism. Thus $\mathbf{B}$ is isomorphic to a subalgebra of $\mathbf{A}$ if and only if there is an embedding of $\mathbf{B}$ into $\mathbf{A}$.

Let J be a set, and for each $j \in J$ let A_j be a nonempty set. Then the Cartesian product, $\prod_{j \in J} A_j$, is the set of all sequences $a = \langle\, a_j : j \in J \,\rangle$ such that for every

$j \in J$, $a_j \in A_j$. For each $i \in J$, there is a surjective function $\pi_i \colon \prod_{j\in J} A_j \to A_i$ mapping the sequence a to its ith component, a_i. We shall reserve the symbol "π" for these mappings. Suppose now that $\langle\, \mathbf{A}_j : j \in J \,\rangle$ is a sequence of algebras, all of similarity type ρ. Then $\prod_{j\in J} \mathbf{A}_j$ is the algebra (of type ρ) whose universe is the Cartesian product of the sets A_j, with basic operations that act coordinatewise, using the basic operations of each $\mathbf{A}_j$. In other words, if f is a basic operation symbol of rank n and $x_1, x_2, \dots, x_n \in \prod_{j\in J} A_j$, then

$$\pi_i f^{\prod \mathbf{A}_j}(x_1, \dots, x_n) = f^{\mathbf{A}_i}(\pi_i x_1, \dots, \pi_i x_n) \qquad \forall\ i \in J.$$

In addition to the basic operations of an algebra, one can create new operations by composing the basic ones. Specifically, let A be a set, f an n-ary operation on A, and let $g_1, \dots, g_n$ be k-ary operations on A. Then the *generalized composition of f with $g_1, \dots, g_n$,* denoted $f[g_1, \dots, g_n]$, is the k-ary operation that maps the k-tuple $\mathbf{a} = (a_1, \dots, a_k)$ to $f\big(g_1(\mathbf{a}), \dots, g_n(\mathbf{a})\big)$.

For each positive integer n and $1 \le j \le n$ we define the jth n-ary *projection operation* by $p_j^n(x_1, \dots, x_n) = x_j$. In particular, p_1^1 is the identity operation. A *clone* on a set A is a set of operations on A containing all projections and closed under generalized composition. The set of all clones on A is obviously ordered by set-theoretic inclusion. The smallest clone consists of nothing but the projection operations, while the largest clone contains all operations on A. It is easy to see that the intersection of a family of clones on A is again a clone. Therefore, if E is any set of operations on A, we define the *clone on A generated by E* to be

$$\mathrm{Clo}^A(E) = \bigcap \{\, C : E \subseteq C \text{ and } C \text{ a clone on } A \,\}. \tag{2.1}$$

For an algebra $\mathbf{A} = \langle A, F\rangle$, the clone of *term operations of* $\mathbf{A}$ is simply $\mathrm{Clo}^A(F)$, the clone on A generated by F. This is typically denoted $\mathrm{Clo}(\mathbf{A})$. For any positive integer m, the set of m-ary members of $\mathrm{Clo}(\mathbf{A})$ is denoted $\mathrm{Clo}_m(\mathbf{A})$.

While (2.1) serves as a definition of $\mathrm{Clo}(\mathbf{A})$, it does not provide any information as to the contents of this clone. Intuitively, an operation on A is a member of $\mathrm{Clo}(\mathbf{A})$ if and only if it can be built up by generalized composition, from the basic operations of $\mathbf{A}$ and the projections. This is formalized in the following theorem. The proof is straightforward; see [20, Theorem 4.3].

THEOREM 2.1. *Let F be a set of operations on a set A, and let m be a positive integer. The set* $\mathrm{Clo}_m^A(F)$ *of* m-ary *members of* $\mathrm{Clo}^A(F)$ *is the smallest set X of* m-ary *operations on A such that*

(i) $p_i^m \in X$ *for* $i = 1, 2, \dots, m$;
(ii) *if* $f \in F$ *and* $g_1, g_2, \dots, g_{\rho(f)} \in X$, *then* $f[g_1, \dots, g_{\rho(f)}] \in X$.

Just as the basic operations of an algebra $\mathbf{A}$ are instances of the operation symbols, we would like to have syntactic objects that correspond to the term operations of $\mathbf{A}$. One way to do this is as follows.

DEFINITION 2.2. *Let $X = \{x_1, x_2, \dots\}$ be a countably infinite set of variables and $\rho \colon F \to \{0, 1, \dots\}$ a similarity type, with F disjoint from X. The set of* terms of type ρ *is the smallest set T of strings such that*

1. $X \subseteq T$;
2. $\rho^{-1}(0) \subseteq T$;
3. *if* $f \in F$ *and* $t_1, \dots, t_{\rho(f)} \in T$, *then* $f(t_1, \dots, t_{\rho(f)}) \in T$.

A term is n-ary *if the only variables that appear in the string come from* $\{x_1, \dots, x_n\}$.

If $\mathbf{A}$ is an algebra and t is an n-ary term, then we can assign an n-ary term operation $t^{\mathbf{A}}$ to t as follows. If $t = x_i$, then $t^{\mathbf{A}} = (p_i^n)^{\mathbf{A}}$. If $t = f(t_1, \dots, t_k)$,

then $t^{\mathbf{A}} = f^{\mathbf{A}}[t_1^{\mathbf{A}}, \ldots, t_k^{\mathbf{A}}]$. Comparing the assertions in Theorem 2.1 to the above definitions, we see that $\mathrm{Clo}(\mathbf{A}) = \{\, t^{\mathbf{A}} : t \text{ is a term} \,\}$.

An *identity* of type ρ is simply a pair of terms, although we usually write it in the form $s \approx t$. An algebra $\mathbf{A}$ satisfies the identity $s \approx t$ if $s^{\mathbf{A}} = t^{\mathbf{A}}$. In the usual terminology of first-order logic, this is the same as asserting that the model $\mathbf{A}$ satisfies the sentence

$$(\forall x_1)(\forall x_2)\cdots(\forall x_n)\,\big(s(x_1, \ldots, x_n) \approx t(x_1, \ldots, x_n)\big),$$

where s and t are n-ary terms.

For example, if our similarity type consists of two binary operation symbols, $+$ and $*$, then both the commutative law $x + y \approx y + x$ and the distributive law $x * (y + z) \approx (x * y) + (x * z)$ are identities that may or may not hold in any particular algebra. Notice that in this example we have adopted the usual custom of writing binary operations in "infix" form and using variables x, y, z instead of x_1, x_2, x_3.

A *quasi-identity* is a first-order sentence of the form

$$(\forall x_1)\cdots(\forall x_n)\,\Big(\bigwedge_{j=1}^{m} s_j(x_1, \ldots, x_n) \approx t_j(x_1, \ldots, x_n) \longrightarrow u(x_1, \ldots, x_n) \approx v(x_1, \ldots, x_n)\Big),$$

where each s_j, t_j, u, and v is an n-ary term, and "$\bigwedge$" denotes conjunction. Every identity is a quasi-identity, by taking $m = 0$. In the context of algebraic structures, quasi-identities are precisely the universal Horn sentences. An example of a quasi-identity that is not an identity is the left-cancellation law $x * y \approx x * z \to y \approx z$, which holds, for example, in the positive integers under multiplication but not for all integers under multiplication. By contrast, the formula

$$(x \neq 0 \ \wedge \ x * y \approx x * z) \to y \approx z$$

is not a quasi-identity, since we do not permit negation symbols.

Now, fix a similarity type ρ, and let $\mathcal{K}$ be a class of algebras, all of type ρ. $\mathcal{K}$ is called a *variety,* or *equational class,* if there is a set Σ of identities (not necessarily finite) such that $\mathcal{K}$ is exactly the class of all algebras satisfying every identity in Σ. Notice that many familiar classes of algebras are varieties. For example, the class of all groups is a variety if we take our similarity type to consist of one binary, one unary, and one nullary operation, and Σ to consist of the five identities

$$x \cdot (y \cdot z) \approx (x \cdot y) \cdot z, \quad x \cdot e \approx e \cdot x \approx x, \quad x \cdot x^{-1} \approx x^{-1} \cdot x \approx e.$$

A classical theorem due to Birkhoff [3] asserts that a class $\mathcal{K}$ is a variety if and only if $\mathcal{K}$ is closed under the formation of subalgebras, arbitrary products, and homomorphic images. This remarkable fact connects the purely syntactic idea of an equation to the familiar algebraic constructions we discussed earlier.

It is convenient to introduce the following notation. Let $\mathcal{K}$ be a class of algebras of the same similarity type. Then $\mathbf{H}(\mathcal{K}), \mathbf{S}(\mathcal{K})$, and $\mathbf{P}(\mathcal{K})$ denote the class of all

algebras isomorphic to a homomorphic image, subalgebra, and product of members of $\mathcal{K}$, respectively. Thus $\mathcal{K}$ is a variety if and only if $\mathcal{K} = \mathbf{H}(\mathcal{K}) = \mathbf{S}(\mathcal{K}) = \mathbf{P}(\mathcal{K})$.

It is easy to see that the intersection of a family of varieties (all of similarity type ρ) is again a variety. In fact, a defining set of identities will be the union of defining sets for each of the component varieties. Thus we define, for any class $\mathcal{K}$, the *variety generated by* $\mathcal{K}$ to be

$$\mathbf{V}(\mathcal{K}) = \bigcap \{ \mathcal{V} : \mathcal{V} \text{ is a variety and } \mathcal{K} \subseteq \mathcal{V} \}.$$

It is not hard to show that for any class $\mathcal{K}$,

$$\mathbf{V}(\mathcal{K}) = \mathbf{HSP}(\mathcal{K}) = \mathrm{Mod}(\mathrm{Id}(\mathcal{K})). \tag{2.2}$$

In this equation, $\mathrm{Id}(\mathcal{K})$ denotes the set of all identities true in every member of $\mathcal{K}$, and $\mathrm{Mod}(\Sigma)$ denotes the class of all algebras in which every identity in Σ holds. For a single algebra $\mathbf{A}$, it is customary to write $\mathbf{V}(\mathbf{A})$ instead of $\mathbf{V}(\{\mathbf{A}\})$.

Just as a variety is defined by identities, a *quasi-variety* is defined by quasi-identities. Most of the assertions we have made about varieties have analogous formulations for quasi-varieties. For example, there is a Birkhoff-type theorem that states that a class $\mathcal{K}$ is a quasi-variety if and only if it is closed under the formation of subalgebras, products, and ultraproducts. (We will not need the notion of an ultraproduct here. See [5, p. 210].) The *quasi-variety generated by* $\mathcal{K}$ is

$$\mathbf{Q}(\mathcal{K}) = \bigcap \{ \mathcal{Q} : \mathcal{Q} \text{ is a quasi-variety and } \mathcal{K} \subseteq \mathcal{Q} \}.$$

Since every variety is also a quasi-variety, we always have $\mathcal{K} \subseteq \mathbf{Q}(\mathcal{K}) \subseteq \mathbf{V}(\mathcal{K})$, but the reverse inclusions are usually false. For example, let $\mathbb{Z}$ denote the group of integers, and $\mathcal{K} = \{\mathbb{Z}\}$. Then $\mathbf{Q}(\mathcal{K})$ is the class of torsion-free Abelian groups, while $\mathbf{V}(\mathcal{K})$ consists of all Abelian groups. (A group is *torsion-free* if no element except the identity has finite order.)

There is one feature of the quasi-variety notion that deserves special mention. Since we will need it later, we state it formally.

THEOREM 2.3. *Let* $\mathbf{A}$ *be a finite algebra. Then* $\mathbf{Q}(\mathbf{A}) = \mathbf{SP}(\mathbf{A})$.

Proof. Since $\mathbf{A} \in \mathbf{SP}(\mathbf{A}) \subseteq \mathbf{Q}(\mathbf{A})$, it suffices to show that $\mathbf{SP}(\mathbf{A})$ is a quasi-variety. Closure under $\mathbf{S}$ and $\mathbf{P}$ is easy. Closure under ultraproducts boils down to the fact that, since A is finite, an ultraproduct of copies of $\mathbf{A}$ is simply isomorphic to $\mathbf{A}$ again. See [5, Lemma 6.5 and Theorem 2.25]. □

An algebra $\mathbf{B}$ is called *simple* if it has more than one element and every homomorphism with domain B is either injective or trivial (i.e., has a one-element image). For example, a group is simple in this sense if and only if it has exactly two normal subgroups. We will need the following easy result.

PROPOSITION 2.4. *Let* $\mathbf{B}$ *be a simple algebra and* $\mathbf{A}$ *be any algebra of the same similarity type. If* $\mathbf{B} \in \mathbf{SP}(\mathbf{A})$, *then* $\mathbf{B} \in \mathbf{S}(\mathbf{A})$.

Proof. By assumption there is a set I and an embedding $\psi\colon \mathbf{B} \to \mathbf{A}^I$. Pick distinct elements a, b from B. Then $\psi(a) \neq \psi(b)$, so for some $i \in I$, $\psi(a)$ and $\psi(b)$ differ in the ith component. Let π_i be the mapping from $\mathbf{A}^I$ to $\mathbf{A}$ that assigns to each I-tuple its ith coordinate. Then $\pi_i \circ \psi$ is a homomorphism from $\mathbf{B}$ to $\mathbf{A}$ which is nontrivial, since $\pi_i\psi(a) \neq \pi_i\psi(b)$. From the simplicity of $\mathbf{B}$, it follows that $\pi_i \circ \psi$ is an embedding. Hence $\mathbf{B} \in \mathbf{S}(\mathbf{A})$. □

TABLE 3.1
Some complexity classes.

$\mathbf{L} = \mathbf{DSPACE}(\log n)$,	logarithmic space;
$\mathbf{NL} = \mathbf{NSPACE}(\log n)$,	nondeterministic logarithmic space;
$\mathbf{P} = \bigcup_{k\geq 1} \mathbf{DTIME}(n^k)$,	polynomial time;
$\mathbf{NP} = \bigcup_{k\geq 1} \mathbf{NTIME}(n^k)$,	nondeterministic polynomial time;
$\mathbf{PSPACE} = \bigcup_{k\geq 1} \mathbf{DSPACE}(n^k)$,	polynomial space;
$\mathbf{EXPTIME} = \bigcup_{k\geq 1} \mathbf{DTIME}(2^{n^k})$,	exponential time;
$\mathbf{2\text{-}EXPTIME} = \bigcup_{k\geq 1} \mathbf{DTIME}(2^{2^{n^k}})$,	hyperexponential time.

3. Complexity preliminaries. Since this paper deals with the computational complexity of problems in universal algebra, we will include a brief review of the complexity classes used in this paper, mainly for the benefit of those readers unfamiliar with the common notation and terminology used. Consult any of [27, 22, 13, 9] for an in-depth treatment of computational complexity.

Languages (i.e., sets of finite strings over some fixed alphabet) are viewed as encodings of problems. Given a function $f\colon \mathbb{N} \to \mathbb{N}$, we denote by $\mathbf{DTIME}(f(n))$ (respectively, $\mathbf{DSPACE}(f(n))$) the set of all languages decidable by a deterministic Turing machine in time (respectively, space) $O(f(n))$. In an analogous way, the nondeterministic classes $\mathbf{NTIME}(f(n))$ and $\mathbf{NSPACE}(f(n))$ are defined in terms of nondeterministic Turing machines. The complexity classes referred to in this paper are defined in Table 3.1 (see [13]). The class **P** consists of those problems for which it is considered to be feasible to use a computer to find a solution. By contrast, a problem lies in **NP** if a proposed solution can be *verified* in polynomial time.

These classes form a chain of inclusions:

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{2\text{-}EXPTIME}. \tag{3.1}$$

Of these the following are known to be proper:

$$\mathbf{NL} \subsetneq \mathbf{PSPACE}, \qquad \mathbf{P} \subsetneq \mathbf{EXPTIME} \subsetneq \mathbf{2\text{-}EXPTIME}. \tag{3.2}$$

All of the other inclusions (except for the obvious ones that follow from (3.2)) represent deep open problems in theoretical computer science, the most famous of which is the seemingly unapproachable $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem.

In addition to the classes listed above, several others are worth noting briefly. First, both **EXPTIME** and **2-EXPTIME** have nondeterministic analogues which could be added to our list. Surprisingly, the same is not true for **PSPACE**. In 1970, Savitch [26] proved that for any function f with $f(n) \geq \log n$, $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{DSPACE}(f(n)^2)$. In particular, $\mathbf{PSPACE} = \mathbf{NPSPACE}$.

Second, for any complexity class **C**, one can define the dual class **co-C**, consisting of those languages (problems) whose *complements* lie in **C**. For example, since the graph-isomorphism problem lies in **NP**, the graph-nonisomorphism problem lies in

co-NP. It is not hard to see that every deterministic class **C** is closed under complements, i.e., **C** = **co-C**. Furthermore, it was shown independently by Immerman and Szelepcsényi [12, 29], that if $f(n) \geq \log n$, then **NSPACE**$(f(n))$ is closed under complements. From this we obtain **NL** = **co-NL**.

Among the most useful tools in complexity theory are the concepts of resource-bounded reducibility and completeness, both borrowed from recursive function theory. Given two languages A and B, we say that A is *polynomial-time, many-one reducible to B,* and we write $A \leq_m^p B$, if there is a polynomial-time computable function f such that

$$x \in A \iff f(x) \in B. \tag{3.3}$$

It is not hard to verify that "$\leq_m^p$" is both reflexive and transitive. A language (i.e., problem) A is $\leq_m^p$*-hard* (or just *hard*) for a class **C** of problems, if for every $C \in \mathbf{C}$, $C \leq_m^p A$. The language A is $\leq_m^p$*-complete* (or just complete) for **C** if A is hard for **C** and $A \in \mathbf{C}$.

Unfortunately, polynomial-time reductions are not useful within the class **P**, since for any nontrivial $A, B \in \mathbf{P}$ we have $A \leq_m^p B \leq_m^p A$. Instead, we must resort to a weaker notion called *log-space reduction.* This simply means that in (3.3), the function f must be computable in logarithmic space.

Since it is generally believed that all of the inclusions in (3.1) are in fact proper, a proof of completeness of a problem A in any of those classes is viewed as providing overwhelming evidence that A does not belong to the immediately preceding class in (3.1). It follows from (3.2) that any problem which is complete for **EXPTIME** (such as our Term-Equiv) fails to lie in **P**. Such a problem is *provably* intractable.

In the problems we will be investigating, the input consists of pairs of finite algebras of *finite similarity type* (i.e., algebras having only finitely many basic operation symbols). Let us be more specific as to the form we assume the input will take. The underlying set of an algebra can be assumed to be $\{0, 1, \ldots, n-1\}$ for some positive integer n. In fact, this set can be represented in the input by its cardinality, which requires $\log n$ bits of storage. (All logarithms will be to the base 2.) A k-ary operation on this set is represented as a table of values or, in other words, a k-dimensional array with both the indices and entries coming from $\{0, 1, \ldots, n-1\}$. Notice that this can be represented in the input stream using $n^k \cdot \log n$ bits.

Let us define the *rank* of an algebra to be the maximum rank of any of its basic operations. Thus an algebra of cardinality n and rank k will require at least $n^k \cdot \log n$ bits to specify.

There are certainly other ways of specifying operations, such as with circuits or Turing machines, but we shall not pursue this idea here. Also, from now on we shall assume that all algebras are finite and of finite similarity type.

4. Discussion of the problems. In this paper we shall consider three equivalence relations on algebraic structures. First, given two algebras **A** and **B** of the same similarity type, is $\mathbf{V}(\mathbf{A}) = \mathbf{V}(\mathbf{B})$? In light of (2.2), this is equivalent to asking whether **A** and **B** satisfy exactly the same identities. Note that this only makes sense if the two algebras have the same similarity type. It was shown in [14] that this problem is decidable. We shall denote this problem Var-Equiv. Thus

$$\text{Var-Equiv} = \{\, (\mathbf{A}, \mathbf{B}) : \mathbf{A} \sim \mathbf{B} \;\&\; \mathbf{V}(\mathbf{A}) = \mathbf{V}(\mathbf{B}) \,\}.$$

Recently, Z. Székely proved that Var-Equiv is **NP**-hard; see [28].

We have an analogous problem for quasi-varieties:

$$\text{QVAR-EQUIV} = \{\, (\mathbf{A}, \mathbf{B}) : \mathbf{A} \sim \mathbf{B} \;\&\; \mathbf{Q}(\mathbf{A}) = \mathbf{Q}(\mathbf{B}) \,\}.$$

The assertion $(\mathbf{A}, \mathbf{B}) \in$ QVAR-EQUIV is equivalent to $\mathbf{A}$ and $\mathbf{B}$ satisfying exactly the same quasi-identities. Surprisingly, even though the logical form of a quasi-identity is much more complicated than that of an identity, QVAR-EQUIV has a relatively low computational complexity compared to VAR-EQUIV. Note that QVAR-EQUIV $\subseteq$ VAR-EQUIV as sets.

The third problem we shall consider is term-equivalence. Two algebras $\mathbf{A}$ and $\mathbf{B}$ are *term-equivalent* if and only if they have the same underlying set and $\mathrm{Clo}(\mathbf{A}) = \mathrm{Clo}(\mathbf{B})$. For this problem, we do not require that $\mathbf{A}$ and $\mathbf{B}$ have the same similarity type, but we do require that they have the same universe:

$$\text{TERM-EQUIV} = \{\, (\mathbf{A}, \mathbf{B}) : A = B \;\&\; \mathrm{Clo}(\mathbf{A}) = \mathrm{Clo}(\mathbf{B}) \,\}.$$

It was shown in [1] that TERM-EQUIV is complete for **EXPTIME**.

There are several restrictions of these problems which are of interest and which turn out to have a lower complexity. In particular, we can bound either the cardinality of the underlying sets or the ranks of the algebras. For example, it was shown in [18] that TERM-EQUIV is complete for **PSPACE** when restricted to *unary algebras,* that is, algebras in which every operation has rank 1. For each of our three problems, we shall consider, in addition to the general case, the subcases obtained by considering only unary algebras and only two-element algebras. We shall denote the subcase by appending a superscript "1" or subscript "2" to the problem. To be precise, let us define

$$U = \{\, \mathbf{A} : \mathbf{A} \text{ is a unary algebra} \,\},$$
$$T = \{\, \mathbf{A} : A = \{0, 1\} \,\}.$$

Then $\mathrm{X}^1 = \mathrm{X} \cap (U \times U)$ and $\mathrm{X}_2 = \mathrm{X} \cap (T \times T)$ for X any one of TERM-EQUIV, VAR-EQUIV, or QVAR-EQUIV.

Our results for each of these nine problems are summarized in Table 4.1. In this

TABLE 4.1
Summary of results.

	QVAR-EQUIV	TERM-EQUIV	VAR-EQUIV
card2	**L**	**NL**	**L**
unary	**NP**	**PSPACE***	**PSPACE**
general	**NP***	**EXPTIME***	**2-EXPTIME**

table, the first row concerns the subcase consisting of two-element algebras, the second concerns the subcase of unary algebras, and the third concerns the general case. Each of the nine entries gives the smallest complexity class known to contain the problem, and a superscript "*" indicates that the result is sharp, i.e., the problem is complete for the given complexity class.

5. The quasi-variety problems. We begin with the problems that ask whether two algebras generate the same quasi-variety. It is sometimes convenient to work with an asymmetric variant of this problem:

$$\text{QVAR-MEM} = \{\, (\mathbf{A}, \mathbf{B}) : \mathbf{A} \sim \mathbf{B} \;\&\; \mathbf{B} \in \mathbf{Q}(\mathbf{A}) \,\}.$$

Since the "**Q**" operator has the usual properties of closure, we obviously have

$$(\mathbf{A},\mathbf{B}) \in \text{QVAR-EQUIV} \iff (\mathbf{A},\mathbf{B}),\ (\mathbf{B},\mathbf{A}) \in \text{QVAR-MEM}. \tag{5.1}$$

It follows that for an instance of size s, membership in QVAR-EQUIV can be tested with two calls to an algorithm for QVAR-MEM, both using inputs of size s. In a natural way, we also have the restricted problems QVAR-MEM^1 and QVAR-MEM$_2$ consisting of pairs of unary and two-element algebras, respectively.

THEOREM 5.1. QVAR-MEM $\in$ **NP**.

Proof. Let $\mathbf{A}$ and $\mathbf{B}$ be a pair of similar, finite algebras. We wish to determine whether $\mathbf{B} \in \mathbf{Q}(\mathbf{A})$. Here is a nondeterministic algorithm. For each unordered pair $\{a,b\}$ of distinct elements of B, guess a function $\psi_{\{a,b\}}\colon B \to A$ such that $\psi_{\{a,b\}}(a) \neq \psi_{\{a,b\}}(b)$. Test whether $\psi_{\{a,b\}}$ is a homomorphism. If it is not, then reject. But if every $\psi_{\{a,b\}}$ passes the homomorphism test, then accept.

To see that our algorithm is correct, suppose that we accept the pair $(\mathbf{A},\mathbf{B})$. Enumerate the doubletons $\{a_i,b_i\}$, $i = 1,2,\dots,n$, of elements of B. Define a function $\psi\colon B \to A^n$ such that the ith coordinate of $\psi(x)$ is $\psi_{\{a_i,b_i\}}(x)$. The fact that every $\psi_{\{a_i,b_i\}}$ is a homomorphism ensures that ψ is a homomorphism. And ψ will be injective since, if $a \neq b$, then $\psi(a)$ and $\psi(b)$ differ in the ith coordinate, where $\{a,b\}$ is the ith pair in our enumeration. It follows that $\mathbf{B}$ is isomorphic to a subalgebra of a direct power of $\mathbf{A}$, and consequently $\mathbf{B}$ lies in the quasi-variety generated by $\mathbf{A}$.

Conversely, suppose that $\mathbf{B} \in \mathbf{Q}(\mathbf{A})$. By Theorem 2.3 there is a set J and an embedding $\psi\colon \mathbf{B} \to \mathbf{A}^J$. Using the notation of the previous paragraph, for each $i = 1,\dots,n$, we have $\psi(a_i) \neq \psi(b_i)$, and hence there is some $j_i \in J$ such that $\psi(a_i)$ and $\psi(b_i)$ differ in their j_ith coordinate. Let $\pi_{j_i}\colon \mathbf{A}^J \to \mathbf{A}$ denote the coordinate projection homomorphism. Then $\psi_{\{a_i,b_i\}} = \pi_{j_i} \circ \psi$ constitutes an appropriate guess. Thus our algorithm will accept the pair $(\mathbf{A},\mathbf{B})$.

Finally, we need to estimate the (nondeterministic) running time of the algorithm. Let s denote the size of the input. A function ψ from B to A can be guessed in time on the order of $|B|\cdot|A|$, which is at most s^2. The verification that ψ is a homomorphism also takes time in $O(s^2)$. The total number of functions we need to construct is $\binom{|B|}{2} \leq s^2$. Thus the total running time lies in $O(s^4)$. □

COROLLARY 5.2. *All of the following lie in* **NP***:* QVAR-EQUIV*,* QVAR-MEM^1*,* QVAR-MEM$_2$*,* QVAR-EQUIV^1*, and* QVAR-EQUIV$_2$*.*

Proof. That QVAR-EQUIV $\in$ **NP** follows from Theorem 5.1 and (5.1). The class **NP** is closed under polynomial-time, many-one reductions. Since QVAR-MEM^1 and QVAR-EQUIV^1 are reducible to (indeed special cases of) QVAR-MEM and QVAR-EQUIV, respectively, they too lie in **NP**. The same argument applies to the two-element versions of the problems. □

We shall improve the bounds on QVAR-MEM$_2$ and QVAR-EQUIV$_2$ in Theorem 5.7 below. But first we consider the **NP**-completeness of the other problems discussed in Corollary 5.2. To do this, we use a transformation from (directed) graphs to unary algebras described in [11]. By a *digraph* we shall mean a structure $\mathbf{G} = \langle G,\theta\rangle$ in which G is a nonempty set and $\theta \subseteq G \times G$. $\mathbf{G}$ is *loopless* if for no x in G do we have $(x,x) \in \theta$.

Let $\mathbf{G} = \langle G,\theta\rangle$ and $\mathbf{H} = \langle H,\tau\rangle$ be digraphs. A morphism from $\mathbf{H}$ to $\mathbf{G}$ is a function $\psi\colon H \to G$ such that $(x,y) \in \tau \implies \big(\psi(x),\psi(y)\big) \in \theta$. $\mathbf{H}$ is a subgraph of $\mathbf{G}$ if $H \subseteq G$ and the inclusion map is a morphism. If $\mathbf{G}_i = \langle G_i,\theta_i\rangle$, $i \in I$ is a family of digraphs, then the product graph is $\langle G,\theta\rangle$, where $G = \prod_{i\in I} G_i$ and $\theta = \{\,(x,y) \in G \times G : (\forall i \in I)\ (x_i,y_i) \in \theta_i\,\}$. Just as for algebras, we use the

operators **S** and **P** to denote closure under the formation of subgraph and product graph, respectively.

Given a digraph $\mathbf{G} = \langle G, \theta \rangle$, we now define an algebra $\mathbf{G}^*$ as follows. The universe of $\mathbf{G}^*$ is the set $G \cup \theta \cup \{u, v\}$ where u and v are points not appearing in either G or θ. $\mathbf{G}^* = \langle\, G^*, f_0, f_1 \,\rangle$, where f_0 and f_1 are unary operations defined by

$$\begin{array}{lll} \forall x \in G & f_0(x) = u, & f_1(x) = v; \\ \forall (x,y) \in \theta & f_0((x,y)) = x, & f_1((x,y)) = y; \\ & f_0(u) = v, & f_1(u) = u, \\ & f_0(v) = v, & f_1(v) = u. \end{array}$$

Furthermore, let $\psi\colon \mathbf{H} \to \mathbf{G}$ be a digraph morphism. We define a function $\psi^*\colon H^* \to G^*$ given by

$$\begin{array}{ll} \forall x \in H & \psi^*(x) = \psi(x); \\ \forall (x,y) \in \tau & \psi^*((x,y)) = (\psi(x),\, \psi(y)); \\ \psi^*(u^H) = u^G, & \psi^*(v^H) = v^G. \end{array}$$

LEMMA 5.3 (see Hedrlín and Pultr [11]). *The assignments $\mathbf{G} \mapsto \mathbf{G}^*$ and $\psi \mapsto \psi^*$ constitute a full and faithful functor from the category of digraphs to that of algebras with two unary operations. In other words, for each pair $\mathbf{H}$, $\mathbf{G}$ of digraphs, and each digraph morphism ψ, the function $\psi^*\colon \mathbf{H}^* \to \mathbf{G}^*$ is a homomorphism, and furthermore, the mapping $\psi \mapsto \psi^*$ is a bijection between the morphisms from $\mathbf{H}$ to $\mathbf{G}$ and the homomorphisms between $\mathbf{H}^*$ and $\mathbf{G}^*$. Also, ψ is an injective map if and only if ψ^* is injective.*

We will show **NP**-completeness of QVAR-MEM[1] by exhibiting a reduction from the problem CLIQUE, which is well known to be complete for **NP**; see [9, p. 194]. For a positive integer n, let $\mathbf{K}_n$ denote the digraph with vertex set $\{1, 2, \ldots, n\}$ and edges $\{\, (x,y) : x \neq y \,\}$. We define

$$\text{CLIQUE} = \{\, (\mathbf{G}, n) : \mathbf{G} \text{ a digraph, } n \geq 1 \text{ and } \mathbf{K}_n \in \mathbf{S}(\mathbf{G}) \,\}.$$

A subset of G isomorphic to some $\mathbf{K}_n$ is called a *clique*.

PROPOSITION 5.4. *Let $\mathbf{G}$ be a loopless digraph and n a positive integer. The following are equivalent.*

(i) $(\mathbf{G}, n) \in$ CLIQUE.
(ii) $\mathbf{K}_n \in \mathbf{SP}(\mathbf{G})$.
(iii) $\mathbf{K}_n^* \in \mathbf{SP}(\mathbf{G}^*)$.

Proof. That (i) implies (ii) is trivial. Now assume (ii), i.e., suppose that ψ is an embedding of $\mathbf{K}_n$ into a power $\mathbf{G}^I$. By Lemma 5.3, we have an injective homomorphism ψ^* from $\mathbf{K}_n^*$ to $(\mathbf{G}^I)^*$. Moreover, it is easy to see that $(\mathbf{G}^I)^*$ is isomorphic to a subalgebra of $(\mathbf{G}^*)^I$. Thus (iii) holds.

Finally, suppose that $\phi\colon \mathbf{K}_n^* \to (\mathbf{G}^*)^I$ is an embedding for some set I. Choose any $i \in I$. Then $\pi_i \circ \phi$ is a homomorphism from $\mathbf{K}_n^*$ to $\mathbf{G}^*$, which, by Lemma 5.3, must be of the form ψ^* for some digraph morphism $\psi\colon \mathbf{K}_n \to \mathbf{G}$. Suppose that x and y are distinct elements of $\mathbf{K}_n$. By definition, there is an edge from x to y; consequently, there must be an edge in $\mathbf{G}$ from $\psi(x)$ to $\psi(y)$. Since $\mathbf{G}$ is assumed to be loopless, it must be the case that $\psi(x) \neq \psi(y)$. Therefore, ψ is injective, so (iii) implies (i). □

It is convenient to introduce one more problem involving the relationship between two algebras. Let

$$\text{SUBALG} = \{\, (\mathbf{A}, \mathbf{B}) : \mathbf{A} \sim \mathbf{B} \;\&\; \mathbf{B} \in \mathbf{S}(\mathbf{A}) \,\}.$$

As with our other problems, SUBALG^1 will refer to the restriction of this problem to the case that $\mathbf{A}$ and $\mathbf{B}$ are unary algebras. It is easy to see that SUBALG and SUBALG^1 lie in **NP** (just guess an injective function and check to see if it is a homomorphism).

THEOREM 5.5. *The problems* QVAR-MEM^1, QVAR-MEM, SUBALG^1, SUBALG, *and* QVAR-EQUIV *are all complete for* **NP**.

Proof. We showed in Theorem 5.1, Corollary 5.2, and the comments just above that all of these problems lie in **NP**. The problem CLIQUE is **NP**-complete, and the transformation $(\mathbf{G}, n) \mapsto (\mathbf{G}^*, \mathbf{K}_n^*)$ can be done in polynomial time. Therefore, we deduce from Proposition 5.4 and Theorem 2.3 that QVAR-MEM^1 is also **NP**-complete. Clearly, QVAR-MEM$^1 \leq_m^p$ QVAR-MEM, so QVAR-MEM is **NP**-complete.

By Lemma 5.3, $(\mathbf{G}, n) \in$ CLIQUE if and only if $(\mathbf{G}^*, \mathbf{K}_n^*) \in$ SUBALG^1. Thus SUBALG^1 and also SUBALG are **NP**-complete. This fact is not new; it was first noted in [21].

To complete the proof, we will reduce SUBALG^1 to QVAR-EQUIV. Given a finite unary algebra $\mathbf{B} = \langle B, F\rangle$, let $\mathbf{B}^+ = \big\langle\, B \cup \{e\},\ F \cup \{p, d\}\,\big\rangle$, where $e \notin B$ and $p, d \notin F$. For each $f \in F$, $f(e) = e$. For all $x, y, z \in B \cup \{e\}$ we define

$$p(x) = e;$$

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

The ternary operation d is called the *discriminator operation*. It is an easy exercise to check that any algebra with a discriminator among its term operations is simple.

Now, given two finite unary algebras $\mathbf{A}$ and $\mathbf{B}$, we have the following equivalences.

$$\mathbf{B} \in \mathbf{S}(\mathbf{A}) \iff \mathbf{B}^+ \in \mathbf{S}(\mathbf{A}^+) \iff \mathbf{Q}(\mathbf{A}^+) = \mathbf{Q}(\mathbf{A}^+ \times \mathbf{B}^+). \tag{5.2}$$

Clearly, the equivalences in (5.2) imply that SUBALG$^1 \leq_m^p$ QVAR-EQUIV. The first equivalence is an easy verification. We check the second. First suppose that $\mathbf{B}^+ \in \mathbf{S}(\mathbf{A}^+)$. Then both $\mathbf{A}^+$ and $\mathbf{B}^+$ are members of $\mathbf{Q}(\mathbf{A}^+)$ which is closed under products. Therefore, $\mathbf{A}^+ \times \mathbf{B}^+ \in \mathbf{Q}(\mathbf{A}^+)$, so $\mathbf{Q}(\mathbf{A}^+ \times \mathbf{B}^+) \subseteq \mathbf{Q}(\mathbf{A}^+)$. On the other hand, there is an embedding of $\mathbf{A}^+$ into $\mathbf{A}^+ \times \mathbf{B}^+$ given by $x \mapsto (x, e)$. Hence $\mathbf{Q}(\mathbf{A}^+) \subseteq \mathbf{Q}(\mathbf{A}^+ \times \mathbf{B}^+)$.

Conversely, suppose that $\mathbf{Q}(\mathbf{A}^+) = \mathbf{Q}(\mathbf{A}^+ \times \mathbf{B}^+)$. There is an embedding of $\mathbf{B}^+$ into $\mathbf{A}^+ \times \mathbf{B}^+$ such that $x \mapsto (e, x)$. Thus $\mathbf{B}^+ \in \mathbf{Q}(\mathbf{A}^+) = \mathbf{SP}(\mathbf{A}^+)$. However, $\mathbf{B}^+$ is a simple algebra, so by Proposition 2.4, $\mathbf{B}^+ \in \mathbf{S}(\mathbf{A}^+)$. □

Remark. Referring back to the proof of Theorem 5.5, it is tempting to try to prove the **NP**-hardness of QVAR-EQUIV by using (5.1) together with the **NP**-completeness of QVAR-MEM and Corollary 5.2. However, there are difficulties with this line of argument. To take an analogous situation, the subgraph isomorphism problem is **NP**-complete, but the graph isomorphism problem is in **NP** but most probably is not **NP**-complete.

Unfortunately, the method used to reduce SUBALG^1 to QVAR-EQUIV does not produce a unary algebra, so we are not able to show that QVAR-EQUIV^1 is complete for **NP**. We leave it as an open problem.

PROBLEM 5.6. *Is* QVAR-EQUIV^1 *complete for* **NP**? *Is* QVAR-EQUIV$^1 \in$ **P**?

Now we turn to the problem QVAR-EQUIV$_2$. Suppose that $\mathbf{A}$ and $\mathbf{B}$ are two-element algebras. We claim that $\mathbf{B} \in \mathbf{Q}(\mathbf{A})$ if and only if $\mathbf{B} \cong \mathbf{A}$. To see this, note that every two-element algebra is simple. Therefore, by Proposition 2.4, Theorem 2.3, and the fact that $|B| = |A|$, we obtain

$$\mathbf{B} \in \mathbf{Q}(\mathbf{A}) = \mathbf{SP}(\mathbf{A}) \implies \mathbf{B} \in \mathbf{S}(\mathbf{A}) \implies \mathbf{B} \cong \mathbf{A}.$$

The converse, that $\mathbf{B} \cong \mathbf{A} \implies \mathbf{B} \in \mathbf{Q}(\mathbf{A})$, is trivial.

THEOREM 5.7. QVAR-EQUIV$_2$, QVAR-MEM$_2 \in \mathbf{L}$.

Proof. As we argued in the previous paragraph, $(\mathbf{B}, \mathbf{A}) \in$ QVAR-MEM$_2$ if and only if $\mathbf{B} \cong \mathbf{A}$. There are only two bijections from B to A, and each of these can be tested to see if it is a homomorphism. The testing requires just a couple of counters, each of which has a space bound that is logarithmic in the size of the input. Thus QVAR-MEM$_2 \in \mathbf{L}$. Now apply assertion (5.1) to deduce that QVAR-EQUIV$_2$ $\in \mathbf{L}$. □

6. The variety problems. The problem VAR-EQUIV asks: if $\mathbf{A}$ and $\mathbf{B}$ are two algebras of the same similarity type, is $\mathbf{V}(\mathbf{A}) = \mathbf{V}(\mathbf{B})$? As with quasi-varieties, it is convenient to introduce an auxiliary problem, VAR-MEM:

$$\text{VAR-MEM} = \{\, (\mathbf{A}, \mathbf{B}) : \mathbf{A} \sim \mathbf{B} \;\&\; \mathbf{B} \in \mathbf{V}(\mathbf{A}) \,\}.$$

Unlike the situation for quasi-varieties, the problems VAR-EQUIV and VAR-MEM are interchangeable from the perspective of complexity. We have

$$\begin{aligned} (\mathbf{A}, \mathbf{B}) \in \text{VAR-EQUIV} &\iff (\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{A}) \in \text{VAR-MEM}, \\ (\mathbf{A}, \mathbf{B}) \in \text{VAR-MEM} &\iff (\mathbf{A}, \mathbf{A} \times \mathbf{B}) \in \text{VAR-EQUIV}. \end{aligned} \tag{6.1}$$

The second equivalence follows from the fact that both $\mathbf{A}$ and $\mathbf{B}$ are homomorphic images of $\mathbf{A} \times \mathbf{B}$.

We begin with the two-element problem. The crucial point is the following theorem.

THEOREM 6.1. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two-element algebras of the same similarity type. Then* $\mathbf{V}(\mathbf{A}) = \mathbf{V}(\mathbf{B})$ *if and only if* $\mathbf{A} \cong \mathbf{B}$.

The proof of Theorem 6.1 requires considerably more universal algebra than does the remainder of this paper. For this reason, we have relegated the proof to the appendix.

THEOREM 6.2. VAR-EQUIV$_2 \in \mathbf{L}$.

Proof. From Theorem 6.1, testing whether $(\mathbf{A}, \mathbf{B}) \in$ VAR-EQUIV$_2$ is equivalent to testing $\mathbf{A} \cong \mathbf{B}$. Arguing as we did at the end of section 5, there are only two possible isomorphisms to test. This can be done deterministically in logarithmic space. □

In order to proceed to the remaining two problems, we need some more detailed information on the relationship between clones, terms, and varieties. Let $\mathbf{A} = \langle A, F \rangle$ be an algebra of cardinality n, and let m be a positive integer. An m-ary operation on A, being a function from A^m to A, can also be thought of as an element of the direct power $A^{(A^m)}$. Visualized this way, it is not hard to see that $\mathrm{Clo}_m(\mathbf{A})$ forms a subalgebra of $\mathbf{A}^{(A^m)}$. In fact, Theorem 2.1 can be viewed as asserting that $\mathbf{Clo}_m(\mathbf{A})$ is the subalgebra of $\mathbf{A}^{(A^m)}$ generated by the set $\{p_1^m, \ldots, p_m^m\}$. Notice that we follow our usual typographic convention and print "**Clo**" in boldface when it is to be used as an algebra. Since varieties are closed under the formation of both powers and subalgebras, it follows that both $\mathbf{A}^{(A^m)}$ and $\mathbf{Clo}_m(\mathbf{A})$ lie in $\mathbf{V}(\mathbf{A})$.

Let us be more precise about how this subalgebra could be constructed. For each natural number j, define a set X_j of m-ary operations on A recursively by

$$\begin{aligned} X_0 &= \{p_1^m, p_2^m, \ldots, p_m^m\}, \\ X_{j+1} &= X_j \cup \left\{\, f[g_1, \ldots, g_{\rho(f)}] : f \in F,\ g_1, \ldots, g_{\rho(f)} \in X_j \,\right\}. \end{aligned} \tag{6.2}$$

It is easy to see that $X_0 \subseteq X_1 \subseteq \cdots \subseteq \mathrm{Clo}_m(\mathbf{A})$. Since the total number of m-ary operations on A is $n^{(n^m)}$, there is some index $q < n^{(n^m)}$ such that $X_q = X_{q+1}$. But

the pair of conditions $X_j \supseteq X_0$ and $X_j = X_{j+1}$ are precisely those of Theorem 2.1. Therefore, $X_q = \mathrm{Clo}_m(\mathbf{A})$.

The case of a unary algebra deserves special consideration. Suppose that $\mathbf{A}$ is unary (i.e., has rank 1). While it is true that, technically speaking, $\mathbf{A}$ has term operations of arbitrarily large rank, these operations are trivial in the sense that they only depend on one of their variables. For example, if f is a basic operation of $\mathbf{A}$, then the term operation $f[p_1^2]$ has rank 2. However, this operation maps any pair (x, y) to $f(x)$. Thus $f[p_1^2]$ is "essentially unary." Since there are only n^n possible unary operations, one can easily show that for any m, we always have $X_{n^n-1} = \mathrm{Clo}_m(\mathbf{A})$.

Now one cannot help but notice the similarity between the definition of the sets X_j in (6.2) and that of a term given in Definition 2.2. Of course, if t is an m-ary term, then $t^{\mathbf{A}} \in \mathrm{Clo}_m(\mathbf{A})$, hence, for some $j < n^{(n^m)}$, $t^{\mathbf{A}} \in X_j$. Conversely, if we view each m-ary term as a tree, then every member of X_j is of the form $t^{\mathbf{A}}$ for some term t of height at most j. Let us write $\mathrm{ht}(t)$ to denote the height of the term t. Putting these two observations together, for every m-ary term t, there is a term t' such that $t^{\mathbf{A}} = (t')^{\mathbf{A}}$ and $\mathrm{ht}(t') < n^{(n^m)}$. Following up on our earlier observation, in the special case that $\mathbf{A}$ is unary, the bound on the height of t' can be reduced to n^n no matter what the value of m.

THEOREM 6.3. *Let $\mathbf{A}$ and $\mathbf{B}$ be finite algebras of the same similarity type. Assume that the cardinalities of $\mathbf{A}$ and $\mathbf{B}$ are n and m, respectively. Then the following are equivalent.*

(i) $\mathbf{B} \in \mathbf{V}(\mathbf{A})$.

(ii) *For every pair of terms s and t, each of height at most $n^{(n^m)}$, if $\mathbf{A}$ satisfies the identity $s \approx t$, then so does $\mathbf{B}$.*

(iii) $\mathbf{B}$ *is a homomorphic image of the algebra* $\mathbf{Clo}_m(\mathbf{A})$.

If $\mathbf{A}$ and $\mathbf{B}$ are unary algebras, then the bound $n^{(n^m)}$ in (ii) *can be reduced to n^n.*

Proof. That (i) implies (ii) follows from (2.2). So assume (ii). Enumerate the elements of B as $b_1, \ldots, b_m$. Define a function $\psi\colon \mathrm{Clo}_m(\mathbf{A}) \to B$ as follows. For each $g \in \mathrm{Clo}_m(\mathbf{A})$, choose a term u such that $\mathrm{ht}(u) < n^{(n^m)}$ and $u^{\mathbf{A}} = g$, and define $\psi(g) = u^{\mathbf{B}}(b_1, \ldots, b_m)$. To see that this is well defined, suppose that s is another term with the properties $s^{\mathbf{A}} = g$ and $\mathrm{ht}(s) < n^{(n^m)}$. Then $s^{\mathbf{A}} = g = u^{\mathbf{A}}$, so $\mathbf{A}$ satisfies the identity $s \approx u$. Therefore, by (ii), $\mathbf{B}$ also satisfies the identity $s \approx u$, hence $s^{\mathbf{B}} = u^{\mathbf{B}}$.

The function ψ is surjective since for every $i \leq m$, $b_i = \psi(p_i^m)$. In order to prove (iii), it remains to show that ψ is a homomorphism. So let $g_1, \ldots, g_k \in \mathrm{Clo}_m(\mathbf{A})$, and let f be a basic k-ary operation symbol. There are terms $s_1, s_2, \ldots, s_k$ and t, all of height less than $n^{(n^m)}$ such that $g_i = s_i^{\mathbf{A}}$ for $i = 1, 2, \ldots, k$, and $f^{\mathbf{A}}[g_1, \ldots, g_k] = t^{\mathbf{A}}$. Let r be the term $f(s_1, \ldots, s_k)$. We need to verify that $\psi\big(f^{\mathbf{A}}[g_1, \ldots, g_k]\big) = f^{\mathbf{B}}\big(\psi(g_1), \ldots, \psi(g_k)\big)$. Note that $\mathrm{ht}(r) = 1 + \max_i \mathrm{ht}(s_i) \leq n^{(n^m)}$. Also, $r^{\mathbf{A}} = t^{\mathbf{A}}$ since

$$r^{\mathbf{A}} = f^{\mathbf{A}}[s_1^{\mathbf{A}}, \ldots, s_k^{\mathbf{A}}] = f^{\mathbf{A}}[g_1, \ldots, g_k] = t^{\mathbf{A}}.$$

Therefore, the equation $r \approx t$ holds in $\mathbf{A}$, hence in $\mathbf{B}$, because of the bounds on the heights. Thus $r^{\mathbf{B}} = t^{\mathbf{B}}$. Now, writing $\mathbf{b} = (b_1, \ldots, b_m)$, we compute

$$\begin{aligned}\psi\big(f^{\mathbf{A}}[g_1, \ldots, g_k]\big) &= \psi(t^{\mathbf{A}}) = t^{\mathbf{B}}(\mathbf{b}) = r^{\mathbf{B}}(\mathbf{b}) \\ &= f^{\mathbf{B}}\big(s_1^{\mathbf{B}}(\mathbf{b}), \ldots, s_k^{\mathbf{B}}(\mathbf{b})\big) = f^{\mathbf{B}}\big(\psi(g_1), \ldots, \psi(g_k)\big).\end{aligned}$$

Finally, assume (iii). $\mathbf{Clo}_m(\mathbf{A})$ is a subalgebra of $\mathbf{A}^{A^m}$; consequently, it lies in $\mathbf{V}(\mathbf{A})$. Since every variety is closed under homomorphic images, $\mathbf{B} \in \mathbf{V}(\mathbf{A})$ as well. □

1. $s^A \leftarrow t^A \leftarrow f_0^{\mathbf{A}}$; $\quad s^B \leftarrow t^B \leftarrow f_0^{\mathbf{B}}$.
2. for $i = 1$ to n^n do
3. guess $j, \ell \in \{0, 1, \ldots, k\}$
4. $s^A \leftarrow f_j^{\mathbf{A}} \circ s^A$; $\quad s^B \leftarrow f_j^{\mathbf{B}} \circ s^B$; $\quad t^A \leftarrow f_\ell^{\mathbf{A}} \circ t^A$; $\quad t^B \leftarrow f_\ell^{\mathbf{B}} \circ t^B$
5. if $\big((\forall x, y \in A)\ (s^A(x) = t^A(y))$ and $(\exists x, y \in B)\ (s^B(x) \neq t^B(y))\big)$ or $\big((\forall x \in A)\ (s^A(x) = t^A(x))$ and $(\exists x \in B)\ (s^B(x) \neq t^B(x))\big)$ then **accept**.

ALGORITHM 6.1. *Testing* $(\mathbf{A}, \mathbf{B}) \notin$ VAR-MEM^1.

It is worthwhile to extract just a bit more information from the proof of Theorem 6.3. Notice that in the proof of (ii) implies (iii), we began with an arbitrary enumeration of the elements of B and constructed a homomorphism from $\mathbf{Clo}_m(\mathbf{A})$ onto $\mathbf{B}$. In the language of universal algebra, this is equivalent to the assertion that the algebra $\mathbf{Clo}_m(\mathbf{A})$ is *freely generated* by $X_0 = \{\, p_i^m : 1 \leq i \leq m \,\}$. For our purposes, we can express this property as the following corollary.

COROLLARY 6.4. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be algebras as in Theorem* 6.3. *The following are equivalent.*

(i) $\mathbf{B} \in \mathbf{V}(\mathbf{A})$.

(ii) *For some bijection* ψ_0 *of* X_0 *with* B, *there exists a homomorphism* ψ *from* $\mathbf{Clo}_m(\mathbf{A})$ *to* $\mathbf{B}$ *such that* $\psi{\restriction}_{X_0} = \psi_0$.

(iii) *For every bijection* ψ_0 *of* X_0 *with* B, *there exists a homomorphism* ψ *from* $\mathbf{Clo}_m(\mathbf{A})$ *to* $\mathbf{B}$ *such that* $\psi{\restriction}_{X_0} = \psi_0$.

Theorem 6.3 suggests an approach that can be used to test the condition $\mathbf{B} \notin \mathbf{V}(\mathbf{A})$: simply *guess* an identity ϵ and check to see whether $\mathbf{A}$ satisfies ϵ while $\mathbf{B}$ fails to satisfy ϵ. This approach seems to be quite effective—at least for unary algebras. For in this case, we have the improved bound n^n in part (ii) of the theorem.

Let us fix a set $F = \{f_1, \ldots, f_k\}$ of operation symbols, each of rank 1. Also, let us add an additional unary operation symbol f_0 which will always be interpreted as the identity operation. This has no effect on the algebras but will save us a subscript in our analysis. A typical term over F is of the form $f_{i_\ell} f_{i_{\ell-1}} \cdots f_{i_2} f_{i_1}(x)$, where $i_1, i_2, \ldots, i_\ell \in \{0, 1, \ldots, k\}$. The height of this term is ℓ. Since each term involves only one variable, every identity is of one of two possible forms:

$$s(x) \approx t(x) \quad \text{or} \quad s(x) \approx t(y).$$

Notice that the second of these is quite degenerate since it requires that the term operations corresponding to s and t both be constant and, in fact, the same constant. Nevertheless, it must be considered in the analysis.

Now suppose that $\mathbf{A}$ and $\mathbf{B}$ are algebras of type F and of cardinalities n and m, respectively. Algorithm 6.1 is a nondeterministic algorithm that accepts the pair $(\mathbf{A}, \mathbf{B})$ if and only if $\mathbf{B} \notin \mathbf{V}(\mathbf{A})$.

How much space is used by this algorithm? Let $p = \max(n, m)$. Each of the four unary operations can be represented as a vector of length p. Each such vector requires $p \log(p) \leq p^2$ bits. We also need space for the counter i, which ranges from 0 to n^n. Since $\log(n^n) = n \log(n) \leq p^2$, i requires another p^2 bits. It follows that the total amount of space required is on the order of p^2 bits.

What is the size of the input? The algebra $\mathbf{A}$ requires $\log(n) + kn \log(n) > n$ bits. Similarly $\mathbf{B}$ requires at least m bits. The total input size is at least $n + m > p$ bits. It follows that our algorithm's space requirements are bounded above by the square of the size of the input.

1. Create empty table for ψ. Fill in ψ_0.
2. for $j = 0$ to $n^{(n^m)} - 2$ do
3. for $f \in F$ do (*let* $k = \rho(f)$)
4. for $g_1, \ldots, g_k \in X_j$ do
5. $b \leftarrow f^{\mathbf{B}}(\psi(g_1), \ldots, \psi(g_k))$
6. $b' \leftarrow \psi(f^{\mathbf{A}}[g_1, \ldots, g_k])$
7. if $b' = \varnothing$ then insert b into the position for b'
8. else if $b' \neq b$ then
9. **reject**
10. **accept**

ALGORITHM 6.2. *Testing* $(\mathbf{A}, \mathbf{B}) \in$ VAR-MEM.

THEOREM 6.5. VAR-MEM^1 *and* VAR-EQUIV^1 *lie in* **PSPACE**.

Proof. Algorithm 6.1 can be used to test whether $(\mathbf{A}, \mathbf{B})$ lies in the *complement* of VAR-MEM^1. Since the algorithm is nondeterministic, we get VAR-MEM$^1 \in$ **co-NPSPACE**. But from Savitch's theorem [26], **NPSPACE** = **PSPACE**. Furthermore, every deterministic class is closed under complements, so **co-PSPACE** = **PSPACE**. Thus VAR-MEM$^1 \in$ **PSPACE**. Now it follows from the relationships in (6.1) that VAR-EQUIV^1 lies in **PSPACE** as well. □

It is not clear whether this is the best possible bound for these two problems. We leave it as an open question.

PROBLEM 6.6. *Are either* VAR-MEM^1 *or* VAR-EQUIV^1 **PSPACE***-complete?*

One might hope to apply the same techniques used above to the unrestricted problem, VAR-EQUIV. Unfortunately, the resources needed to evaluate an arbitrary term in a given algebra jump dramatically as soon as we allow a binary operation. Our approach instead is to try to construct the homomorphism guaranteed by Theorem 6.3 (iii). However, rather than use the construction given in the proof of that theorem, we will construct a homomorphism ψ recursively, based on (6.2). The best we seem to be able to do is the following hyperexponential bound.

THEOREM 6.7. VAR-EQUIV, VAR-MEM $\in$ **2-EXPTIME**.

Proof. Let $|A| = n > 1$ and $|B| = m$. Let F denote the set of basic operation symbols of $\mathbf{A}$ and $\mathbf{B}$, and let r be the maximum rank of the members of F. In light of Theorem 6.5, we shall assume that $r \geq 2$. It follows from Corollary 6.4 that we can test $\mathbf{B} \in \mathbf{V}(\mathbf{A})$ by choosing an arbitrary bijection $\psi_0 \colon X_0 \to B$ and extending it, if possible, to a homomorphism $\psi \colon \mathbf{Clo}_m(\mathbf{A}) \to \mathbf{B}$. So we fix any bijection ψ_0.

Let us sketch an algorithm for extending (if possible) ψ_0 to a homomorphism ψ. Create a table with two columns. In the left-hand column, list every m-ary operation of A. For each m-ary operation g, the right-hand column will contain $\psi(g)$ if and when it is defined. To begin with, leave every entry in the right-hand column blank, except that, for each $1 \leq i \leq m$, put $\psi_0(p_i^m)$ in the row containing p_i^m.

Now let j be an integer, $0 \leq j < n^{(n^m)} - 1$, and suppose we have successfully defined ψ on X_j. We attempt to extend ψ to X_{j+1}. For each $f \in F$ and each $g_1, \ldots, g_k \in X_j$ (with $k = \rho(f)$), we proceed as follows. Look up $\psi(g_1), \ldots, \psi(g_k)$ in the table and compute $b = f^{\mathbf{B}}(\psi(g_1), \ldots, \psi(g_k))$. Also, compute the operation $f^{\mathbf{A}}[g_1, \ldots, g_k]$. If the entry in the table corresponding to $f^{\mathbf{A}}[g_1, \ldots, g_k]$ is blank, then insert b and continue. If the entry is already equal to b, continue. But if the table contains some value $b' \neq b$, then we terminate the algorithm and reject. See the pseudocode in Algorithm 6.2.

If the algorithm runs to completion without a rejection, then we have successfully

found the extension of ψ_0. A rejection means that ψ does not exist.

We must analyze the time requirements of Algorithm 6.2. As we noted earlier, the algebra $\mathbf{A}$ will require at least $n^r \log n$ bits to represent. Let S denote the size of the input $(\mathbf{A}, \mathbf{B})$. Then $S \geq n^r + m^r > r$. Notice also that $S \geq |F|$ since each operation requires at least one bit.

To prove the theorem, we must show that the time required to run Algorithm 6.2 on the input $(\mathbf{A}, \mathbf{B})$ is bounded above by a function of the form $2^{2^{p(S)}}$ for some polynomial p. Since $O\big(2^{2^{\text{poly}}}\big)$ is closed under sums and products, it suffices to check that we can bound the time required by each step of the algorithm by a member of $O\big(2^{2^{\text{poly}}}\big)$. Observe first that

$$\log(n^{m+1}) = (m+1)\log n \leq (m+1)n \leq S,$$

and hence $n^{m+1} \leq 2^S$. Similarly,

$$\log n^{(n^m)} = n^m \log n \leq n^{m+1} \leq 2^S,$$

so

$$n^{(n^m)} \leq 2^{2^S}. \tag{6.3}$$

Let us go step-by-step through Algorithm 6.2. The initial setup (step 1) requires time on the order of n^m to fill in each entry. Since there are $n^{(n^m)}$ entries, the total time required for this step lies in $O\big(2^{2^{\text{poly}}}\big)$ by inequality (6.3). Steps 2–7 constitute a triply-nested loop. The total number of iterations is at most the product of the upper bounds on the counters in steps 2, 3, and 4. Step 2 requires $n^{(n^m)} \in O\big(2^{2^{\text{poly}}}\big)$ iterations. Step 3 runs from 1 to $|F|$, and we have already observed that $|F| \leq S$. For each value of j (in step 2), step 4 will loop $|X_j|^k$ times. We have $k = \rho(f) \leq r$ and $|X_j| \leq |\mathrm{Clo}_m(\mathbf{A})| \leq n^{(n^m)}$, so an upper bound on the counter is $(n^{(n^m)})^r \leq 2^{2^{rS}}$. Finally, the time required for each trip through the body of the triple-loop is the sum of the times for steps 5–7. Step 7 is negligible. Step 5 requires r lookups in the table for ψ and one lookup in the table for $f^{\mathbf{B}}$. The former takes time $r \cdot n^{(n^m)} \leq S \cdot 2^{2^S}$ which is clearly in $O\big(2^{2^{\text{poly}}}\big)$. The latter requires $m^r \leq S$ program steps. Step 6 is the reverse of step 5: one lookup in ψ (time $\leq n^{(n^m)}$) and $(r+1)n^m$ lookups in operation tables. Thus the running time for the entire algorithm lies in $O\big(2^{2^{\text{poly}}}\big)$. □

Theorem 6.7 provides a rather disappointing bound for VAR-EQUIV. It is also somewhat surprising that there seems to be such a dramatic (i.e., exponential) difference in the complexities of QVAR-EQUIV and VAR-EQUIV (see Table 4.1). Perhaps one can do better.

PROBLEM 6.8. *Is* VAR-EQUIV *complete for* **2-EXPTIME**? *Is* VAR-EQUIV *in* **EXPSPACE**?

As we mentioned earlier, Székely proved in [28] that VAR-EQUIV is **NP**-hard. This result can also be obtained using the construction in Theorem 5.5; see especially the equivalences in (5.2).

7. Term-equivalence. Recall that the algebras $\mathbf{A}$ and $\mathbf{B}$ are term-equivalent if $A = B$ and $\mathrm{Clo}(\mathbf{A}) = \mathrm{Clo}(\mathbf{B})$. From a universal algebraic standpoint, term-equivalent algebras are generally interchangeable. When considering the complexity of the problem TERM-EQUIV, it is convenient, once again, to consider a slightly different problem. Thus we define the problem CLO-MEM to consist of all pairs (F, g) in which $F \cup \{g\}$

is a set of operations on some finite set A and $g \in \mathrm{Clo}^A(F)$. The problem CLO-MEM^1 is similar, but we require all of the operations in $F \cup \{g\}$ to be unary. For the problem CLO-MEM$_2$, the set A has cardinality two.

Historically, CLO-MEM was the first problem, of those discussed in this paper, to be considered in the literature. Kozen proved in 1977 [18] that CLO-MEM^1 is complete for **PSPACE**. In 1982, Friedman proved that CLO-MEM is complete for **EXPTIME** [8]. However, that manuscript was never published. A different proof of this result, as well as a proof that TERM-EQUIV is complete for **EXPTIME**, appears in [1].

Let $\mathbf{A} = \langle A, F\rangle$ and $\mathbf{B} = \langle B, G\rangle$. Then

$$\begin{array}{c} (\mathbf{A}, \mathbf{B}) \in \text{TERM-EQUIV} \\ \Updownarrow \\ A = B \;\&\; (\forall g \in G)(\forall f \in F)\ \big((F, g), (G, f) \in \text{CLO-MEM}\big). \end{array} \tag{7.1}$$

Conversely, if $F \cup \{g\}$ is a set of operations on A, then

$$(F, g) \in \text{CLO-MEM} \iff \big(\langle A, F\rangle, \langle A, F \cup \{g\}\rangle\big) \in \text{TERM-EQUIV}. \tag{7.2}$$

Of course, similar relationships hold for the unary and two-element variants of these problems.

Now it follows easily from (7.2) that CLO-MEM is log-space reducible to TERM-EQUIV. This is true for the general, unary, and two-element variants of the problems. However, a reduction in the other direction is a bit problematic. Let us first consider the general case. Given a pair $(\mathbf{A}, \mathbf{B})$ of size S, equivalence (7.1) tells us that we can test $(\mathbf{A}, \mathbf{B}) \in$ TERM-EQUIV by making several calls to an algorithm for CLO-MEM. The input to each such call will certainly have size at most S hence will run in time at most $2^{p(S)}$ for some polynomial p. Furthermore, there will clearly be at most S such calls. Hence a bound on the running time for TERM-EQUIV will be $S \cdot 2^{p(S)}$ which is still exponential in S. Combining our observations, we conclude that the (general) problem TERM-EQUIV is complete for **EXPTIME** (see Table 4.1).

For CLO-MEM^1 and CLO-MEM$_2$, we need to argue a bit differently, since we will be interested in a space-bound. Let C denote one of these two problems, and let T denote the corresponding term-equivalence problem. Suppose we have an algorithm for C that runs in space $f(x)$ on an input of size x. As is commonplace, we assume that f is a monotonically increasing function. In applying (7.1) to test $(\mathbf{A}, \mathbf{B}) \in$ T, the first call to C will require space bounded above by $O(f(S))$. But subsequent calls to C can reuse the same space. Hence the total space requirement for T is on the order of $\log S + f(S)$. (The $\log S$ term accounts for some counters.)

For the specific case C = CLO-MEM^1, Kozen's result tells us that the function f is a polynomial, so we conclude that TERM-EQUIV^1 is complete for **PSPACE**, as we assert in Table 4.1. When $C =$ CLO-MEM$_2$, we can certainly make the following claim.

LEMMA 7.1. *If $f(S) \in O(\log S)$, then both* CLO-MEM$_2$ *and* TERM-EQUIV$_2$ *lie in* **NL**.

So our remaining task is to prove that $f(S)$ is indeed on the order of $\log S$. In other words, we must find a nondeterministic algorithm for CLO-MEM$_2$ that runs in log-space. To do this, we will take advantage of the very detailed description of the lattice of clones on $\{0, 1\}$ that was discovered by Post in 1941 [24]. This will allow us to check the condition $(F, g) \in$ CLO-MEM$_2$ using a finite number of nondeterministic tests, each of which uses very little space. There have been several more recent

treatments of Post's results. The reader might wish to consult the first chapter of Pippenger's book [23]. All we really need here is a description of the completely meet-irreducible members of the lattice of clones. For this we mostly follow the discussion given in Szendrei [30, pp. 36ff].

Let A be any set, and let k be a positive integer. A subset θ of A^k is called a k-ary *relation on* A. If f is an n-ary operation on A, we say that f *preserves* θ, ($f \mid\!: \theta$) if for all arrays $\langle\, a_{ij} : 1 \le i \le n,\ 1 \le j \le k \,\rangle$ we have

$$(\forall i \le n\ \ \mathbf{a}^i \in \theta) \implies \langle f(\mathbf{a}_1), \ldots, f(\mathbf{a}_k)\rangle \in \theta,$$

where $\mathbf{a}_j = \langle a_{1j}, \ldots, a_{nj}\rangle$ and $\mathbf{a}^i = \langle a_{i1}, \ldots, a_{ik}\rangle$. The relationship $f \mid\!: \theta$ is equivalent to asserting that θ is a subalgebra of the algebra $\langle A, f\rangle^k$. One way to visualize this is to think of the $\langle a_{ij}\rangle$ as forming an $n \times k$ matrix. If each row of the matrix constitutes an element of θ, then the row obtained by applying f to each column is again a member of θ. We extend this notation to multiple operations by defining $F \mid\!: \theta$ if and only if for every $f \in F$ we have $f \mid\!: \theta$.

For a fixed relation θ we define

$$\mathcal{F}^A(\theta) = \{\, f : f \text{ is an operation on } A \text{ and } f \mid\!: \theta \,\}.$$

More generally, if Θ is a set of relations on A (of various ranks), then

$$\mathcal{F}^A(\Theta) = \bigcap_{\theta \in \Theta} \mathcal{F}^A(\theta).$$

We usually omit the superscript "A" if no confusion will result. It is easy to check that for any set A and family Θ, $\mathcal{F}(\Theta)$ is always a clone on A. However, more to the point for us, for every *finite* set A, every clone on A is of the form $\mathcal{F}(\Theta)$ for some (not necessarily finite) family Θ. See [30, Corollary 1.4].

We now restrict our attention to $A = \{0, 1\}$. We define the following relations on A.

$$\begin{aligned}
\nu &= \{\, \langle 0,1\rangle, \langle 1,0\rangle \,\}, & \lambda &= \{\, \langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle \,\},\\
\mu &= \{\, \langle x,y,z\rangle : z = x \wedge y \,\}, & \chi &= \{\, \langle x,y,z\rangle : z = x \vee y \,\},\\
\varepsilon &= \{\, \langle x,y,z\rangle : x = y \text{ or } y = z \,\}, & \sigma &= \{\, \langle x,y,z,w\rangle : x \oplus y = z \oplus w \,\},\\
\kappa_m &= \{0,1\}^m - \{\langle 1,1,\ldots,1\rangle\} \quad \text{for } m \ge 1,\\
\tilde{\kappa}_m &= \{0,1\}^m - \{\langle 0,0,\ldots,0\rangle\} \quad \text{for } m \ge 1.
\end{aligned}$$

Let $\Sigma_0 = \{\nu, \mu, \chi, \lambda, \varepsilon, \sigma\}$ and, for every $m > 0$, $\Sigma_m = \Sigma_0 \cup \{\, \kappa_j, \tilde{\kappa}_j : j \le m \,\}$. Set $\Sigma = \bigcup_{m \ge 1} \Sigma_m$. From Post's analysis, we get the following theorem.

THEOREM 7.2 (Post). *Let C be a clone on $\{0,1\}$. Then for some family $\Theta \subseteq \Sigma$, $C = \mathcal{F}(\Theta)$.*

Let θ be a fixed k-ary relation on $\{0,1\}$. We first show that there is a simple nondeterministic algorithm for the *complement* of the problem

$$\textsc{Pres}(\theta) = \{\, f : f \text{ an operation on } \{0,1\} \text{ and } f \mid\!: \theta \,\}.$$

Given an n-ary operation f, we guess an $n \times k$ matrix and accept if each row of the matrix lies in θ but the result of applying f to the columns fails to lie in θ. A more formal description is given in Algorithm 7.1.

LEMMA 7.3. *For any fixed relation θ on $\{0,1\}$, $\textsc{Pres}(\theta) \in \mathbf{NL}$.*

1. Let $n = \rho(f)$
2. Guess $a_{ij} \in \{0,1\}$, $1 \le i \le n$, $1 \le j \le k$
3. For $i = 1$ to n do
4. if $\mathbf{a}^i \notin \theta$ then **reject**
5. if $\langle f(\mathbf{a}_1), \ldots, f(\mathbf{a}_k)\rangle \notin \theta$ then **accept**

ALGORITHM 7.1. $f \notin \text{PRES}(\theta)$.

---

1. Let $n = \rho(f)$
2. $\mathbf{x} \leftarrow \langle 1, 1, \ldots, 1\rangle \in \{0,1\}^n$
3. for $i = 1$ to m do
4. guess $\mathbf{a} \in \{0,1\}^n$
5. if $f(\mathbf{a}) = 1$ then $\mathbf{x} \leftarrow \mathbf{x} \wedge \mathbf{a}$ *(coordinatewise conjunction)*
6. else **reject**
7. if $\mathbf{x} = \langle 0, 0, \ldots, 0\rangle$ then **accept**
8. else **reject**

ALGORITHM 7.2. $(f, m) \notin \text{KAPPA}$.

Proof. Algorithm 7.1 provides a test for the complement of PRES(θ). Since **NL** is closed under complements (see [12, 29]), it suffices to show that this algorithm requires only log-space. Since the size of the input (an n-ary operation) is 2^n bits, we must verify that the space requirement of the algorithm lies in $O(n)$. However, the algorithm only requires space for the counters i and j, the Boolean matrix $\langle a_{ij}\rangle$, and k pointers into the table of values for f. Since k is a constant, the total space required is indeed in $O(n)$. □

We will also need a variation of Algorithm 7.1 to handle the κ_m and $\tilde{\kappa}_m$. Algorithm 7.2 gives a procedure that takes as input an operation f and a positive integer m and accepts (nondeterministically) if and only if f does not preserve κ_m. Using the same argument as in Lemma 7.3, we can conclude that

$$\text{KAPPA} = \{\, (f, m) : f \not\colon \kappa_m \,\} \in \mathbf{NL},$$

with an analogous result for $\tilde{\kappa}_m$. Let us remark here that it is not hard to modify Algorithm 7.1 to run in deterministic logarithmic space. However, that modification does not seem to work for KAPPA, so there does not appear to be anything to gain by including the argument here.

Our next task is to put a bound on the size of the family Θ that we need to consider in Theorem 7.2. Notice that, except for the six members of Σ_0, the members of Σ form two infinite sequences, and, furthermore, these two sequences are dual to each other.

LEMMA 7.4. *Let g be an n-ary operation on $\{0,1\}$.*

(i) *For every $m > 1$, if g preserves κ_m, then g preserves κ_{m-1}.*

(ii) *If g preserves κ_n, then g preserves κ_m for all $m > n$.*

The same results hold with $\tilde{\kappa}$ in place of κ.

Proof. Suppose $g \not\colon \kappa_m$. Let $\langle a_{ij}\rangle$ be an $n \times (m-1)$ matrix in which every row is a member of κ_{m-1}. This simply means that no row consists of all 1s. We wish to argue that $\langle g(\mathbf{a}_1), \ldots g(\mathbf{a}_{m-1})\rangle \in \kappa_{m-1}$, i.e., is not all 1s. Create an $n \times m$ matrix by repeating the last column of $\langle a_{ij}\rangle$. Then each row of this new matrix lies in κ_m, so by assumption, the m-tuple $\langle g(\mathbf{a}_1), \ldots, g(\mathbf{a}_{m-1}), g(\mathbf{a}_{m-1})\rangle \in \kappa_m$. Therefore, $\langle g(\mathbf{a}_1), \ldots, g(\mathbf{a}_{m-1})\rangle$ is not equal to $\langle 1, 1, \ldots, 1\rangle$, as desired.

For the second claim, assume that $g \not\colon \kappa_n$ and let $m > n$. Let $\langle a_{ij}\rangle$ be an $n \times m$

```
1. n ← ρ(g)
2. for θ ∈ Σ_n do
3.     if F |: θ then
4.         if not(g |: θ) then reject
5. accept
```

ALGORITHM 7.3. $g \in \mathrm{Clo}(F)$.

matrix in which every row is a member of κ_m. In other words, each row contains a "0." Therefore we can find $1 \le j_1 < j_2 < \cdots < j_n \le m$ such that the $n \times n$ submatrix $\langle a_{ij_k} : 1 \le i \le n,\ 1 \le k \le n\rangle$ retains the property that every row contains a "0." Since $g \mid\!: \kappa_n$, we get $\langle g(\mathbf{a}_{j_1}), \ldots, g(\mathbf{a}_{j_n})\rangle \in \kappa_n$, i.e., for some k, $g(\mathbf{a}_{j_k}) = 0$. Therefore, $\langle g(\mathbf{a}_1), \ldots, g(\mathbf{a}_n)\rangle \in \kappa_m$. □

We can now combine Theorem 7.2 and Lemmas 7.3 and 7.4 to put a finite bound on the work needed in order to test the relationship $g \in \mathrm{Clo}(F)$.

THEOREM 7.5. *Let F be a set of operations on $\{0,1\}$, and let g be an n-ary operation on $\{0,1\}$. Then $g \in \mathrm{Clo}(F)$ if and only if for every $\theta \in \Sigma_n$, $F \mid\!: \theta \implies g \mid\!: \theta$.*

Proof. Suppose first that $g \in \mathrm{Clo}(F)$. Let θ be any relation such that $F \mid\!: \theta$. Since $\mathcal{F}(\theta)$ is a clone, $F \subseteq \mathcal{F}(\theta)$, and $\mathrm{Clo}(F)$ is, by definition, the *smallest* clone containing F, we obtain $\mathrm{Clo}(F) \subseteq \mathcal{F}(\theta)$; hence $g \mid\!: \theta$.

Conversely, suppose the condition holds. By Theorem 7.2, there is some subset Θ of Σ such that $\mathrm{Clo}(F) = \mathcal{F}(\Theta)$. We need to show that for every $\theta \in \Theta$, $g \mid\!: \theta$. If $\theta \in \Sigma_n$, then, since $F \subseteq \mathrm{Clo}(F)$, we have $F \mid\!: \theta$, so $g \mid\!: \theta$ by assumption. So suppose that $\theta \in \Sigma - \Sigma_n$. Then $\theta = \kappa_m$ or $\theta = \tilde{\kappa}_m$ for some $m > n$. Without loss of generality, assume the former. Again, since $F \subseteq \mathrm{Clo}(F) = \mathcal{F}(\Theta)$, we get $F \mid\!: \kappa_m$. By Lemma 7.4(i), $F \mid\!: \kappa_n$; hence $g \mid\!: \kappa_n$ by assumption. Therefore, by Lemma 7.4(ii), $g \mid\!: \kappa_m$. □

Since the family Σ_n is finite, Theorem 7.5 provides an algorithm for testing $g \in \mathrm{Clo}(F)$. Pseudocode is given in Algorithm 7.3. Let us analyze the space requirements for this algorithm. The size of the input is at least 2^n since that is the size of g, and also at least $|F|$ since each operation takes up at least 1 bit. Line 1 can be implemented by setting θ first to each member of Σ_0, followed by $\kappa_1, \tilde{\kappa}_1, \kappa_2, \ldots, \tilde{\kappa}_n$ using a loop. The amount of space needed to manage the loop is in $O(\log n)$.

The notation "$F \mid\!: \theta$" is shorthand for the following code fragment.

```
x ← True
for f ∈ F do
    if not(f |: θ) then x ← False
return x
```

When $\theta \in \Sigma_0$, the test $f \mid\!: \theta$ is implemented using an algorithm for PRES(θ). When $\theta = \kappa_m$ or $\tilde{\kappa}_m$, we use KAPPA or its dual. Clearly, the entire algorithm requires only a couple of counters besides the calls to PRES and KAPPA and hence lies in **NL**. Combining this with Lemma 7.1, we have proved the following theorem.

THEOREM 7.6. *Both* CLO-MEM$_2$ *and* TERM-EQUIV$_2$ *lie in* **NL**.

Once again, it is not clear that these are optimal results, so we pose a problem.

PROBLEM 7.7. *Is* TERM-EQUIV$_2$ *complete for* **NL**? *Does it lie in* **L**?

Appendix. Proof of Theorem 6.1. This appendix provides a proof of Theorem 6.1. In fact, we shall prove a slightly stronger theorem which allows us to conclude that both VAR-EQUIV$_2$ and VAR-MEM$_2$ are members of **L**. The line of argument we

follow was suggested to us by J. Berman. An alternate proof can be obtained from [15, Corollary 2.5], using the fact that every two-element algebra is both term-minimal and strictly simple.

An algebra is called *congruence distributive* if its lattice of congruences is distributive. The algebra is *congruence permutable* if for every pair of congruences α and β we have $\alpha \circ \beta = \beta \circ \alpha$. Here,

$$\alpha \circ \beta = \{\, (x, z) : (\exists y)\ (x, y) \in \alpha \ \&\ (y, z) \in \beta \,\}.$$

A variety is called congruence distributive (respectively, permutable) if every member is congruence distributive (permutable). See [5, Definition 5.8] for a discussion of these two important properties. We also require the following result.

LEMMA A.1 (see Berman [2, Lemma 1]). *Let* $\mathbf{A}$ *be an algebra with universe* $\{0, 1\}$. *Then at least one of the following hold.*

(i) $\mathbf{V}(\mathbf{A})$ *is congruence distributive.*

(ii) $\mathbf{V}(\mathbf{A})$ *is congruence permutable.*

(iii) *Every basic operation of* $\mathbf{A}$ *is one of the following: a conjunction of variables, a disjunction of variables, negation of a variable, a projection operation, or a constant operation.*

An n-ary operation f is "a conjunction of variables" if $f(x_1, \dots, x_n) = x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k}$, where $2 \le k \le n$, $1 \le i_1 < i_2 < \cdots < i_k \le n$ and "$\wedge$" denotes the usual logical "and" operation on $\{0, 1\}$. A disjunction of variables is similar. We write $0' = 1$ and $1' = 0$. The function $f(x_1, \dots, x_n) = x_i'$ is the negation of the variable x_i.

Let Δ denote the equivalence relation on $\{0, 1\}^2$ that identifies the pairs $(0, 0)$ and $(1, 1)$ and the pairs $(0, 1)$ and $(1, 0)$. One way to think of Δ is as the kernel of the exclusive-or function $\{0, 1\}^2 \to \{0, 1\}$. For a fixed algebra $\mathbf{A}$ on $\{0, 1\}$, Δ may or may not be a congruence relation on $\mathbf{A}^2$. By using Lemma A.1, it is not hard to see that Δ is a congruence if and only if $\mathbf{A}$ is Abelian. (See [20, section 4.13], especially Theorem 4.152.) In the case that Δ is indeed a congruence on $\mathbf{A}^2$, we define $\mathbf{A}_\nabla$ to be the algebra $\mathbf{A}^2/\Delta$. Note that it is possible for $\mathbf{A}$ and $\mathbf{A}_\nabla$ to be isomorphic. In fact, $\mathbf{A} \cong \mathbf{A}_\nabla$ if and only if there is an element $e \in \{0, 1\}$ such that $\{e\}$ is a subalgebra of $\mathbf{A}$. (To see this, show that the function $x \mapsto (x, e)/\Delta$ is an isomorphism.) As we shall see, in the case that $\mathbf{A}_\nabla$ exists and is not isomorphic to $\mathbf{A}$, $\mathbf{V}(\mathbf{A}_\nabla)$ is the unique proper, nontrivial subvariety of $\mathbf{V}(\mathbf{A})$.

THEOREM A.2. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be algebras on* $\{0, 1\}$ *of the same similarity type, and suppose that* $\mathbf{B} \in \mathbf{V}(\mathbf{A})$. *Then either* $\mathbf{B} \cong \mathbf{A}$ *or* $\mathbf{B} \cong \mathbf{A}_\nabla$.

Proof. Since both $\mathbf{A}$ and $\mathbf{B}$ have cardinality two, they are *strictly simple* algebras, i.e., they are simple and have no proper, nontrivial subalgebras. Suppose first that $\mathcal{V} = \mathbf{V}(\mathbf{A})$ is congruence distributive. Since $\mathbf{B} \in \mathbf{V}(\mathbf{A})$ and $\mathbf{B}$ is simple, we obtain $\mathbf{B} \in \mathbf{HS}(\mathbf{A})$ from Jónsson's lemma [5, Theorem 6.8]. From the strict simplicity of $\mathbf{A}$ we get $\mathbf{B} \cong \mathbf{A}$. Notice that we never obtain the conclusion $\mathbf{B} \cong \mathbf{A}_\nabla$ here. This is not surprising since a congruence distributive variety contains no nontrivial Abelian algebras.

For the remainder of the proof, we assume that $\mathcal{V}$ is not congruence distributive. Suppose that $\mathcal{V}$ is congruence permutable. Then, in particular, $\mathcal{V}$ is congruence modular [5, Theorem 5.10], so by [7, Theorem 12.1] and the strict simplicity of $\mathbf{A}$, $\mathcal{V}$ is an Abelian variety, and hence $\mathbf{A}$ is an Abelian algebra. Therefore, since $\mathbf{B} \in \mathbf{V}(\mathbf{A})$, from [7, Theorem 12.4] we deduce that either $\mathbf{B} \cong \mathbf{A}$ or $\mathbf{B} \cong \mathbf{A}_\nabla$.

So now we are reduced to the case that $\mathcal{V}$ is neither congruence distributive nor congruence permutable. This forces us into case (iii) of Lemma A.1. Each of the

TABLE A.1

$\mathbf{A}$	$\mathbf{A}_\nabla$
$\langle\{0,1\}, '\rangle$	$\langle\{0,1\}, p_1^1\rangle$
$\langle\{0,1\}, ', 0\rangle$	$\langle\{0,1\}, p_1^1, 0\rangle$
$\langle\{0,1\}\rangle$	
$\langle\{0,1\}, 0\rangle$	
$\langle\{0,1\}, 1\rangle$	
$\langle\{0,1\}, 0, 1\rangle$	$\langle\{0,1\}, 0, 0\rangle$

basic operations of $\mathbf{A}$ and of $\mathbf{B}$ must be of one of the types described there. Suppose that $\mathbf{A}$ contains a conjunction of variables among its basic operations. If $\mathbf{A}$ also contains either a disjunction of variables or a negation operation, then $\mathbf{A}$ will have a lattice reduct and consequently $\mathcal{V}$ will be congruence distributive, which contradicts our assumption. Therefore, if $\mathbf{A}$ contains a conjunction of variables, then $\mathbf{A}$ is term-equivalent to a semilattice, possibly with one or two constant operations. However, it is well known that the variety generated by such an algebra has, up to isomorphism, a unique two-element member. Thus we conclude that in this case too, $\mathbf{B} \cong \mathbf{A}$. Obviously, we arrive at the same conclusion if we assume at the outset that $\mathbf{A}$ has a basic operation that is a disjunction of variables.

Now suppose that neither $\mathbf{A}$ nor $\mathbf{B}$ has a conjunction or a disjunction of variables among its basic operations. By looking at the available operations in Lemma A.1 (iii), we conclude that $\mathbf{A}$ is term-equivalent to one of the algebras listed in Table A.1. It is easy to check that for each algebra $\mathbf{A}$ in the table, $\mathbf{A}_\nabla$ exists. Furthermore, in each case, it is obvious that $\mathbf{A}$ and $\mathbf{A}_\nabla$ are the only two-element algebras in $\mathbf{V}(\mathbf{A})$. Using these observations, we obtain the conclusion in the remaining cases. □

Theorem 6.1 follows immediately from Theorem A.2. For if $\mathbf{B}$ and $\mathbf{A}$ generate the same variety, then we certainly have $\mathbf{B} \in \mathbf{V}(\mathbf{A})$, so by Theorem A.2, either $\mathbf{B} \cong \mathbf{A}$ or $\mathbf{B} \cong \mathbf{A}_\nabla$. However, we claim that if $\mathbf{A} \ncong \mathbf{A}_\nabla$, then $\mathbf{A}_\nabla$ always generates a proper subvariety of $\mathbf{V}(\mathbf{A})$, contrary to the assumption that $\mathbf{V}(\mathbf{B}) = \mathbf{V}(\mathbf{A})$. The easiest way to see this is probably just to treat the various cases that arose in Theorem A.2 individually. The case that $\mathbf{V}(\mathbf{A})$ is congruence permutable is discussed in [7]. For the algebras listed in Table A.1, it is a simple matter to find an identity satisfied by $\mathbf{A}_\nabla$ that fails in $\mathbf{A}$. For example, consider the first line of the table. The similarity type consists of a single unary operation symbol, f. Then the identity $f(x) \approx x$ separates $\mathbf{A}$ from $\mathbf{A}_\nabla$.

COROLLARY A.3. VAR-MEM$_2 \in \mathbf{L}$.

Proof. From Theorem A.2, $(\mathbf{A}, \mathbf{B}) \in$ VAR-MEM$_2$ if and only if either $\mathbf{B} \cong \mathbf{A}$ or $\mathbf{B} \cong \mathbf{A}_\nabla$. We have already observed that the former condition can be checked in log-space. To check the latter, it is enough to check the following two maps to see if either is a homomorphism.

$$\mathbf{A}^2 \to \mathbf{B} \qquad (x, y) \mapsto x \oplus y,$$
$$\mathbf{A}^2 \to \mathbf{B} \qquad (x, y) \mapsto x \oplus y \oplus 1.$$

Here $x \oplus y$ is the exclusive-or of x and y. □

Acknowledgments. Finally, we would like to thank Joel Berman, Gary Leavens, and Ross Willard for many helpful discussions on this and related topics.

REFERENCES

[1] C. Bergman, D. Juedes, and G. Slutzki, *Computational complexity of term-equivalence*, Internat. J. Algebra Comput., 9 (1999), pp. 113–128.

[2] J. Berman, *A proof of Lyndon's finite basis theorem*, Discrete Math., 29 (1980), pp. 229–233.

[3] G. Birkhoff, *On the structure of abstract algebras*, Proc. Cambr. Philos. Soc., 31 (1935), pp. 433–454.

[4] S. Burris, *Computers and universal algebra: some directions*, Algebra Universalis, 34 (1995), pp. 61–71.

[5] S. Burris and H. P. Sankappanavar, *A Course in Universal Algebra*, Springer-Verlag, New York, 1981.

[6] H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specification* 1, EATCS Monographs on Theoretical Computer Science 6, Springer-Verlag, Berlin, 1985.

[7] R. Freese and R. McKenzie, *Commutator Theory for Congruence Modular Varieties*, London Math. Soc. Lecture Notes Ser. 125, Cambridge University Press, Cambridge, UK, 1987.

[8] H. Friedman, *Function Composition and Intractability* I, Manuscript, 1982.

[9] 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.

[10] G. Grätzer, *Universal Algebra*, 2nd ed., Springer-Verlag, New York, 1979.

[11] Z. Hedrlín and A. Pultr, *On full embeddings of categories of algebras*, Illinois J. Math., 10 (1966), pp. 392–405.

[12] N. Immerman, *Nondeterministic space is closed under complementation*, SIAM J. Comput., 17 (1988), pp. 935–938.

[13] D. Johnson, *A catalog of complexity classes*, in Handbook of Theoretical Computer Science, Vol. A, J. van Leeuwen, ed., Elsevier, Amsterdam, 1990, pp. 69–161.

[14] J. Kalicki, *On comparison of finite algebras*, Proc. Amer. Math. Soc., 3 (1952), pp. 36–40.

[15] K. Kearnes and Á. Szendrei, *A characterization of minimal locally finite varieties*, Trans. Amer. Math. Soc., 349 (1997), pp. 1749–1768.

[16] J. Köbler, U. Schöning, and J. Torán, *The Graph Isomorphism Problem: Its Structural Complexity*, Birkhäuser Boston, Cambridge, MA, 1993.

[17] D. Kozen, *Complexity of finitely presented algebras*, in Proceedings of the Ninth Annual Symposium on the Theory of Computing, ACM, New York, 1977, pp. 164–177.

[18] D. Kozen, *Lower bounds for natural proof systems*, in 18th Annual Symposium on Foundations of Computer Science, Providence, RI, IEEE Computer Society, Los Alamitos, CA, 1977, pp. 254–266.

[19] L. Kučera and V. Trnková, *The computational complexity of some problems in universal algebra*, in Universal Algebra and its Links with Logic, Algebra, Combinatorics and Computer Science, P. Burmeister et al., eds., Heldermann Verlag, Berlin, 1984, pp. 261–289.

[20] R. McKenzie, G. McNulty, and W. Taylor, *Algebras, Lattices, Varieties*, I, Wadsworth and Brooks/Cole, Monterey, CA, 1987.

[21] G. L. Miller, *Graph isomorphism: General remarks*, J. Comput. System Sci., 18 (1979), pp. 128–142.

[22] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, MA, 1994.

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

[24] E. Post, *Two-Valued Iterative Systems of Mathematical Logic*, Princeton University Press, Princeton, NJ, 1941.

[25] D. Sanella and A. Tarlecki, *On observational equivalence and algebraic specification*, J. Comput. System Sci., 34 (1987), pp. 150–178.

[26] W. J. Savitch, *Relationships between nondeterministic and deterministic tape complexities*, J. Comput. System Sci., 4 (1970), pp. 177–192.

[27] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, Boston, MA, 1997.

[28] Z. Székely, *Complexity of the Finite Algebra Membership Problem for Varieties*, Ph.D. thesis, University of South Carolina, Columbia, SC, 1998.

[29] R. Szelepcsényi, *The method of forced enumeration for nondeterministic automata*, Acta Inform., 26 (1988), pp. 279–284.

[30] Á. Szendrei, *Clones in Universal Algebra*, Séminaire de Mathématiques Supérieures, L'Université de Montréal, Montréal, Canada, 1986.

[31] W. Taylor, *Equational logic*, Houston J. Math. Survey, (1979), pp. iii+83.

[32] I. Wegener, *The Complexity of Boolean Functions*, John Wiley and Sons, Chichester, UK, 1987.

[33] J. Wing, *A specifier's introduction to formal methods*, Computer, 23 (1990), pp. 8–24.