COMPUTATIONAL COMPLEXITY OF SOME PROBLEMS INVOLVING CONGRUENCES ON ALGEBRAS

CLIFFORD BERGMAN AND GIORA SLUTZKI

Abstract. We prove that several problems concerning congruences on algebras are complete for nondeterministic log-space. These problems are: determining the congruence on a given algebra generated by a set of pairs, and determining whether a given algebra is simple or subdirectly irreducible. We also consider the problem of determining the smallest fully invariant congruence on a given algebra containing a given set of pairs. We prove that this problem is complete for nondeterministic polynomial time.

Key words and phrases. congruence, simple algebra, nondeterministic log-space, graph accessibility

One of the fundamental constructions in algebra is the formation of quotient structures. Every quotient of an algebra $\mathbf{A}$ is a homomorphic image of $\mathbf{A}$, and conversely, every homomorphic image is isomorphic to a quotient of $\mathbf{A}$. For familiar sorts of algebraic structures such as groups or rings, a quotient is often determined by a special subset, i.e., a normal subgroup or an ideal. But for an arbitrary algebraic structure, it is necessary to describe quotient algebras by means of a more general device called a *congruence relation.*

A congruence relation on an algebra $\mathbf{A}$ is an equivalence relation closed under the operations of $\mathbf{A}$ (see Definition 1.1). The set of congruences on an algebra forms a complete lattice in which the meet operation coincides with intersection. Thus, given a set θ of ordered pairs, we can talk about the smallest congruence containing θ, denoted $\mathrm{Cg}^{\mathbf{A}}(\theta)$.

In this paper we consider the complexity of the problem of determining $\mathrm{Cg}^{\mathbf{A}}(\theta)$, where $\mathbf{A}$ is a finite algebra of finite similarity type. Specifically, we define Gen-Con to be the following problem:

$$\text{Gen-Con} = \{\, \langle \mathbf{A}, \theta, a, b\rangle : a, b \in A,\ \theta \subseteq A^2 \text{ and } (a,b) \in \mathrm{Cg}^{\mathbf{A}}(\theta) \,\}.$$

In this context, the 'problem' is that of determining whether a given quadruple $\langle \mathbf{A}, \theta, a, b\rangle$ is or is not a member of the set Gen-Con. In Theorem 3.5 we prove that Gen-Con is complete for nondeterministic log-space (denoted **NL**).

2000 *Mathematics Subject Classification.* 08A30, 68Q17, 08A35, 05C40.

An abbreviated version of this paper appeared in LICS 2000.

Related to the problem of generating congruences are two others of interest to algebraists. An algebra is called *simple* if it has exactly two congruences, and is *subdirectly irreducible* if it has a smallest nontrivial congruence. Subdirectly irreducible algebras, in particular, are critical to the study of equational classes of algebras. We define

$$\textsc{Simp} = \{\, \mathbf{A} : \mathbf{A} \text{ is a simple algebra} \,\}$$
$$\mathrm{SI} = \{\, \mathbf{A} : \mathbf{A} \text{ is a subdirectly irreducible algebra} \,\}.$$

In Theorem 3.5 we prove that both of these problems are complete for **NL**.

We prove that all three problems lie in **NL** by describing nondeterministic algorithms that run in log-space (see Theorems 2.3 and 2.4). To prove **NL**-hardness, we use reductions from some well-known problems in the theory of directed graphs (Lemma 3.4). The very naturalness of the reductions suggests a close relationship between path connectedness (the graph problem) and that of generating congruences. We are able to apply the same techniques to a problem of Bélohlávek and Chajda [1], to show that testing a subset of an algebra to determine whether it is a class of some congruence can be performed in nondeterministic log-space, see Theorem 2.5.

In the final section of the paper we consider an analogous problem for *fully invariant congruences,* that is, congruences that are preserved by all endomorphisms of the algebra. We show that the problem of determining the fully invariant congruence generated by a set of pairs is **NP**-complete.

Congruence relations play a basic role in several areas of computer science. So basic in fact, that they are often taken for granted and not mentioned explicitly. For example, an implementation of an abstract data type is usually represented by an algebra. (In practice, the algebras used in the theory of data types are multi-sorted, but that causes no theoretical complications.) Typically, this algebra is required to be minimal, i.e., have no proper subuniverses. Under these assumptions, the most natural way to produce additional implementations of the same abstract data type is by forming a quotient algebra of the original. In particular, if **T** represents the "ground term algebra" (see [24]), then every minimal algebra can be obtained as a quotient in this way.

If **A** is an algebra of an appropriate similarity type ("signature", in the terminology of the field) and if E is an equationally defined specification, then **A** has a largest quotient structure, $\mathbf{A}/\psi$, satisfying E. It is easy to show that ψ will be a fully invariant congruence of **A** in this case. Again, by taking $\mathbf{A} = \mathbf{T}$ to be the ground term algebra, $\mathbf{A}/\psi$ will represent the initial semantics of the data type.

Fully invariant congruences are useful in the study of equational theories of algebras. For example, if **A** is an n-generated free algebra in a variety $\mathcal{V}$, then there is a one-to-one correspondence between the fully invariant congruences of **A** and the subclasses of $\mathcal{V}$ defined by sets of identities limited to n variables. More generally, for any algebra **A** and variety $\mathcal{V}$, the reflection of **A** into $\mathcal{V}$ (see [17, page 89]) is of the form $\mathbf{A}/\theta$, where θ is a fully invariant congruence of $\mathbf{A}$. See [23] for a survey of results and applications of fully invariant congruences.

Fully invariant congruences also appear in the study of term rewriting systems. A set of rewrite rules can be considered to be a binary relation $\theta \subseteq T^2$, where T is the set of all terms in some fixed similarity type over a set X of variables. Then the set of all pairs of terms (t, t') that have a common reduct is equal to the fully invariant congruence generated by θ.

A congruence relation on an algebra $\mathbf{A}$ can be considered to be a set of ordered pairs that is simultaneously an equivalence relation on A and a subuniverse of $\mathbf{A}^2 = \mathbf{A} \times \mathbf{A}$. In analogy with Gen-Con, it is natural to define the problem

$$\text{Gen-SubAlg} = \big\{\langle \mathbf{A}, X, a\rangle : X \subseteq A,\ a \in A \text{ and } a \text{ lies in} \\ \text{the subuniverse of } \mathbf{A} \text{ generated by } X\big\}.$$

It is easy to find a reduction of Gen-Con to Gen-SubAlg, see for example, [3, Theorem 5.5]. Thus Gen-Con can be no harder than Gen-SubAlg. However, in [14] Jones and Laaser proved that Gen-SubAlg is complete for **P** (the class of problems solvable in polynomial time). It is known that **NL** is contained in the class of problems solvable in polynomial time, and it is generally believed that the inclusion is proper. Thus Gen-SubAlg is apparently strictly harder than Gen-Con.

Since it lies in **NL**, there is an algorithm for Gen-Con that runs in polynomial time. Our Algorithm 1 is, of course, nondeterministic. But even if it were converted to a deterministic algorithm in the natural way (i.e., by an exhaustive search for a successful computation path), it would not be particularly efficient, running in time proportional to the square, or perhaps even the cube of s, the size of the input. This is because the algorithm repeatedly recomputes numerous quantities, rather than saving them (since the space required to save the information exceeds $O(\log s)$ bits). By contrast, in a recent note [7], R. Freese exhibited an algorithm for Gen-Con that runs in linear time. However, Freese's algorithm uses linear, rather than logarithmic, space. In the 1980s, Demel, Demlová and Koubek [4, 5] presented linear-time algorithms for many of the problems discussed in this paper.

1. Background Material

We provide here only the barest summary of the notions we need from universal algebra and complexity theory. For more details on universal algebra, the reader should consult any of [3, 9, 19], and for computational complexity, [11, 20, 22]. Also, the first two sections of our paper [2] contain a more extensive discussion of both of these topics.

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

A. The set A is called the *universe* and F the set of *basic operations* of the algebra $\mathbf{A}$. If F is finite, the algebra is said to be of *finite similarity type.* A *subuniverse* of $\mathbf{A}$ is a subset closed under the basic operations.

Definition 1.1. Let $\mathbf{A} = \langle A, F\rangle$ be an algebra. A *congruence* on $\mathbf{A}$ is a set $\psi \subseteq A \times A$ such that

- ψ is an equivalence relation on A, and
- $\big(\forall f \in F\big)\big(\forall (a_1, b_1), \dots, (a_n, b_n) \in \psi\big)$
  $(f(a_1, \dots, a_n),\, f(b_1, \dots, b_n)) \in \psi$.
  Here, the rank of f is n.

The set of congruences on $\mathbf{A}$ is denoted $\mathrm{Con}(\mathbf{A})$. The smallest element of this set is the identity relation $\delta_A = \{\, (x, x) : x \in A \,\}$, while the largest is the relation A^2. It is easy to see that $\mathrm{Con}(\mathbf{A})$ is closed under arbitrary nonempty intersections. Given a set $\theta \subseteq A \times A$ we define

$$\mathrm{Cg}^{\mathbf{A}}(\theta) = \bigcap \{\, \psi \in \mathrm{Con}(\mathbf{A}) : \theta \subseteq \psi \,\},$$

called the *congruence on* $\mathbf{A}$ *generated by* θ.

A nontrivial algebra $\mathbf{A}$ is called *simple* if $\mathrm{Con}(\mathbf{A}) = \{\delta_A, A^2\}$, while $\mathbf{A}$ is called *subdirectly irreducible* if there is a congruence $\mu \neq \delta_A$ such that for all $\psi \in \mathrm{Con}(\mathbf{A}) - \{\delta_A\}$, $\psi \supseteq \mu$. The congruence μ is called the *monolith* of $\mathbf{A}$.

The formal definitions of complexity theory are usually given in terms of *languages,* i.e., sets of finite strings over some fixed alphabet. Associated with each language L is a decision problem: Given a string x, decide whether $x \in L$. The amount of time or space required by a Turing machine to perform this computation generally depends on the length of the input string x. The language L is said to be computable in *polynomial time* if there is a polynomial p such that some deterministic Turing machine can decide whether an input string x of length s lies in L in time $O\big(p(s)\big)$. The set of all languages computable in polynomial time is denoted $\mathbf{P}$.

The set $\mathbf{NL}$ consists of those languages computable by a *nondeterministic* Turing machine whose space requirements are in $O(\log s)$, for an input of length s. We say that such a problem is computable in *nondeterministic log-space.* Similarly, $\mathbf{NP}$ denotes the set of languages computable in nondeterministic polynomial time.

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

Given two problems A and B, we say that A is *log-space reducible* to B (A $\leq_{\log}$ B) if there is a function f, computable in (deterministic) log-space, such that for every instance x of A, $x \in \mathrm{A} \iff f(x) \in \mathrm{B}$. B is said to be *hard for* $\mathbf{NL}$ if every member of $\mathbf{NL}$ is log-space reducible to B, and B is *complete for* **NL** if it is both hard for **NL** and a member of **NL**. It is not hard to see that '$\leq_{\log}$' is reflexive and transitive. Thus if B is known to be **NL**-complete and if B $\leq_{\log}$ A $\in$ **NL**, then A is **NL**-complete as well.

It is not hard to show that **NL** $\subseteq$ **P** $\subseteq$ **NP**. It is generally believed, although still unproved, that each of these inclusions is proper. It follows from this belief that a proof that a problem B is complete for one of these classes is strong evidence that B does not belong to any of the preceding classes in this list of inclusions.

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

Let $\mathbf{A} = \langle A, F\rangle$ be an algebra of cardinality n. Suppose that $q = |F|$ and the maximum rank of any member of F is r. Then, as an input instance to either Simp or SI, the size of $\mathbf{A}$ is at least $\max(n^r, nq)$. Similarly, let s denote the size of a typical instance, $\langle \mathbf{A}, \theta, a, b\rangle$, of Gen-Con. This is bounded below by $\max(n^r, |\theta|, nq)$. We can certainly conclude that

$$\log s \geq \max(r \log n,\, \log q). \tag{1}$$

2. Membership in NL

In order to prove that Gen-Con lies in **NL**, we need a slight variation on the classical theorem, due to Maltsev [18], describing the congruence on an algebra $\mathbf{A}$ generated by a set of pairs. The only difference between our formulation and that found in most texts is that we replace the monoid of all unary polynomial operations on $\mathbf{A}$ with a smaller and more manageable subset that we now describe. The proofs of Lemma 2.1 and Theorem 2.2 are identical to those of Theorems 4.18 and 4.19 in [19]. A treatment very similar to ours can be found in Section 2.1.2 of [24].

Let A be a set and f an n-ary operation on A, for some $n \geq 1$. We define

$$f_{(A)} = \bigl\{ f(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n) : 1 \leq i \leq n,\ a_1, \ldots, a_n \in A \bigr\}.$$

Thus, $f_{(A)}$ is the set of all unary operations on A obtained by substituting elements of A for all but one of the variables in f. The members of $f_{(A)}$ are called *elementary translations.* We write $C_{(A)}$ for the set of unary constant operations on A. Finally, if F is any set of operations on A, we let $F_{(A)} = C_{(A)} \cup \bigcup \{\, f_{(A)} : f \in F \,\}$.

Lemma 2.1. *Let A be a set and let F be a set of operations on A. Then* $\operatorname{Con}\langle A, F\rangle = \operatorname{Con}\langle A, F_{(A)}\rangle$.

Algorithm 1 GEN-CON$(\mathbf{A}, \theta, a, b)$

```
 (1)  $z \leftarrow a, \quad n \leftarrow |A|$
 (2)  for $i = 0$ to $n - 1$ do
 (3)      Choose $z' \in A$
 (4)      Choose $(u, v) \in \theta$
 (5)      if $\{u, v\} = \{z, z'\}$ then goto 11
 (6)      for $j = 1$ to $n^2 - 1$ do
 (7)          Choose $g \in F_{(A)}$
 (8)          $u \leftarrow g(u), \quad v \leftarrow g(v)$
 (9)          if $\{u, v\} = \{z, z'\}$ then goto 11
             od
(10)      Reject
(11)      if $z' = b$ then Accept
(12)      $z \leftarrow z'$
         od
(13)  Reject
```

For a set S of unary operations on A, let S^* denote the submonoid of the monoid of all self-maps of A generated by S. In particular, the identity map is an element of S^*.

Theorem 2.2. *Let $\mathbf{A} = \langle A, F\rangle$ be a finite algebra, $\theta \subseteq A \times A$ and $a, b \in A$. Then $(a, b) \in \mathrm{Cg}^{\mathbf{A}}(\theta)$ if and only if there are elements $z_0, z_1, \dots, z_m \in A$, pairs $(c_0, d_0), \dots, (c_{m-1}, d_{m-1}) \in \theta$ and operations $f_0, \dots, f_{m-1} \in (F_{(A)})^*$ such that*

$$\begin{gathered} a = z_0, \quad b = z_m, \quad \textit{and} \\ \{z_i, z_{i+1}\} = \{f_i(c_i), f_i(d_i)\} \quad \textit{for } i = 0, 1, \dots, m-1. \end{gathered} \tag{2}$$

Notice that in the above theorem, we can assume that $m < |A|$. For if not, then there are indices $j < k$ such that $z_j = z_k$. In that case, the sequence $z_0, z_1, \dots, z_j, z_{k+1}, \dots, z_m$ (along with the associated sequence of (c_i, d_i) and f_i) serve as witnesses to $(a, b) \in \mathrm{Cg}^{\mathbf{A}}(\theta)$.

It is a simple matter to turn the characterization in Theorem 2.2 into a procedure for computing GEN-CON.

Theorem 2.3. GEN-CON $\in$ **NL**.

Proof. Consider the nondeterministic algorithm labeled Algorithm 1. Essentially this procedure takes a guess at the sequences $z_0, z_1, \dots, z_n$; $(c_0, d_0), \dots, (c_{n-1}, d_{n-1})$; and $f_0, \dots, f_{n-1}$ in Theorem 2.2. If it finds such sequences, the algorithm accepts the input $\langle \mathbf{A}, \theta, a, b\rangle$. In each trip through the main loop (starting at statement 2), z contains the value of z_i. We nondeterministically choose values z' to be z_{i+1} and u, v to be c_i, d_i. In steps 5–9, we choose an operation $f \in F^*_{(A)}$ and test whether $\{z, z'\} = \{f(c_i), f(d_i)\}$. If this equality holds, we set (at step 12) z_{i+1} to be the value of z' and continue. If the equality fails, we reject the instance.

The computation of the operation f is accomplished by nondeterministically choosing a series of operations $g \in F_{(A)}$ whose composite is to be f. We don't keep track of all of these g's. Rather, we follow the images of c_i and d_i under these maps by recording them in the variables u and v. The length of the composition needed to obtain f can be bounded by n^2, since that is how many pairs (u, v) are possible. (And there is no need to encounter a pair more than once.)

It is also not necessary to construct the entire set $F_{(A)}$ in line 7. The set $F = \{g_0, \dots, g_{q-1}\}$ is part of the input. For $0 \leq i < n$, let $g_{q+i}(x)$ be the constant operation with value i. To choose g, pick integers k and ℓ with $0 \leq k < q + n$ and $1 \leq \ell \leq \operatorname{rank}(g_k)$, and members $a_1, \dots, a_r$ of A. The data $\langle k, \ell, a_1, \dots, a_r\rangle$ is sufficient to determine the operation $g(x) = g_k(a_1, \dots, a_{\ell-1}, x, a_{\ell+1}, \dots)$.

The total auxiliary memory required by Algorithm 1 is the space for storing the variables: $z, z', i, j, n, u, v, k, \ell, a_1, \dots, a_r$. Each of these holds an integer in the range $[0, n)$, hence requires only $\log n$ bits of storage, except for k and ℓ which require $\log(q+n)$ and $\log r$ bits respectively. Thus the total space requirement is on the order of $(r+7)\log n+\log(q+n)+\log r \in O(\log s)$, where s is the size of the instance, by the inequality (1). □

Theorem 2.3 can also be obtained from Immerman's theorem [12]. Immerman showed that every language definable in FO(TC) (first-order logic with a transitive closure operator) lies in **NL**. It follows from Theorem 2.2 that Gen-Con can be so defined. Similar remarks apply to the following theorem.

Theorem 2.4. *Both* Simp *and* SI *lie in* **NL**.

Proof. Observe that a nontrivial algebra $\mathbf{A}$ is simple if and only if

$$(\forall a \neq b)(\forall c \neq d)\ (a, b) \in \mathrm{Cg}^{\mathbf{A}}(c, d). \tag{3}$$

For each a, b, c, d, the truth of $(a, b) \in \mathrm{Cg}^{\mathbf{A}}(c, d)$ can be determined with a single call to Gen-Con. The computation required to verify formula (3) can be accomplished with four nested loops. It is important to observe that the space required for the call to Gen-Con can be *reused* on each trip through the loop. Thus, in addition to the space required by one call to Gen-Con, we only need to allocate space for the four loop counters, which run from 0 to $|A| - 1$. Thus Simp $\in$ **NL**.

Similarly, $\mathbf{A}$ is subdirectly irreducible if and only if

$$(\exists a \neq b)(\forall c \neq d)\ (a, b) \in \mathrm{Cg}^{\mathbf{A}}(c, d). \tag{4}$$

Using an argument similar to that used for simplicity, we see that SI $\in$ **NL**. □

We conclude this section with a discussion of a problem first considered in Bělohlávek and Chajda [1]. Let us define

$$\textsc{Cong-Class} = \{\, \langle \mathbf{A}, C\rangle : \mathbf{A} \text{ an algebra and } C \text{ a congruence class of some congruence on } \mathbf{A} \,\}.$$

If ψ is a congruence of an algebra $\mathbf{A}$, then a *congruence class of* ψ is a set of the form $a/\psi = \{\, x \in A : (a, x) \in \psi \,\}$ for some fixed element a of A.

Bělohlávek and Chajda show that when restricted to those algebras that generate a congruence-regular variety, the problem Cong-Class lies in **P**. However, using the techniques we have developed in this section, we are able to show that not only can the congruence-regularity assumption be dropped, but Cong-Class actually lies in **NL**, a (presumably proper) subclass of **P**.

Theorem 2.5. Cong-Class $\in$ **NL**.

Proof. Let $\mathbf{A}$ be an algebra, and $C \subseteq A$. Since the empty set is never a congruence class, we assume that C is nonempty. Define $\psi = \mathrm{Cg}^{\mathbf{A}}(C^2)$. It is easy to see that C is a class of some congruence if and only if C is a class of the congruence ψ. Fix an element $c \in C$. By the definition of ψ, we clearly have $C \subseteq c/\psi$, thus we need only check the reverse inclusion. In other words, we wish to check the condition

$$(\forall x \in A) \ \ \langle \mathbf{A},\, C^2, x, c\rangle \in \textsc{Gen-Con} \implies x \in C.$$

This condition can be checked with a simple loop. Strictly speaking, we can not call Gen-Con as a subroutine, since that would require enough space to hold the structure $\langle \mathbf{A}, C^2, x, c\rangle$. Instead, the code from Algorithm 1 must be inserted directly into the loop with references to θ replaced by C. Thus Cong-Class lies in **NL**. □

Unlike our primary problems, Gen-Con, SI and Simp, we have been unable to determine whether Cong-Class is complete for **NL**. We leave that as an open problem.

Problem. Is Cong-Class complete for **NL**?

3. NL-Hardness of the problems

We now turn to the problem of determining a lower bound for each of these problems. Specifically, we wish to show that each of the three problems discussed in Theorems 2.3 and 2.4 is **NL**-hard. For this we will use some facts from the complexity theory of finite graphs.

A *directed graph* (digraph) is a structure $\langle G, \varepsilon\rangle$, in which G is a nonempty, finite set (the vertices) and $\varepsilon \subseteq G \times G$ (the edges).

Let $\mathbf{G} = \langle G, \varepsilon\rangle$ be a digraph and $a, b \in G$. A *path* from a to b of length n is a sequence of vertices $a = v_0, v_1, \ldots, v_n = b$ such that for every $0 \le i < n$, $(v_i, v_{i+1}) \in \varepsilon$. For every vertex a, we agree that there is a path from a to a (of length 0). We define

$$R(a) = \{\, b \in G : \text{there is a path from } a \text{ to } b \,\}.$$

One of the best-known problems in complexity theory is the Graph Accessibility Problem:

$$\mathrm{GAP} = \big\{ \langle \mathbf{G}, a, b\rangle : \mathbf{G} \text{ a digraph, } a, b \in G \text{ and } b \in R(a) \big\}.$$

In other words, GAP is the problem of determining whether there is a path from a to b in a given digraph. This problem was shown to be complete for **NL** in [13], although the result is also implicit in [21]. It is used as the motivating problem for nondeterministic log-space in [20], where it is called REACHABILITY.

The digraph $\mathbf{G}$ is called *strongly connected* if for every $a \in G$, $R(a) = G$. In other words, for every a and b, there is a directed path from a to b. The vertex b will be called an *attractor* if, for every vertex a, $b \in R(a)$. Associated with these notions, we introduce two more problems.

$$\begin{aligned} \textsc{Str-Con} &= \{\, \mathbf{G} : \mathbf{G} \text{ is strongly connected} \,\} \\ \textsc{Attract} &= \{\, \mathbf{G} : \mathbf{G} \text{ has an attractor} \,\}. \end{aligned}$$

Str-Con was proved to be **NL**-complete by Laaser, see [13]. As far as we know, the problem Attract is new.

Theorem 3.1. *Each of the problems* GAP*,* Str-Con *and* Attract *is complete for* **NL***.*

Proof. We mentioned above that both GAP and Str-Con are complete for **NL**. Let $\mathbf{G}$ be a digraph. Observe that

$$\begin{aligned} \mathbf{G} \in \textsc{Str-Con} &\iff (\forall b)(\forall a)\ \langle \mathbf{G}, a, b\rangle \in \mathrm{GAP} \\ \mathbf{G} \in \textsc{Attract} &\iff (\exists b)(\forall a)\ \langle \mathbf{G}, a, b\rangle \in \mathrm{GAP}. \end{aligned} \tag{5}$$

In a manner similar to that used for SI in the proof of Theorem 2.4, an algorithm for Attract (and also for Str-Con) can be based on two nested loops, with a call to GAP inside the innermost loop. The space used for the GAP computation can be reused. Thus Attract lies in **NL**.

To show that Attract is **NL**-hard, we shall give a log-space reduction of GAP to Attract. Let $\langle \mathbf{G}, a, b\rangle$ be an instance of GAP, where $\mathbf{G} = \langle G, \varepsilon\rangle$. Let $H = G \cup \{c\}$ (where $c \notin G$) and $\beta = \varepsilon \cup \{\, (v, a) : v \in G \,\} \cup \{(b, c)\}$. Let $\mathbf{H} = \langle H, \beta\rangle$. We claim that $\langle \mathbf{G}, a, b\rangle \in \mathrm{GAP}$ if and only if $\mathbf{H} \in \textsc{Attract}$, with c as the attractor.

To see this, suppose first that there is a path $\mathbf{p}$ from a to b in $\mathbf{G}$. Then for any vertex v of G, the sequence $v, \mathbf{p}, c$ is a path in $\mathbf{H}$ from v to c. Thus c is an attractor. Conversely, if c is an attractor in $\mathbf{H}$, then there is a path (in $\mathbf{H}$) from a to c. But such a path must include b, and (since there is no exit from c) only the last vertex in the path is equal to c. Thus, there is a path in $\mathbf{G}$ from a to b, so that $\langle \mathbf{G}, a, b\rangle \in \mathrm{GAP}$.

This reduction is clearly computable in log-space, since the only auxiliary storage that is needed is for several counters. Thus Attract is **NL**-complete. □

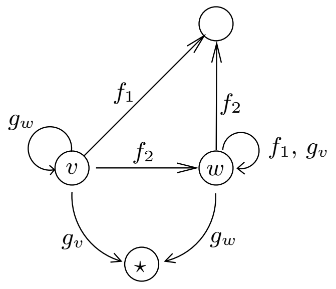

FIGURE 1. Part of an algebra $A(\mathbf{G})$

The reader has surely noticed the structural similarity between the conditions in (5) and those in equivalences (3) and (4):

$$\begin{aligned} \mathbf{A} \in \text{SIMP} &\iff (\forall a \neq b)(\forall c \neq d)\ (a,b) \in \mathrm{Cg}^{\mathbf{A}}(c,d) \\ \mathbf{G} \in \text{STR-CON} &\iff (\forall b) \qquad (\forall a) \qquad \langle \mathbf{G}, a, b\rangle \in \text{GAP} \end{aligned}$$

$$\begin{aligned} \mathbf{A} \in \text{SI} &\iff (\exists a \neq b)(\forall c \neq d)\ (a,b) \in \mathrm{Cg}^{\mathbf{A}}(c,d) \\ \mathbf{G} \in \text{ATTRACT} &\iff (\exists b) \qquad (\forall a) \qquad \langle \mathbf{G}, a, b\rangle \in \text{GAP}. \end{aligned}$$

We shall now exhibit reductions between the graph problems of Theorem 3.1 and the algebra problems discussed in Theorem 2.4. For this we use the following construction.

Let $\mathbf{G} = \langle G, \varepsilon\rangle$ be a digraph. Fix an element $\star \notin G$ and let $G^\star = G \cup \{\star\}$. Define a new graph $\mathbf{G}^\star = \langle G^\star, \varepsilon\rangle$. (Thus $\star$ is an isolated point of $\mathbf{G}^\star$.) For $v \in G^\star$ define $N[v] = \{v\} \cup \{\, w : (v,w) \in \varepsilon \,\}$ (the closed neighborhood of v), and let $k = \max_{v\in G}\big|N[v]\big|$. For each $1 \le i \le k$, choose a function $f_i \colon G^\star \to G^\star$ in such a way that for all $v \in G^\star$, $\{\, f_i(v) : 1 \le i \le k \,\} = N[v]$. In other words, for each edge from v to w there should be some i with $f_i(v) = w$. Note that for all i we have $f_i(\star) = \star$. Also, for each $v \in G$ we define the operation g_v on $G^\star$ by

$$g_v(w) = \begin{cases} \star & \text{if } w = v; \\ w & \text{otherwise.} \end{cases}$$

Finally, we define an algebra

$$A(\mathbf{G}) = \big\langle G^\star,\ \langle f_i\rangle_{1\le i\le k},\ \langle g_v\rangle_{v\in G}\big\rangle.$$

The construction of $A(\mathbf{G})$ is illustrated schematically in Figure 1.

Let us make two observations about the algebra $A(\mathbf{G})$. First, for any element a of G, the subuniverse generated by a is $R(a) \cup \{\star\}$. Second, $A(\mathbf{G})$ is a unary algebra, that is, each of its basic operations is of rank 1. A useful fact about unary algebras is the following lemma. The proof is an easy verification.

Lemma 3.2. *Let $\mathbf{B}$ be a unary algebra and S a subuniverse of $\mathbf{B}$. Then the binary relation $\psi_S = \{\, (x,y) : x,y \in S \text{ or } x = y \,\}$ is a congruence on $\mathbf{B}$.*

Note that the congruence ψ_S has exactly one nontrivial congruence class, namely S itself. For the next two lemmas, we omit the superscript '$A(\mathbf{G})$' in the notation $\mathrm{Cg}(x, y)$.

Lemma 3.3. (1) *Let a and b be vertices of $\mathbf{G}$. Then $(b, \star) \in \mathrm{Cg}(a, \star)$ if and only if $b \in R(a)$.*

(2) *If c and d are distinct elements of $G^\star$, then $\mathrm{Cg}(c, \star) \subseteq \mathrm{Cg}(c, d)$.*

Proof. Let $S = R(a) \cup \{\star\}$ be the subalgebra of $A(\mathbf{G})$ generated by a. Since $a, \star \in S$, it follows from Lemma 3.2 that $\mathrm{Cg}(a, \star) \subseteq \psi_S$. Thus, if $b \equiv \star \pmod{\mathrm{Cg}(a, \star)}$ then, since $b \neq \star$, we get $b \in S$, in fact, $b \in R(a)$. Conversely, if $b \in R(a)$, then there is a sequence of indices $i_1, i_2, \dots, i_m$ such that $b = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_m}(a)$. Since $\star$ is fixed by each f_i, we obtain $(b, \star) \in \mathrm{Cg}(a, \star)$.

For the second claim, if $d = \star$ then the inclusion is trivial. So suppose $d \neq \star$. Since $c \neq d \neq \star$, working modulo $\mathrm{Cg}(c, d)$ we have $c = g_d(c) \equiv g_d(d) = \star$. Since $\mathrm{Cg}(c, \star)$ is the smallest congruence identifying c with $\star$, we get $\mathrm{Cg}(c, \star) \subseteq \mathrm{Cg}(c, d)$. □

The relationship between the algebraic problems and the graph problems is given in the following lemma.

Lemma 3.4. *For any digraph $\mathbf{G}$ and $a, b \in G$ we have*

$$\begin{aligned}
\langle \mathbf{G}, a, b\rangle \in \mathrm{GAP} &\iff \big\langle A(\mathbf{G}), \{(a, \star)\}, b, \star\big\rangle \in \textsc{Gen-Con};\\
\mathbf{G} \in \textsc{Str-Con} &\iff A(\mathbf{G}) \in \textsc{Simp};\\
\mathbf{G} \in \textsc{Attract} &\iff A(\mathbf{G}) \in \mathrm{SI}.
\end{aligned}$$

Proof. The first equivalence follows immediately from Lemma 3.3(1). Suppose that $\mathbf{G}$ is strongly connected. To show $A(\mathbf{G})$ simple, pick a pair c, d of distinct elements from $G^\star$. We wish to show that $\mathrm{Cg}(c, d)$ is the universal congruence. Without loss of generality, assume that $c \neq \star$. By Lemma 3.3(2), we have $(c, \star) \in \mathrm{Cg}(c, d)$. By assumption $R(c) = G$, so by Lemma 3.3(1), the congruence class of c modulo $\mathrm{Cg}(c, \star)$ contains all of $G^\star$. Thus $\mathrm{Cg}(c, \star)$, hence also $\mathrm{Cg}(c, d)$ is universal.

Conversely, suppose that $A(\mathbf{G})$ is simple. Pick vertices a, b in G. Since $\mathrm{Cg}(a, \star)$ is the universal congruence, we apply Lemma 3.3(1) again to obtain $b \in R(a)$.

Now we address the third equivalence. Suppose that b is an attractor of $\mathbf{G}$. We wish to show that $\mathrm{Cg}(b, \star)$ is the smallest nontrivial congruence (the monolith) of $A(\mathbf{G})$. Choose any pair c, d of distinct elements. Assume that $c \neq \star$. By assumption, $b \in R(c)$, so again using Lemma 3.3, $\mathrm{Cg}(b, \star) \subseteq \mathrm{Cg}(c, \star) \subseteq \mathrm{Cg}(c, d)$.

For the converse, suppose that $\mathrm{Cg}(c, d)$ is the monolith of $A(\mathbf{G})$, with $\star \neq c \neq d$. By Lemma 3.3, $\mathrm{Cg}(c, \star) \subseteq \mathrm{Cg}(c, d)$. Since $c \neq \star$, $\mathrm{Cg}(c, \star)$ is not the identity congruence, hence by the minimality of $\mathrm{Cg}(c, d)$, we get $\mathrm{Cg}(c, \star) = \mathrm{Cg}(c, d)$. But then, for any $a \in G$, $\mathrm{Cg}(c, \star) \subseteq \mathrm{Cg}(a, \star)$, hence by Lemma 3.3(1), $c \in R(a)$. In other words, c is an attractor of $\mathbf{G}$. □

Finally, we can combine Lemma 3.4 and Theorem 3.1 to obtain our main theorem.

Theorem 3.5. *Each of the problems* Gen-Con, Simp *and* SI *is complete for* **NL**.

Remarks:

(1) From Lemma 3.4 we see that Gen-Con remains complete for **NL** if we restrict to instances $\langle \mathbf{A}, \theta, a, b\rangle$ in which $\mathbf{A}$ is a unary algebra and $|\theta| = 1$.

(2) It is natural to wonder about the complexity of recognizing congruences on an algebra. In other words, given an algebra $\mathbf{A}$ and a binary relation θ on A, determine whether θ is a congruence on $\mathbf{A}$. It is not hard to see that this can be done in (deterministic) log-space.

First, one can verify that θ is an equivalence relation using three nested loops, each running through the elements of A. For example, if a and b are two of the loop counters, then we can test the symmetry of θ by verifying that whenever (a, b) is in θ, so is (b, a).

To test the second condition of Definition 1.1, use two sets of variables $a_1, \dots, a_r$ and $b_1, \dots, b_r$. (Here r denotes the maximum rank of any of the basic operations.) For each basic operation f, have each of $(a_1, \dots, a_r)$ and $(b_1, \dots, b_r)$ traverse the entire set A^r. Whenever we have $(a_i, b_i) \in \theta$ for all $i \leq k$, verify that $\big(f(a_1, \dots, a_k), f(b_1, \dots, b_k)\big) \in \theta$. This requires $2r$ counters, each using $\log n$ bits. Note that an input instance to this problem is almost identical to that of Gen-Con, so we conclude from inequality (1) that our space requirements are bounded by the logarithm of the size of the input.

(3) In the construction of $A(\mathbf{G})$, the sequence $\langle g_v\rangle_{v\in G}$ of unary operations can be replaced with a single binary operation given by

$$x \cdot y = \begin{cases} \star & \text{if } x = y \\ y & \text{otherwise.} \end{cases}$$

This does not result in any space-saving when all operations are given via tables, but might be very efficient if the operations are allowed to be presented by other means, such as Boolean circuits.

(4) GAP is a problem for directed graphs. There is an analogous problem, called UGAP, for undirected graphs. It follows at once that UGAP $\in$ **NL**. However, it is an open question whether UGAP is complete for **NL**. The complexity class **SL** (symmetric log-space) is defined in such a way that UGAP is complete for **SL**. See Lewis and Papadimitriou[16] for details. The completeness of UGAP for **NL** is equivalent to the assertion that **SL** = **NL**.

A set A can be viewed as an algebra in which the set of basic operations is empty. In that case, for any subset θ of A^2, $\mathrm{Cg}^A(\theta)$ is nothing but the smallest equivalence relation on A containing θ.

Now it is easy to see that $(a,b) \in \mathrm{Cg}^A(\theta)$ if and only if a and b lie in the same connected component of the undirected graph $\langle A, \bar{\theta}\rangle$, where $\bar{\theta} = \theta \cup \{\, (y,x) : (x,y) \in \theta \,\}$. In other words, UGAP coincides with the special case of Gen-Con in which the "algebra" is constrained to have no basic operations. In our experience, this special case is of lesser complexity than is the general case. This suggests that one ought to try to prove that Gen-Con $\notin$ **SL**, thereby settling the question of whether **SL** and **NL** are distinct.

Recall the problem Cong-Class mentioned at the end of Section 2. We proved in Theorem 2.5 that Cong-Class lies in **NL**. Since we have been unable to prove that this problem is complete for **NL**, we are led to wonder whether Cong-Class might lie in an interesting proper subclass. **SL** seems to be a natural candidate. As a companion to Problem 2, we ask

Does Cong-Class lie in **SL**?

4. Fully invariant congruences

An *endomorphism* of an algebra $\mathbf{A} = \langle A, F\rangle$ is a homomorphism from $\mathbf{A}$ to itself, in other words, a function $h\colon A \to A$ such that for all $f \in F$ and $a_1, \dots, a_m \in A$, $h\big(f(a_1,\dots,a_m)\big) = f\big(h(a_1),\dots,h(a_m)\big)$. The collection of all endomorphisms of $\mathbf{A}$ is denoted $\mathrm{End}(\mathbf{A})$.

A congruence ψ on $\mathbf{A}$ is called *fully invariant* if for all $(x,y) \in \psi$ and all $h \in \mathrm{End}(\mathbf{A})$, $\big(h(x), h(y)\big) \in \psi$. We denote by $\mathrm{Con}_{\mathrm{fi}}(\mathbf{A})$ the set of fully invariant congruences of $\mathbf{A}$. It is immediate from the definition that

$$\mathrm{Con}_{\mathrm{fi}}(\langle A, F\rangle) = \mathrm{Con}(\langle A,\, F \cup \mathrm{End}(\mathbf{A})\rangle). \tag{6}$$

This equation has several consequences. First, both δ_A and A^2 are fully invariant congruences on $\mathbf{A}$. Second, for any $\theta \subseteq A^2$, there is a smallest fully invariant congruence on $\mathbf{A}$ containing θ. We shall write $\mathrm{Cg}^{\mathbf{A}}_{\mathrm{fi}}(\theta)$ for this congruence. Finally, Theorem 2.2 can be applied to compute $\mathrm{Cg}^{\mathbf{A}}_{\mathrm{fi}}(\theta)$ (with F replaced by $F \cup \mathrm{End}(\mathbf{A})$).

Parallel to our problem Gen-Con, we define

$$\text{Gen-Con}_{\text{FI}} = \big\{\, \langle \mathbf{A}, \theta, a, b\rangle : a, b \in A,\ \theta \subseteq A^2 \text{ and } (a,b) \in \mathrm{Cg}^{\mathbf{A}}_{\mathrm{fi}}(\theta) \big\}.$$

With minor modifications, Algorithm 1 can be used to compute Gen-Con$_{\text{FI}}$. In light of equation (6), if Algorithm 1 is used to compute Gen-Con$_{\text{FI}}$, then in step 7, g must be chosen from $(F \cup \mathrm{End}(\mathbf{A}))_{(A)}$ rather than from $F_{(A)}$. But note that $(F \cup \mathrm{End}(\mathbf{A}))_{(A)} = F_{(A)} \cup \mathrm{End}(\mathbf{A})$. Thus, we provide a modified algorithm, Algorithm 2, in which this step is replaced with the sequence 7a–7e. The idea behind this sequence of steps is as follows. We first toss a coin. If the coin comes up 'heads', we choose $g \in F_{(A)}$ as before. However, on 'tails', we guess an arbitrary function $g\colon A \to A$ and then check to see if g is an endomorphism of $\mathbf{A}$. If it is, we proceed to step 8. If not, we reject this instance.

Algorithm 2 $\textsc{Gen-Con}_{\textsc{fi}}(\mathbf{A}, \theta, a, b)$

```
(1)  $z \leftarrow a, \quad n \leftarrow |A|$
(2)  for $i = 0$ to $n-1$ do
(3)     Choose $z' \in A$
(4)     Choose $(u, v) \in \theta$
(5)     if $\{u, v\} = \{z, z'\}$ then goto 11
(6)     for $j = 1$ to $n^2 - 1$ do
 7a.         Toss a coin
 7b.         If heads then choose $g \in F_{(A)}$
 7c.            else do
 7d.               Choose $g\colon A \to A$
 7e.               If $g \notin \mathrm{End}(\mathbf{A})$ then Reject
                   od
(8)          $u \leftarrow g(u), \quad v \leftarrow g(v)$
(9)          if $\{u, v\} = \{z, z'\}$ then goto 11
            od
(10)    Reject
(11)    if $z' = b$ then Accept
(12)    $z \leftarrow z'$
        od
(13) Reject
```

Unlike the original algorithm, this modified version can not be executed in log-space. This is because we require enough space to hold the entire function g whenever the coin comes up 'tails'. Since a function from A to A is a list of n integers in the range $\{0 \ldots n-1\}$, the space requirement for g is $n \log n$. In general, this will not be bounded by the logarithm of the size of the input (see inequality (1)).

However, our modified algorithm does run in (nondeterministic) polynomial time. The verification that a function g is an endomorphism requires one pass through each of the tables for the basic operations of the algebra. Since the algorithm reaches step 7 at most n^3 times, the total running time will be bounded by a polynomial in the size of the input.

As an alternative, one can prove that $\textsc{Gen-Con}_{\textsc{fi}} \in \mathbf{NP}$ by observing that in light of Theorem 2.2, $\textsc{Gen-Con}_{\textsc{fi}}$ can be defined by a second-order, existential sentence. From Fagin's theorem [6] it follows that any language defined in this way lies in **NP**.

We now wish to prove that $\textsc{Gen-Con}_{\textsc{fi}}$ is hard for **NP**. We will do this by reducing the well-known problem CLIQUE to $\textsc{Gen-Con}_{\textsc{fi}}$. For a positive integer n, let $\mathbf{K}_n$ denote the digraph with vertex set $\{1, 2, \ldots, n\}$ and (directed) edges $\{\,(x, y) : x \neq y\,\}$. If $\mathbf{G}$ is a digraph, then a *clique* of $\mathbf{G}$ is a subgraph isomorphic to some $\mathbf{K}_n$. We call $\mathbf{G}$ *loopless* if it has no edges of the form (x,x). We define

$$\textsc{Clique} = \big\{(\mathbf{G},n) : \mathbf{G} \text{ a loopless digraph, } n \geq 1, \\ \text{and } \mathbf{G} \text{ has a clique of size } n\big\}.$$

The problem Clique is known to be **NP**-complete, see [8, p. 194].

Let $\mathbf{G} = \langle G, \varepsilon\rangle$ and $\mathbf{H} = \langle H, \tau\rangle$ be digraphs. A *homomorphism from* $\mathbf{H}$ *to* $\mathbf{G}$ is a function $t\colon H \to G$ such that $(x,y) \in \tau$ implies $(t(x), t(y)) \in \varepsilon$. Note that a loopless graph $\mathbf{G}$ has a clique of size n if and only if there is a homomorphism from $\mathbf{K}_n$ to $\mathbf{G}$.

In [10], Hedrlín and Pultr described an elegant transformation from digraphs to unary algebras that has been used several times [2, 15] to reduce problems involving graphs to similar problems involving algebraic structures. Given a digraph $\mathbf{G} = \langle G, \varepsilon\rangle$, we shall define an algebra $\widehat{\mathbf{G}}$ as follows. The universe of $\widehat{\mathbf{G}}$ is the set $\widehat{G} = G \cup \varepsilon \cup \{u, v\}$ where u and v are points not appearing in either G or ε. $\widehat{\mathbf{G}} = \langle \widehat{G}, f_0, f_1\rangle$ where f_0 and f_1 are unary operations defined by

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

Furthermore, let $t\colon \mathbf{H} \to \mathbf{G}$ be a digraph homomorphism. We define a function $\hat{t}\colon \widehat{H} \to \widehat{G}$ given by

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

Theorem 4.1 (Hedrlín and Pultr, [10])**.** *The mappings $\mathbf{G} \mapsto \widehat{\mathbf{G}}$ and $t \mapsto \hat{t}$ 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 homomorphism t, the function $\hat{t}\colon \widehat{\mathbf{H}} \to \widehat{\mathbf{G}}$ is a homomorphism and furthermore, the mapping $t \mapsto \hat{t}$ is a bijection between the homomorphisms from $\mathbf{H}$ to $\mathbf{G}$ and the homomorphisms between $\widehat{\mathbf{H}}$ and $\widehat{\mathbf{G}}$.*

It follows that any homomorphism from $\widehat{\mathbf{H}}$ to $\widehat{\mathbf{G}}$ must preserve u and v, and map vertices to vertices and edges to edges.

Lemma 4.2. Clique $\leq_{\log}$ Gen-Con$_{\text{FI}}$.

Proof. Let $\langle \mathbf{G}, n\rangle$ be an instance of Clique, with $\mathbf{G} = \langle G, \varepsilon\rangle$. Fix a new vertex a and define $\mathbf{G}' = \big\langle G \cup \{a\},\, \varepsilon \cup (\{a\} \times G) \cup (G \times \{a\})\big\rangle$. That is, there is an edge from a to each vertex of $\mathbf{G}$ as well as an edge in the opposite direction. Let $\mathbf{K} = \mathbf{K}_{n+1}$ and let $\mathbf{G}' + \mathbf{K}$ denote the disjoint union of the graphs $\mathbf{G}'$ and $\mathbf{K}$.

Now define $\mathbf{G}''$ to be $\mathbf{G}'+\mathbf{K}$ and set $\mathbf{A} = \widehat{\mathbf{G}''}$ (see Theorem 4.1). Pick two distinct vertices $\bar{a}$ and $\bar{b}$ from $\mathbf{K}$, and let $\bar{e}$ be the edge from $\bar{a}$ to $\bar{b}$. Finally, let $\theta = \{(\bar{a}, \bar{e})\}$ and $\psi = \mathrm{Cg}_{\mathrm{fi}}^{\mathbf{A}}(\theta)$. To complete the proof of the Lemma, we shall show that

$$\langle \mathbf{G}, n\rangle \in \textsc{Clique} \iff \langle \mathbf{A}, \theta, a, \bar{a}\rangle \in \textsc{Gen-Con}_{\textsc{fi}},$$

that is, $\mathbf{G}$ has a clique of size n if and only if $(a, \bar{a}) \in \psi$.

Suppose first that $\mathbf{G}$ has a clique of size n. Then $\mathbf{G}'$ has a clique of size $n+1$ that includes the vertex a. Therefore, there is a graph homomorphism t_0 from $\mathbf{K}$ to $\mathbf{G}'$. Because of the symmetry of $\mathbf{K}$, we can assume that $t_0(\bar{a}) = a$. The map t_0 can be extended to a graph homomorphism $t\colon \mathbf{G}'' \to \mathbf{G}''$ by mapping each vertex of G' to itself. Theorem 4.1 yields an (algebra) homomorphism $\hat{t}\colon \mathbf{A} \to \mathbf{A}$. Note that by the definition of $\hat{t}$, we have $\hat{t}(\bar{a}) = a$. Now, using the fact that ψ is a fully invariant congruence, we compute

$$(\bar{a}, \bar{e}) \in \psi \implies (f_0(\bar{a}), f_0(\bar{e})) = (u, \bar{a}) \in \psi \implies$$
$$(\hat{t}(u), \hat{t}(\bar{a})) = (u, a) \in \psi \implies (a, \bar{a}) \in \psi.$$

Conversely, suppose $(a, \bar{a}) \in \psi$. Since $\mathbf{A}$ is a unary algebra and $\widehat{\mathbf{K}}$ is a subalgebra, by Lemma 3.2 there is a congruence $\nu = \widehat{K}^2 \cup \delta_A$ on $\mathbf{A}$. Since $(a, \bar{a}) \notin \nu$, we certainly have $\psi \nsubseteq \nu$. On the other hand, $\theta \subseteq \nu$, so ν is not fully invariant. (For otherwise, $\psi = \mathrm{Cg}_{\mathrm{fi}}(\theta) \subseteq \nu$.) It follows that some endomorphism of $\mathbf{A}$ must fail to map $\widehat{K}$ to itself. By Theorem 4.1, this endomorphism is of the form $\hat{t}$, for some $t\colon \mathbf{G}'' \to \mathbf{G}''$, and it must be the case that t does not map K into itself. But since $\mathbf{K}$ is complete and is disjoint from $\mathbf{G}'$, t must actually map $\mathbf{K}$ to $\mathbf{G}'$. Therefore, $\mathbf{G}'$ contains a clique of size $n+1$. At most one of the vertices in the clique can be equal to a, so we conclude that $\mathbf{G}$ has an n-clique. $\square$

Theorem 4.3. $\textsc{Gen-Con}_{\textsc{fi}}$ *is* $\mathbf{NP}$*-complete*

Proof. Our modified version of Algorithm 1 shows that $\textsc{Gen-Con}_{\textsc{fi}} \in \mathbf{NP}$. Since $\textsc{Clique}$ is $\mathbf{NP}$-complete, it follows from Lemma 4.2 that $\textsc{Gen-Con}_{\textsc{fi}}$ is $\mathbf{NP}$-complete as well. $\square$

The notion of "full invariance" can be extended to objects other than congruences on algebras. For example, let $\mathbf{G} = \langle G, \varepsilon\rangle$ be a digraph and $S \subseteq G$. Let us call S fully invariant if for every $h \in \mathrm{End}(\mathbf{G})$, $h(S) \subseteq S$. Furthermore define the fully invariant subset generated by a set S (denoted $\mathrm{Sg}_{\mathrm{fi}}^{\mathbf{G}}(S)$) to be the smallest fully invariant subset of $\mathbf{G}$ containing S. Notice that $\mathrm{Sg}_{\mathrm{fi}}^{\mathbf{G}}(S) = \bigcup \{\, h(S) : h \in \mathrm{End}(\mathbf{G}) \,\}$. We define two problems:

$$\textsc{FI-Subset} = \{\, \langle \mathbf{G}, S\rangle : \mathbf{G} \text{ a digraph and } S \text{ a fully invariant subset} \,\}$$
$$\textsc{Gen-Subset}_{\textsc{fi}} = \left\{\, \langle \mathbf{G}, S, a\rangle : \mathbf{G} \text{ a digraph and } a \in \mathrm{Sg}_{\mathrm{fi}}^{\mathbf{G}}(S) \,\right\}.$$

Using the same ideas as in Theorem 4.3, we can prove the following.

Theorem 4.4. $\textsc{FI-Subset}$ *is complete for* $\mathbf{co\text{-}NP}$. $\textsc{Gen-Subset}_{\textsc{fi}}$ *is complete for* $\mathbf{NP}$.

Proof. Suppose that $\langle \mathbf{G}, S\rangle$ is an instance of FI-Subset. Let P denote the complement of FI-Subset. To show that S is *not* fully invariant, we can guess a function $h\colon G \to G$ and verify that h is an endomorphism and that $h(S) \nsubseteq S$. This gives a nondeterministic algorithm for P that runs in polynomial time. To reduce Clique to P, let $\mathbf{G}$ be a loopless graph and n a positive integer. Let $\mathbf{H} = \mathbf{G} + \mathbf{K}_n$. Then it is easy to see that K_n fails to be fully invariant in $\mathbf{H}$ if and only if there is a (graph) homomorphism from $\mathbf{K}_n$ to $\mathbf{G}$. This in turn is equivalent to the existence of an n-clique in $\mathbf{G}$. Thus P is **NP**-complete, and therefore FI-Subset is complete for **co-NP**.

Now for the second problem. The condition $\langle \mathbf{G}, S, a\rangle \in$ Gen-Subset$_{\text{FI}}$ can be checked by guessing a function $h\colon G \to G$, checking that h is an endomorphism of $\mathbf{G}$, and that $a \in h(S)$. To prove that Clique $\leq_{\log}$ Gen-Subset$_{\text{FI}}$ follow the construction given in Lemma 4.2 to produce the graph $\mathbf{G}''$. Then one easily sees that $\mathbf{G}$ has an n-clique if and only if $a \in \mathrm{Sg}_{\mathrm{fi}}^{\mathbf{G}''}(K)$. Thus Gen-Subset$_{\text{FI}}$ is **NP**-complete. □

References

[1] R. Bělohlávek and I. Chajda, *Congruence classes in regular varieties*, Acta Math. Univ. Comenian. **LXVIII** (1999), no. 1, 71–75.

[2] C. Bergman and G. Slutzki, *Complexity of some problems concerning varieties and quasivarieties of algebras*, SIAM J. Comput. **30** (2000), no. 2, 359–382.

[3] S. Burris and H. P. Sankappanavar, *A course in universal algebra*, Springer-Verlag, New York, 1981.

[4] J. Demel, M. Demlová, and V. Koubek, *Fast algorithms constructing minimal subalgebras, congruences, and ideals in a finite algebra*, Theoret. Comput. Sci. **36** (1985), 203–216.

[5] M. Demlová, J. Demel, and V. Koubek, *Simplicity of algebras requires to investigate almost all operations*, Comment. Math. Univ. Carolin. **23** (1982), no. 2, 325–335.

[6] R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, Complexity of Computation (Providence, R.I.) (R. Karp, ed.), Proceedings of the SIAM-AMS Symposium in Applied Math, vol. 7, American Mathematical Society, 1974, pp. 27–41.

[7] R. Freese, *Computing congruences efficiently*, Manuscript available at `http://www.math.hawaii.edu/~ralph/Preprints/cg.pdf`, March 1999.

[8] 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.

[9] G. Grätzer, *Universal algebra*, second ed., Springer-Verlag, New York, 1979.

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

[11] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, MA, 1979.

[12] N. Immerman, *Languages that capture complexity classes*, SIAM J. Comput. **16** (1987), no. 4, 760–778.

[13] N. D. Jones, *Space-bounded reducibility among combinatorial problems*, J. Comput. System Sci. **11** (1975), no. 1, 68–85.

[14] N. D. Jones and W. T. Laaser, *Complete problems for deterministic polynomial time*, Theoret. Comput. Sci. **3** (1977), 105–117.

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

[16] H. R. Lewis and C. H. Papadimitriou, *Symmetric space-bounded computation*, Theoret. Comput. Sci. **19** (1982), no. 2, 161–187.

[17] S. Mac Lane, *Categories for the working mathematician*, Graduate Texts in mathematics, vol. 5, Springer-Verlag, New York, 1971.

[18] A.I. Maltsev, *On the general theory of algebraic systems*, Mat. Sbornik **77** (1954), 3–20 (Russian).

[19] R. McKenzie, G. McNulty, and W. Taylor, *Algebras, lattices, varieties*, vol. I, Wadsworth & Brooks/Cole, Belmont, CA, 1987.

[20] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, MA, 1994.

[21] W. J. Savitch, *Relationships between nondeterministic and deterministic tape complexities*, J. Comput. System Sci. **4** (1970), no. 2, 177–192.

[22] M. Sipser, *Introduction to the theory of computation*, PWS Publishing Company, Boston, MA, 1997.

[23] J. Varlet, *Remarks on fully invariant congruences*, Contributions to universal algebra (Amsterdam), Colloq. Math. Soc. János Bolyai, vol. 17, North-Holland, 1977, pp. 515–554.

[24] W. Wechler, *Universal algebra for computer scientists*, Springer-Verlag, Berlin, 1992.

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

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