MATHEMATICAL SYSTEMS THEORY **10**, 1–17 (1976)

New Problems Complete for Nondeterministic Log Space

by

NEIL D. JONES, Y. EDMUND LIEN*

Department of Computer Science
University of Kansas
Lawrence, Kansas 66045

and

WILLIAM T. LAASER

Department of Computer Science
Stanford University
Palo Alto, California 94305

ABSTRACT

It is shown that a variety of problems have computational complexity equivalent to that of finding a path through a directed graph. These results, which parallel those of Karp at a lower complexity level, concern satisfiability of propositional formulas with two literals per clause, generation of elements by an associative binary operation, solution of linear equations with two variables per equation, equivalence of generalized sequential machines with final states, and deciding the $LL(k)$ and $LR(k)$ conditions for context-free grammars. Finally, several problems are shown equivalent to reachability in undirected graphs, including bipartiteness and satisfiability of formulas with the "exclusive or" connective.

Introduction. Let **L** and **P** denote the classes of all sets recognizable by deterministic Turing machines in log space and polynomial time, respectively; and let **NL** and **NP** denote the corresponding classes recognized by nondeterministic Turing machines. It is well known that $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP}$; but whether any of these containments is proper is still open. For example, the existence of a polynomial-time recognizable set whose recognition requires more than log space has not been established, although it seems likely that such sets exist. In [JL], several familiar problems in **P** were shown to require more than log space, if one assumes that any such set exists.

It is also known that either proper containment or equality at any point in $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP}$ has a number of very significant consequences with respect to the computational complexity of a wide variety of theoretically and practically important problems (see [S], [C], [K], [HH], [HS], [J], [JL]).

* The work of this author was partly supported by Kansas General Research Grant 3756–5038. A preliminary version appears in the Proceedings of the Eighth Princeton Conference on Information Sciences and Systems, Princeton University, pp. 184–188, March 1974.

The class **L** comprises a surprisingly large class of sets; the class of functions computable in log space is also very large, including, for example, all the functions used in [C] and [K] to reduce one problem to another. These two classes are closely related, since it is easily verified that a function $f(x)$ is computable in log space if and only if the set $\{\langle x, i, b\rangle | b \text{ is the } i^{\text{th}} \text{ symbol of } f(x)\}$ is in **L** and for all x, $f(x)$ is bounded in length by a polynomial in the length of x. For example, the numerical relations $x = y$, $x < y$ and $x \leq y$ are in **L**, where x and y are integers or rationals. The functions $x+y$, $x-y$, $\max(x,y)$ and $x \cdot y$ are computable in log space, although it is not known whether $x \div y$ is. These results can be extended to show that matrix multiplication and sorting a list of numbers may also be done in log space. It has also been shown by Lynch [L] that the set of true propositional formulas is in **L**.

We now turn to the relation between **L** and the other classes.

Definition 1. Let $L, M \subseteq \Sigma^*$ be languages where $|\Sigma| \geq 2$ and let $\mathscr{C}$ be a class of languages over Σ. Then:

I. L is *reducible* to M (written $L \leq M$) if and only if there is a function $f\colon \Sigma^* \to \Sigma^*$ such that
  (a) f is computable by a deterministic Turing machine in log space, and
  (b) for all $x \in \Sigma^*$, $x \in L$ if and only if $f(x) \in M$.

II. L is *$\mathscr{C}$-hard* if and only if $M \leq L$ for all $M \in \mathscr{C}$.

III. L is *$\mathscr{C}$-complete* or *complete for $\mathscr{C}$* if and only if $L \in \mathscr{C}$ and L is $\mathscr{C}$-hard.

Here we assume the details, terminology and definitions of [J]. The relation $\leq$ used above is the same as $\leq^{lms}_{\log}$ there, and **L** and **NL** are abbreviations for $\text{DSPACE}_{\log}$ and $\text{NSPACE}_{\log}$, respectively.

LEMMA 2. *Let L, M and N be languages over Σ with $|\Sigma| \geq 2$. Then:*

I. *If $L \leq M$ and $M \leq N$ then $L \leq N$.*

II. *If $L \leq M$ and M is in* **L**, **NL**, **P** *or* **NP**, *then L is in the same class.*

III. *If $L \leq M$ and L is $\mathscr{C}$-hard, then M is $\mathscr{C}$-hard.*

IV. *If L is complete for* **NL**, *then* $\mathbf{L} \subsetneq \mathbf{NL}$ *if and only if* $L \in \mathbf{NL} - \mathbf{L}$; *and similarly for the inclusions* $\mathbf{NL} \subseteq \mathbf{P}$ *and* $\mathbf{P} \subseteq \mathbf{NP}$.

Proofs are straightforward (e.g., [J]). Thus a set complete for a class $\mathscr{C}$ serves as a single representative of the entire class, with respect to the open containment problems.

Note on representation of problems. In this paper we shall not specify in detail just how objects such as graphs, grammars, etc., are represented by strings over Σ^*. The justification for this comes from Lemma 2: as long as there are log-space computable functions which translate one representation into another, any two representations are equivalent with respect to membership in or completeness for **L**, **NL**, **P** or **NP**. For example, representations of graphs via incidence matrices, adjacency lists, or sets of ordered pairs are all equivalent in this sense. A computational problem, whose complexity is to be studied, will be specified by giving the input information provided to the computation (under the heading INPUT) and the property that the input information must possess (under the heading PROPERTY). A Turing machine Z solves the problem if, whenever given information described under INPUT, Z has at least one accepting computation if and only if the input satisfies the specified property. A similar convention was used in [K].

In [J] and in a paper by Savitch [S1] several problems concerning graphs and finite automata were shown to be **NL**-complete. In particular, we mention the following important problem.

Let $\Gamma = \langle N, A \rangle$ denote a directed graph with node set $N = \{1, 2, \dots, n\}$ for some $n \geq 1$ and arc set $A \subseteq N \times N$. The following was proven in [J]; it is also implicit in [S], where the term "threadable maze" is used.

THEOREM 3. *The graph accessibility problem, GAP, defined below, is complete for* **NL**.

INPUT: a directed graph $\Gamma = \langle \{1, 2, \dots, n\}, A \rangle$
PROPERTY: Γ has a path from node 1 to node n.

We now summarize the results of this paper. First, the following problems (which are precisely defined in later sections) are shown to be complete for **NL**.

Propositional Logic. To determine whether a formula containing at most two literals per clause is unsatisfiable; or whether it may be shown unsatisfiable by means of unit resolution.

Algebra. To determine whether an element of a finite set is generated from a subset by a given associative binary operation.

Linear Systems and Graphs. To determine whether a directed graph is bilateral; or strongly connected. To determine whether there is no 0-1 solution to $K \cdot x \geq b$ where b is a vector and K is a matrix with each row containing at most two nonzero entries. To determine whether there is a positive solution y to $y \cdot K = 0$, where each row of K contains at most two distinct nonzero entries in $\{1, -1\}$.

Automata. To determine whether two deterministic generalized sequential machines with final states are inequivalent.

Context-Free Grammars. The membership problem for linear grammars; determining whether a *cfg* is not a precedence grammar, or whether a given nonterminal may ever appear in a sentential form. The problems of determining whether a *cfg* generates an infinite set, or is not $LL(k)$, or not $LR(k)$ for fixed k, are shown to be complete for **NL** if restricted to grammars without ε-production or useless nonterminals.

Finally, several problems are shown equivalent, although it is not known whether they are either in **L** or complete for **NL**. These include the accessibility problem for undirected graphs, determining whether a graph is nonbipartite, and some restricted covering problems.

Problems in Propositional Logic

We now fill two gaps in the complexity of various elementary problems in propositional logic. Some terminology is introduced first.

A propositional variable or its negation is called a *literal*. A *clause* is a finite set of literals, representing a disjunction. A clause containing just one literal is called a *unit clause.* By *unit resolution* we refer to an inference rule [CL] which allows one to deduce clause $\mathscr{F}$ from clauses $(\mathscr{F} \vee \sim C)$ and C, where C is a unit clause.

Let SAT denote the satisfiability problem defined below.

INPUT: a propositional formula $\mathscr{F}$ in conjunctive normal form with at most three variables per clause
PROPERTY: $\mathscr{F}$ is satisfiable.

UNSAT and UNIT denote the similar problems with "$\mathscr{F}$ is unsatisfiable" or "$\mathscr{F}$ is provably unsatisfiable by unit resolution" replacing the PROPERTY statement respectively.

SAT was shown complete for **NP** by Cook and Karp in [C] and [K], and UNIT was shown complete for **P** in [JL] (that proof actually held for at most four literals per clause, but is easily strengthened to three).

Now let SAT_2, UNSAT_2 and UNIT_2 denote the corresponding problems for formulas containing at most two literals per clause.

THEOREM 4. *$UNSAT_2$ and $UNIT_2$ are complete for* **NL**.

Proof. We first show that UNIT_2 and UNSAT_2 are nondeterministically recognizable in log space. Each clause $(y \vee z)$ in formula $\mathscr{F}$ will be considered as the pair of implications $(\sim y \to z)$ and $(\sim z \to y)$.

Assertion. $\mathscr{F}$ is in UNIT_2 (UNSAT_2) if and only if there is a sequence of literals $x_1, x_2, \ldots, x_k$ such that I and II hold (I holds, and either II or III also holds).

I. $(x_i \to x_{i+1})$ is in $\mathscr{F}$ for $i = 1, \ldots, k-1$
II. x_1 is a clause of $\mathscr{F}$ and $x_k = \sim x_1$
III. $x_1 = \sim x_j = x_k$ for some j with $1 < j < k$.

Proof of Assertion. The "if" part is obvious. The "only if" part for UNIT_2 is also immediate, since any deduction of the empty clause may be placed in linear form due to the two-literal-per-clause limitation. For the "only if" part for UNSAT_2, suppose no sequence satisfying (II) or (III) exists. Then the following procedure will satisfy $\mathscr{F}$.

```
Step 1. Let $U = (x|x$ is a unit clause of $\mathscr{F}\} \cup \{\sim x_1|$ there are $x_1, x_2, \ldots, x_n$
        such that $x_n = \sim x_1$, and $(x_i \to x_{i+1})$ is in $\mathscr{F}$ for $i = 1, 2, \ldots, n\}$
Step 2. while some variable has not been assigned a truth value
        do begin
Step 3.     Let $V = U \cup \{x_n|$ there is a sequence $x_1, \ldots, x_n$ such that $x_1 \in U$ and
            $(x_i \to x_{i+1})$ is in $\mathscr{F}$ for $i = 1, 2, \ldots, n\}$
Step 4.     Assign "TRUE" to every literal in $V$
Step 5.     Let $U = V \cup \{u\}$ where $u$ is the first variable which has not been
            assigned a truth value.
            end
```

Clearly this procedure will eventually terminate since there are but a finite number of variables. Furthermore, U and V can never contain both a literal x and its negation since otherwise x would satisfy II if they appeared in Step 1 or it would satisfy III if they appeared in Step 3. Therefore the final set V represents a model which satisfies $\mathscr{F}$. Hence the Assertion is proved.

Therefore, to test if $\mathscr{F}$ is unsatisfiable, we may nondeterministically guess a sequence of literals which satisfies either II or III. Furthermore, to do this, we need only store the index of the initial literal and the index of the literal presently being considered. Therefore UNSAT_2 is nondeterministically recognizable in log space. Similarly, UNIT_2 is also in **NL**.

Now it suffices to show that $\text{GAP} \leq \text{UNIT}_2$ and $\text{GAP} \leq \text{UNSAT}_2$. Let Γ be a graph on nodes $1, 2, \ldots, n$. Construct the formula

$$\mathscr{F}_\Gamma = 1 \wedge \sim n \wedge \left(\bigwedge_{\substack{\langle i,j\rangle \text{ is an} \\ \text{arc of } \Gamma}} (\sim i \vee j) \right)$$

Intuitively we may give $\mathscr{F}_\Gamma$ the following interpretation. Each literal i stands for the statement "Node i is reachable." Thus $\mathscr{F}_\Gamma$ states that node 1 is reachable (i.e., node 1 is where we are starting), node n is not reachable, and for each pair of nodes i and j, if i is reachable and $\langle i, j\rangle$ is an arc, then node j is reachable. Therefore, there is a path from node 1 to node n (i.e., node n is reachable) if and only if $\mathscr{F}_\Gamma$ is unsatisfiable. Note also that $\mathscr{F}$ is unsatisfiable if and only if $\mathscr{F}_\Gamma$ is unsatisfiable by unit resolution.

Remarks. These results seem to depend in a significant way on the logical operations and formats chosen. For example, UNSAT for formulas in disjunctive normal form is essentially trivial. More interesting, if "$\vee$" is interpreted as "exclusive or", a new version of UNSAT_2 is obtained which appears to be complete for a subclass of **NL**. This is discussed further in the last section of the paper.

A Problem in Algebra

Let X be a set and o be a binary operation defined on X. The set W *generated* by a subset S of X is the smallest subset of X which contains S and is closed under o. An element $w \in X$ is said to be generated by S if and only if $w \in W$. We shall study the following problem in algebra.

Associative Generation Problem (AGEN)
INPUT: a finite set X, an associative binary operation o defined on X, a subset S of X and an element $w \in X$
PROPERTY: w is generated by S.

When the restriction that the binary operation be associative is removed, the resulting problem, called GEN in [JL], proves to be **P**-complete. The next theorem, showing that AGEN is **NL**-complete, is an interesting contrast to that result.

THEOREM 5. *AGEN is complete for* **NL**.

Proof. First we prove that AGEN is in **NL**. Let S be a subset of X and o be an associative operation on X. It is clear that an element w is generated by S if and only if $w = x_1 o x_2 o \ldots o x_k$ for some elements $x_1, x_2, \ldots, x_k$ in S. Note that the parenthesization of the x_i's is irrelevant. Let ${}_1x_i$ denote the element $x_1 o x_2 o \ldots o x_i$; hence, ${}_1x_k = w$. We observe that if ${}_1x_1, {}_1x_2, \ldots, {}_1x_k$ is the shortest sequence which yields w, then every element in the sequence must be distinct, otherwise a shorter sequence which yields w can be obtained. Consequently, w is generated by S if and only if $w = x_1 o x_2 o \ldots o x_k$ for some elements $x_1, x_2, \ldots, x_k$ in S and $k \leq |X|$.

Therefore, to test if w is generated by S, we may nondeterministically select a sequence of at most $|X|$ elements from S. We need only store the index of w and the index of current ${}_1x_i$. When an element is selected from S, ${}_1x_i$ is updated to ${}_1x_{i+1}$ and is compared to w. This concludes the proof that AGEN is in **NL**.

Now it suffices to show that GAP $\leq$ AGEN. Let $\Gamma = \langle N, A\rangle$ be a directed graph with nodes $N = \{1, 2, \ldots, n\}$. We reduce GAP to AGEN as follows.

Let

$$X = N \cup N \times N \cup \{\#\} \qquad \text{where } \# \text{ is not in } N \text{ or } N \times N,$$
$$S = A \cup \{1\}, \qquad w = n$$

and let o be defined by

$$\begin{array}{lll} \text{I.} & \forall x \in X, & x o \# = \# o x = \# \\ \text{II.} & \forall x, y \in N, & x o y = \# \\ \text{III.} & \forall x, y, z \in N, & \langle x, y\rangle o z = \# \\ & & x o \langle y, z\rangle = z \quad \text{if } x = y \\ & & \text{and } x o \langle y, z\rangle = \# \quad \text{if } x \neq y \\ \text{IV.} & \forall x, y, u, v \in N, & \langle x, y\rangle o \langle u, v\rangle = \langle x, v\rangle \quad \text{if } y = u \\ & & \text{and } \langle x, y\rangle o \langle u, v\rangle = \# \quad \text{if } y \neq u \end{array}$$

The operation o is easily shown to be associative. Furthermore, if there is a path $1, x_1, x_2, \ldots, x_i, n$ from 1 to n, we must have $1 o \langle 1, x_1\rangle o \langle x_1, x_2\rangle o \ldots o \langle x_i, n\rangle = n = w$. On the other hand, if w is generated by S, a directed path from 1 to n can also be established. Therefore, GAP $\leq$ AGEN. □

Problems in Linear Systems and Graphs

In [K], the 0–1 Integer Programming problem was shown to be **NP**-complete. It is also known that even if restricted to at most three variables in each inequality, the problem is still **NP**-complete. In this section, we consider two special cases of solvability of linear equations and inequalities, and show that 0–1 Integer Programming with at most two variables per inequality is **NL**-complete.

We first show that some primitive relations or functions can be decided or computed in deterministic log space.

LEMMA 6. *Let x, y, u and v be integers represented in binary form.*

I. *The relations $x = y$, $x < y$ and $x \leq y$ are in* **L**.

II. *The functions $x + y$, $x \dot{-} y$ ($x \dot{-} y = 0$ if $x < y$ and $x - y$ if $x \geq y$) and $x \cdot y$ are computable in deterministic log space.*

III. *The relations $x/y = u/v$, $x/y < u/v$ and $x/y \leq u/v$ are in* **L**.

Proof. The relations $x = y$, $x < y$ and $x \leq y$, and the functions $x + y$ and $x - y$ are easily decided or computed by familiar symbol-by-symbol algorithms, which can clearly be done in log space. For x ·y, let $x = a_m \ldots a_1 a_0$, $y = b_n \ldots b_1 b_0$ and $x \cdot y = c_p \ldots c_1 c_0$. Then

$$c_k = (a_k \cdot b_0 + a_{k-1} \cdot b_1 + \ldots + a_0 \cdot b_k + carry) \bmod 2$$

The portion before *carry* may clearly be calculated in log k space; and *carry*, which is bounded by k, can be obtained as a result of calculating c_{k-1}. Thus c_0, $c_1, \ldots, c_k$ can be calculated in sequence as required. Finally, III follows directly from I and II. □

Let K represent a matrix over rationals, and x, y and b represent vectors over rationals. It is assumed that K must satisfy one or both of the following conditions:

Condition I. Each row of K has at most two nonzero entries

Condition II. Each nonzero entry of K is in $\{1, -1\}$ and no row has two identical nonzero entries.

THEOREM 7. *The problem IP_2 defined below is* **NL**-*complete.*

INPUT: a matrix K satisfying condition I, and a vector b

PROPERTY: there exists no 0–1 vector x such that $K \cdot x \geq b$.

Proof. The reduction from SAT to 0–1 Integer Programming in [K] can be used to show that $\text{UNSAT}_2 \leq \text{IP}_2$. It suffices to show that $\text{IP}_2 \leq \text{UNSAT}_2$. Let $K \cdot x \geq b$ be a system of m linear inequalities over n variables. Under Condition I, the i^{th} inequality must be of the following forms

$$K_{ij}\, x_j \geq b_i \qquad \ldots \text{Case I}$$

or

$$K_{ij}\, x_j + K_{ik}\, x_k \geq b_i \qquad \ldots \text{Case II}$$

Let $\{z_1, z_2, \ldots, z_n\}$ be proposition variables. For the i^{th} inequality, we construct formula C_i as follows.

Case I. In solving $K_{ij}\, x_j \geq b_i$ for a 0–1 solution, we must have one of the four cases:

(a) $K_{ij} \geq b_i > 0$. In this case, x_j must be 1. Let C_i be z_j.

(b) $b_i > K_{ij} > 0$ or $b_i > 0 > K_{ij}$. Let C_i be FALSE, for there is no solution to the inequality.

(c) $0 \geq b_i > K_{ij}$. Let C_i be $\sim z_j$, since x_j must be 0.

(d) All other cases. Let C_i be TRUE. In these cases, x_j can be either 1 or 0.

It should be clear that C_i is satisfiable if and only if the i^{th} inequality has a 0–1 solution.

Case II. Suppose $K_{ij}\, x_j + K_{ik}\, x_k \geq b_i$ is the i^{th} inequality. We omit the details of the proof in this case. It is basically similar to the proof in Case I. We need to sort five numbers $K_{ij}+K_{ik}, K_{ij}, K_{ik}, b_i$ and 0. Then define C_i according to the ordering. Some possible forms of C_i are $\sim z_j \vee z_k$, $\sim z_j \vee z_k$, $z_j \vee z_k$, etc. Every possible form of C_i can be expressed in conjunctive normal form of at most two clauses and at most two literals per clause. Again, C_i is satisfiable if and only if the i^{th} inequality has a 0–1 solution.

Finally, we have a formula $\mathscr{F} = \wedge_{i=1}^{m} C_i$ which is satisfiable if and only if $K \cdot x \geq b$ has a $0-1$ solution. To conclude the proof, we need to note that the functions mentioned in the reduction process above, such as $K_{ij}+K_{ik}$ and sorting five numbers, can be done in deterministic log space. This is shown in Lemma 6. □

The next problem to be considered is the existence of a positive solution to a system of homogeneous linear equations satisfying both conditions I and II. Our result is obtained by establishing a relation between homogeneous linear equations and bilateral directed graphs.

Let $\Gamma = \langle N, A \rangle$ be a directed graph with nodes $N = \{1, 2, \ldots, n\}$ and arcs $A = \{a_1, a_2, \ldots, a_m\}$. Define an $m \times n$ matrix K where

$$K_{ij} = \begin{cases} 1 & \text{if arc } a_i \text{ is directed from node } j \\ -1 & \text{if arc } a_i \text{ is directed to node } j \\ 0 & \text{otherwise} \end{cases}$$

The matrix K, called the *incidence matrix* of Γ, satisfies both conditions I and II.

We define a relation R on N where any two nodes i and j are said to be $i\,R\,j$ if and only if i is reachable from j and vice versa. It is easy to see that R is an equivalence relation. The set N can be partitioned into equivalence classes $D_1, D_2, \ldots, D_r$. A graph Γ is *strongly connected* if and only if $r = 1$. We say that $D_i \leq D_j$, for any i and j, if $D_i = D_j$ or some node in D_i is reachable from some node in D_j. A graph Γ is *bilateral* if and only if there are not two distinct equivalence classes D_i and D_j such that $D_i \leq D_j$. Note that every equivalence class in a graph Γ corresponds to a strongly connected subgraph of Γ.

LEMMA 8. *Let $\Gamma = \langle N, A \rangle$ be a directed graph with incidence matrix K. Γ is bilateral if and only if $y \cdot K = 0$ has a positive solution (every component is positive).*

Proof. We first prove the "only if" part. Suppose Γ is strongly connected. A directed circuit σ passing through every arc at least once can be found. Let y_j be the number of times that σ passes through arc a_j. It is easily seen that, for each node i,

$$\sum_{K_{ji}=-1} y_j = \sum_{K_{ki}=1} y_k$$

This is because the left-hand side indicates the number of times that σ enters node i, which should be equal to the right-hand side, the number of times that σ leaves i. A simple induction on the number of equivalence classes can be used to obtain the similar result for bilateral graphs. Thus $y \cdot K = 0$.

For the "if" part, let y be a positive vector such that $y \cdot K = 0$. For the graph Γ, there must be a maximal element D in the partially ordered set $\{D_1, D_2, \ldots, D_r\}$ of equivalence classes on relation R, i.e., $D \leq D'$ implies $D = D'$. Assume that Γ is not bilateral; then we can find a maximal element D such that there is an arc connecting a node i in D to a node i' in some other equivalence class D'. Without loss of generality, we assume that $D = \{1, 2, \ldots, p\}$. Let K_j denote the column vector of K corresponding to node j. Since D is maximal, we have

$$\sum_{j=1}^{p} K_j \geq 0$$

But the arc $\langle i, i' \rangle$ makes

$$\sum_{j=1}^{p} K_j \neq 0.$$

Therefore

$$y \cdot \sum_{j=1}^{p} K_j = \sum_{j=1}^{p} y \cdot K_j \neq 0.$$

But $y \cdot K_j = 0$ because $y \cdot K = 0$. A contradiction has been established. Therefore, Γ must be bilateral. □

Consider the following decision problems, BILATERAL (or STRONG)

INPUT: a finite directed graph Γ
PROPERTY: Γ is bilateral (or Γ is strongly connected).

It is shown in [J] that STRONG is **NL**-complete. The proof was established by showing that GAP $\leq$ STRONG and that STRONG is in **NL**. The same reduction also establishes that GAP $\leq$ BILATERAL. To determine if a graph is bilateral, we need only check for every arc $\langle i, j \rangle$ if i is reachable from j. Therefore, we have the following result.

LEMMA 9. *BILATERAL is complete for* **NL**.

By Lemmas 8 and 9, we can establish the following result in linear equations.

THEOREM 10. *The PS problem defined below is* **NL***-complete.*
INPUT: a matrix K satisfying conditions I and II
PROPERTY: there is a positive vector y such that $y \cdot K = 0$.

Proof. Lemma 8 essentially asserts that BILATERAL $\leq$ PS. Therefore, PS is at least **NL**-hard. It now suffices to show that PS $\leq$ BILATERAL. Suppose K is a matrix satisfying conditions I and II. Without loss of generality, we will exclude rows with all 0's. Define a column vector b such that

$$b_i = \begin{cases} 1 & \text{if } i^{th} \text{ row of } K \text{ has no } 1 \\ -1 & \text{if } i^{th} \text{ row of } K \text{ has no } -1 \\ 0 & \text{otherwise} \end{cases}$$

Append vector b to K and call the resulting matrix K'. Matrix K' must correspond to a directed graph Γ, since every row of K' has exactly two nonzero entries, one 1 and one -1. Note that any positive solution to $y \cdot K' = 0$ is a positive solution to $y \cdot K = 0$. Further, if $y \cdot K = 0$ has a positive solution y_0, then $y_0 \cdot b = -y_0 \cdot (-b) = -y_0 \cdot \Sigma K_j = -\Sigma y_0 \cdot K_j = 0$ where ΣK_j is the componentwise sum of all column vectors of K. Hence $y \cdot K' = 0$ (and consequently $y \cdot K = 0$) has a positive solution if and only if Γ is bilateral. □

A Problem Concerning Automata

It has been seen in [J] and [S] that many natural problems concerning finite automata (e.g., emptiness and infiniteness for deterministic or nondeterministic machines) are **NL**-complete. These results were obtained by exploiting the similarity of graphs and paths to transition diagrams and accepting state sequences. The following result applies to a more complex device and involves a significantly less straightforward construction. The device is a natural model for studying equivalence of programs, which need only yield identical outputs in case their inputs are well-formed.

Definition 11. *A deterministic generalized sequential machine with final states* (*dgsmf*) is a tuple $M = \langle K, \Sigma, \Delta, \delta, \lambda, q_0, F\rangle$ where K, Σ and Δ are alphabets (of *state*, *input* and *output* symbols, respectively), $\delta: K \times \Sigma \to K$ and $\lambda: K \times \Sigma \to \Delta^*$ are maps (the *next state* and *output* functions), $q_0 \in K$ *and* $F \subseteq K$ (the *initial state* and the set of *final states*).

The domains of δ and λ can be extended to $K \times \Sigma^*$ in a natural way. The *set accepted by M* is $L(M) = \{x | \delta(q_0, x) \in F\}$, and the partial *function* M: $\Sigma^* \to \Delta^*$ *computed by M* is defined by

$$M(x) = \begin{cases} \lambda(q_0, x) & \text{if } x \in L(M) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Two *dgsmf*'s are said to be *equivalent* if and only if they accept the same sets and compute the same functions. We shall assume that all *dgsmf*'s have common input and output alphabets in the following.

THEOREM 12. *The following problem, INEQV, is* **NL**-*complete.*

INPUT: two *dgsmf*'s M^1 and M^2
PROPERTY: M^1 and M^2 are inequivalent.

Proof. INEQV is **NL**-hard because GAP $\leq$ INEQV is easily established by the methods of [J]. To see that INEQV is in **NL**, we need an auxiliary result. Let $M^i = (K^i, \Sigma, \Delta, \delta^i, \lambda^i, q_0^i, F^i)$ for $i = 1, 2$ be the *dgsmf*'s.

Assertion. M^1 is equivalent to M^2 if and only if $M^1(x) = M^2(x)$ for all $x \in \Sigma^*$ such that $|x| \leq 2 \cdot |K^1| \cdot |K^2|$.

Proof of Assertion. "Only if" is trivial; so suppose for the sake of argument that $M^1(x) = M^2(x)$ for all x with $|x| \leq 2 \cdot |K^1| \cdot |K^2|$ but M^1 and M^2 are not equivalent. Then there must be a shortest string $z = a_1 a_2 \ldots a_m$ in Σ^* for which $M^1(z) \neq M^2(z)$. Now the sequence of state pairs (q_0^1, q_0^2), $(\delta^1(q_0^1, a_1), \delta^2(q_0^2, a_1))$, $\ldots, (\delta^1(q_0^1, a_1 a_2 \ldots a_m), \delta^2(q_0^2, a_1 a_2 \ldots a_m))$ must contain some state pair (p^1, p^2)

at least three times, since $m > 2 \cdot |K^1| \cdot |K^2|$. Thus we may decompose z into $z = z_0 z_1 z_2 z_3 (z_1 \neq \varepsilon$ and $z_2 \neq \varepsilon)$ so z_0, $z_0 z_1$ and $z_0 z_1 z_2$ yield the same state pair (p^1, p^2). Now let $y_0^1 = \lambda^1(q_0^1, z_0)$, $y_i^1 = \lambda^1(p^1, z_i)$ for $i = 1, 2, 3$ and define y_0^2, y_1^2, y_2^2 and y_3^2 similarly. By the fact that z is shortest, we obtain equations

$$y_0^1 y_3^1 = y_0^2 y_3^2$$
$$y_0^1 y_1^1 y_3^1 = y_0^2 y_1^2 y_3^2$$

and

$$y_0^1 y_2^1 y_3^1 = y_0^2 y_2^2 y_3^2$$

By simple string manipulation these imply $M^1(z) = y_0^1 y_1^1 y_2^1 y_3^1 = y_0^2 y_1^2 y_2^2 y_3^2 = M^2(z)$, a contradiction. Hence the proof of the assertion is complete.

Returning to the proof of our theorem, clearly M^1 and M^2 are inequivalent if and only if there is a string $x \in \Sigma^*$ such that either

I. one of $\delta^1(q_0^1, x)$ and $\delta^2(q_0^2, x)$ is final and the other not; or
II. both are final, and m^{th} symbol of $M^1(x)$ differs from m^{th} symbol of $M^2(x)$ for some $m \leq \max(|M^1(x)|, |M^2(x)|)$.

By the result of the Assertion we may choose $m \leq 2 \cdot w \cdot |K^1| \cdot |K^2|$ where w is the length of the longest output produced in any one transition of M^1 or M^2. The following nondeterministic procedure will find such an x if one exists.

```
Step 1. Read $M^1$ and $M^2$;
Step 2. Calculate $2 \cdot w \cdot |K^1| \cdot |K^2|$;
Step 3. Choose a random $m$ with $0 \leq m \leq 2 \cdot w \cdot |K^1| \cdot |K^2|$;
Step 4. Set $\ell^1 \leftarrow \ell^2 \leftarrow 0$; $p^1 \leftarrow q_0^1$; $p^2 \leftarrow q_0^2$;
Step 5. for $i \leftarrow 0$ step 1 until $2 \cdot |K^1| \cdot |K^2|$ do
        begin
            if $p^1$ and $p^2$ are not both final or both nonfinal, then accept $\langle M^1, M^2 \rangle$
            as inequivalent and stop;
            choose $a \in \Sigma$ at random;

            $\alpha^1 \leftarrow \lambda^1(p^1, a)$; $\alpha^2 \leftarrow \lambda^2(p^2, a)$; $p^1 \leftarrow \delta^1(p^1, a)$; $p^2 \leftarrow \delta^2(p^2, a)$;

            Let $\alpha^1 = c_1 c_2 \ldots c_r$ $(c_i \in \Delta)$;
            if $\ell^1 + 1 \leq m \leq \ell^1 + r$ then $b^1 \leftarrow c_{m-\ell^1}$;
            $\ell^1 \leftarrow \ell 1 + r$;

            Let $\alpha^2 = d_1 d_2 \ldots d_s$ $(d_i \in \Delta)$;
            if $\ell^2 + 1 \leq m \leq \ell^2 + s$ then $b^2 \leftarrow d_{m-\ell^2}$;
            $\ell^2 \leftarrow \ell^2 + s$;

        end
Step 6. if $b^1 \neq b^2$ then accept $\langle M^1, M^2 \rangle$ as inequivalent
                      else reject $\langle M^1, M^2 \rangle$.
```

It should be clear that this algorithm works as specified. Only a fixed number of memory cells are used, each of which is either a binary number bounded by a polynomial in the size of M^1 and M^2, or can be represented by a pointer to M^1 or M^2. Thus it operates in log space. □

This result is also of interest since it illustrates a natural problem in which the addition of nondeterminism converts the complexity from rather simple (in **NL**) to infinitely complex (undecidable, see Griffiths [G]).

Problems Concerning Contect-Free Grammars

We now show that several parsing-related problems are **NL**-complete. The term context-free grammar will be abbreviated to *cfg*. The notation a^k denotes a string of a's with length equal to k, and Σ^{*k} denotes the set which contains all strings in Σ^* of length at most k. A leftmost derivation of string β from a string α will be denoted $\alpha \overset{*}{\underset{L}{\Rightarrow}} \beta$. We assume that readers are familiar with the concepts of linear *cfg*, $LR(k)$ grammars and simple precedence grammars [AU].

THEOREM 13. *Each of the following problems is* **NL**-*complete.*

LINMEMBER:

INPUT: a linear *cfg* $G = (N, \Sigma, P, S)$ and a string $x \in \Sigma^*$
PROPERTY: $x \in L(G)$.

ACCESSIBLE:

INPUT: a *cfg* $G = (N, \Sigma, P, S)$ and a nonterminal $B \in N$
PROPERTY: $S \overset{*}{\underset{L}{\Rightarrow}} \alpha B \beta$ for some $\alpha, \beta \in (N \cup \Sigma)^*$.

NONPRECEDENCE:

INPUT: a *cfg* $G = (N, \Sigma, P, S)$
PROPERTY: G is not a simple precedence grammar.

FIRST$_k$: (for each value $k = 1, 2, \ldots$)

INPUT: an ε-free *cfg* $G = (N, \Sigma, P, S)$, and strings $\alpha \in (N \cup \Sigma)^*$, $x \in \Sigma^*$
PROPERTY: $\alpha \overset{*}{\underset{L}{\Rightarrow}} x$ and $|x| < k$, or $|x| = k$ and $\alpha \overset{*}{\underset{L}{\Rightarrow}} x\beta$ for some $\beta \in (N \cup \Sigma)^*$.

Proof. We first show that each of these is **NL**-hard by reducing GAP to it. Let $\Gamma = \langle N, A \rangle$ with $N = \{1, \ldots, n\}$, and consider the linear *cfg* $G_1 = (\{A_1, \ldots, A_n\}, \{1\}, P, A_1)$, where P contains productions $A_n \to 1$, $A_n \to 1A_n$, and $A_i \to A_j$ for every arc (i, j) of Γ. Clearly Γ has a path from 1 to n if and only if $1 \in L(G_1)$, if and only if A_n is accessible from S, if and only if $S \overset{*}{\underset{L}{\Rightarrow}} 1^k$. For the remaining case, construct $G_2 = (\{A_0, A_1, \ldots, A_n\}, \{a_1, \ldots, a_n\}, P, A_0)$ where P contains the productions

$$\begin{aligned} &A_0 \to a_1A_0 \mid a_2A_0 \mid \cdots \mid a_nA_0 \mid a_n \mid a_nA_1, \\ &A_n \to a_n, \text{ and} \\ &A_i \to a_iA_j \text{ for each arc } (i, j) \text{ of } \Gamma. \end{aligned}$$

Clearly $a_n \lessdot a_n$; further $a_n \gtrdot a_n$ holds if and only if there is a path from 1 to n. No other conflicts are possible, so G_2 violates the precedence condition if and only if Γ is in GAP.

The question "$x \in L(G)$?" for a linear *cfg* G may be decided by a standard top-down parsing procedure without backtrack, nondeterministically choosing a goal production at each step. This procedure needs to store only a single nonterminal and two pointers to positions within x; thus the first problem is clearly in **NL**. Note: I. H. Sudborough [Su] has exhibited a single linear grammar G, whose membership problem is also complete for **NL**.

To see that ACCESSIBLE is in **NL**, consider the graph $\Gamma = (N, A)$, where (C, D) is an arc in A if and only if P contains a production of the form $C \to \gamma D \delta$, with $\gamma, \delta \in (N \cup \Sigma)^*$. Clearly B is accessible from S if and only if Γ has a path from node S to node B. Therefore ACCESSIBLE $\leq$ GAP and so ACCESSIBLE is in **NL** by Lemma 2.

The precedence relations $\lessdot$, $\doteq$ and $\gtrdot$ may clearly be recognized in **NL**; thus the existence of multiple relations between adjacent symbols is determinable in nondeterministic log space. The other conditions (no ε-productions or productions with identical right sides) are clearly so determinable, so NONPRECEDENCE is in **NL**.

The FIRST$_k$ property may be determined by constructing a graph with node set $\{\alpha\} \cup (N \cup \Sigma)^{*k}$, with an arc from σ to τ if and only if $|\tau| \leq k$ and either $\sigma \Rightarrow \tau$ with $|\tau| < k$, or $\sigma \Rightarrow \tau\theta$ with $|\tau| = k$ for some $\theta \in (N \cup \Sigma)^*$. Clearly the property is satisfied if and only if a path exists from α to x, due to the fact that G has no ε-productions. Thus FIRST$_k$ $\leq$ GAP. □

Define a nonterminal to be *productive* if it generates at least one terminal string. We now show that violations of the $LL(k)$ and $LR(k)$ conditions may be detected within **NL**, on the assumption that the grammar concerned has no ε-productions and only productive nonterminals. However it appears likely that the productivity condition itself is not **NL**-testable, since it is complete for **P** (immediate from the fact that context-free emptiness is complete for **P**). Hence we formulate the following definition.

Definition 14. A set L is **NL**-*complete with respect to a set D* if

I. any set in **NL** is reducible to $L \cap D$, and
II. there is a set L' in **NL** such that $L' \supseteq L \cap D$ and $L' \cap (D - L) = \varnothing$.

Condition II above states the existence of a nondeterministic log space Turing machine which correctly determines whether its input is in L, providing the input is already known to be in D.

THEOREM 15. *The following problems are* **NL**-*complete with respect to* $D = \{cfg\ G \mid G$ *has no* ε-*productions or nonproductive nonterminals*$\}$

CFINFINITE:

INPUT: a *cfg* G without ε-productions or nonproductive nonterminals
PROPERTY: $L(G)$ is an infinite set of strings.

NONLL$_k$: (for each $k \geq 1$)

INPUT: a *cfg* G without ε-productions or nonproductive nonterminals
PROPERTY: G is not an $LL(k)$ grammar.

NONLR$_k$: (for each $k \geq 0$)

INPUT: a *cfg* G without ε-productions or nonproductive nonterminals
PROPERTY: G is not an $LR(k)$ grammar.

Proof. To show these **NL**-hard, let $\Gamma = (\{1, 2, \ldots, n\}, A)$ be a directed graph. First, consider *cfg* G_1 of the previous theorem, augmented by adding productions $A_i \rightarrow 1$ for $i = 1, \ldots, n$. Clearly Γ has a path from 1 to n if and only if the modified G_1 generates an infinite set, so the CFINFINITE problem is **NL**-hard.

Now let $G = (\{A_1, \ldots, A_n\}, \{a_1, \ldots, a_n, b_1, \ldots, b_n\}, P, A_1)$ where P contains productions $A_i \rightarrow a_j A_j$ if and only if (i, j) is an arc of Γ, $A_i \rightarrow b_i$ for $i = 1, \ldots, n$, and $\mathrm{A}_n \rightarrow \mathrm{A}_n \mathrm{A}_n$. If a path from 1 to n exists, G is ambiguous and so not $LL(k)$ or $LR(k)$ for any k; and if no such path exists, G is easily seen to be $LL(1)$ and $LR(0)$. Thus the remaining problems are **NL**-hard.

Now let G be a *cfg* without ε-productions or nonproductive non-terminals. Clearly $L(G)$ is infinite if and only if there is a derivation $A \overset{*}{\Rightarrow} \alpha A \beta$ for some nonterminal A and α, $\beta \in (N \cup \Sigma)^*$. This can certainly be recognized nondeterministically in log space.

Now let G be a *cfg* without ε-productions or nonproductive nonterminals, which we are to test for violation of the $LL(k)$ or $LR(k)$ conditions. By Theorem 5.2 of [AU], G is not $LL(k)$ if and only if there are distinct productions $A \rightarrow \beta$, $A \rightarrow \gamma$ in P, and strings w, x, $y \in \Sigma^*$, $\alpha \in (N \cup \Sigma)^*$ such that $x \neq y$, $S \underset{\ell m}{\overset{*}{\Rightarrow}} wA\alpha$, $x \in$ FIRST$_k$ $(\beta\alpha)$, and $y \in$ FIRST$_k$ $(\gamma\alpha)$. Construct a graph Γ whose nodes are strings in $N(N \cup \Sigma)^{*k}$. Define (σ, τ) to be an arc if and only if for some $u \in \Sigma^*$, $\theta \in (N \cup \Sigma)^*$, either $\sigma \underset{\ell m}{\overset{*}{\Rightarrow}} u\tau$ and $|\tau| \leq k$, or $\sigma \underset{\ell m}{\overset{*}{\Rightarrow}} u\tau\theta$ and $|\tau| = k+1$. Clearly $S \underset{\ell m}{\overset{*}{\Rightarrow}} wA\alpha$ for some w, α if and only if there is a path from S to some node of the form $A\theta$; and this may be determined within **NL**. Thus G is not $LL(k)$ if and only if the following procedure accepts G:

Step 1. Choose distinct productions $A \rightarrow \beta$ and $A \rightarrow \gamma$
Step 2. Choose $x \neq y$ from Σ^{*k}
Step 3. Find a string δ such that a path from S to $A\delta$ exists
Step 4. Accept G if and only if $x \in$ FIRST$_k$ $(\beta\delta)$ and $y \in$ FIRST$_k$ $(\gamma\delta)$. Thus violation of the $LL(k)$ condition may be nondeterministically recognized in log space.

Now we consider $LR(k)$ grammars. For each string $u \in \Sigma^{*k}$, [HSU] (Proposition 2.4) describes a nondeterministic finite automaton $M(u) = (Q, N \cup \Sigma, \delta_u, q_0, F)$, such that G is not $LR(k)$ if and only if for some $u \in \Sigma^*$ and some $\gamma \in (\mathrm{N} \cup \Sigma)^*$, $\delta_u(q_0, \gamma)$ contains two or more final states. Further, $|M(u)| = 0(k^2 n)$ where n is the size of G. It is easily seen that there is a deterministic $\log n$ space procedure which, when given p, $q \in Q$ and $X \in N \cup \Sigma$, will determine whether $q \in \delta_u$ (p, X), and whether q is final (Note: this uses the assumption that G has no ε-productions or nonproductive nonterminals, since calculations of FIRST$_k$ are involved). Clearly we may now determine whether G is not $LR(k)$ in $\log n$ space by simply guessing $u \in \Sigma^{*k}$, guessing a string $\gamma \in (N \cup \Sigma)^*$ a symbol at a time, and simulating two computations by $M(u)$ on γ. Then G is accepted if and only if the two computations each yield final states. □

A Class of Problems between NL and L

Finally we shall consider a class of equivalent problems which are not known to be complete for **NL**, or to possess deterministic log space algorithms. They may be of interest for just this reason.

Let $\langle N, A\rangle$ be an *un*directed graph with $N = \{1, 2, \ldots, n\}$. A node x_k is reachable from x_1 if there is a sequence of nodes $x_1, x_2, \ldots, x_k$ such that $\{x_i, x_{i+1}\}$ for $1 \leq i \leq k-1$ is an arc. The following problem is the counterpart of GAP for undirected graphs.

I. Undirected Graph Accessibility Problem (UGAP)
INPUT: an undirected graph, nodes 1 and n
PROPERTY: n is reachable from 1.

UGAP appears to be easier than GAP; however, it remains unknown whether UGAP is in **L**. In the rest of the section, we consider the following problems which are shown equivalent to UGAP (that is, each is reducible to UGAP, and vice versa).

II. Nonbipartite Graph (NBG)
INPUT: an undirected graph $\Gamma = \langle N, A\rangle$
PROPERTY: Γ is not bipartite, i.e., there exists no function $f\colon N \to \{0, 1\}$ such that for every arc $\{x, y\}$ in $A, f(x) \neq f(y)$.

III. UNSAT$_\times$
INPUT: a set of propositional formulas $C_1, C_2, \ldots, C_p$ such that each C_i $(1 \leq i \leq p)$ is either a literal or an "exclusive or" of two literals.
PROPERTY: $C_1 \wedge C_2 \wedge \ldots \wedge C_p$ is unsatisfiable.

IV. Exact Cover—2 (EC$_2$)
INPUT: a family of sets $\{S_1, S_2, \ldots, S_n\}$ where $S_i \subseteq U$ and every element in U appears at most twice in $S_1, S_2, \ldots, S_n$
PROPERTY: there is no subfamily $\{T_1, T_2, \ldots, T_h\}$ of $\{S_1, S_2, \ldots, S_n\}$ such that T_i's $(1 \leq i \leq h)$ are mutually disjoint and $\bigcup_{i=1}^{h} T_i = U$.

V. Clique Cover—2 (CC$_2$)
INPUT: an undirected graph $\Gamma' = \langle N', A'\rangle$
PROPERTY: N' cannot be covered by two cliques; (a clique is a subgraph with each pair of distinct nodes connected by an arc).

VI. Hitting Set—2 (HS$_2$)
INPUT: sets $U_1, U_2, \ldots, U_m$ where $U_j \subseteq S$ and $|U_j| \leq 2$ for all $l \leq j \leq m$
PROPERTY: there is no subset $W \subseteq S$ such that $|W \cap U_j| = 1$ for all $1 \leq j \leq m$.

NBG, EC$_2$, CC$_2$ and HS$_2$ are, respectively, special cases of the **NP**-complete problems CHROMATIC NUMBER, EXACT COVER, CLIQUE COVER and HITTING SET in [K].

THEOREM 16. *The problems*: UGAP, NBG, $\text{UNSAT}_\times$, EC_2, CC_2 *and* HS_2 *are all equivalent, in the sense that any two problems in the list are reducible to one another.*

Proof. First we show that UGAP $\leq$ NBG. Given a graph $\Gamma = \langle N, A\rangle$, construct a graph $\Gamma' = \langle N', A'\rangle$ as follows:

$$\begin{aligned} N' &= \{i, i' \mid i \in N\} \cup \{x, x' \mid x \in A\} \cup \{w\} \qquad (\text{where } w \notin N \cup A) \\ A' &= \{\{i, x\}, \{x, j\}, \{i', x'\}, \{x', j'\} \mid x = \{i, j) \text{ is an arc in } A\} \cup \\ &\quad \{\{1, 1'\}, \{n, w\}, \{n', w\}\} \end{aligned}$$

Γ' contains a cycle of odd length, and hence is not bipartite, if and only if Γ has a path between nodes 1 and n.

Next, we show that NBG $\leq$ UGAP. Given a graph Γ with nodes $N = \{1, 2, \ldots, n\}$ and arcs A. Construct a graph Γ_3 as follows:

Step 1. Construct $\Gamma_1 = \langle N_1, A_1\rangle$ such that $N_1 = \{\langle i, 0\rangle, \langle i, 1\rangle \mid i \in N\}$ and $A_1 = \{\{\langle j, 0\rangle, \langle k, 1\rangle\}, \{\langle j, 1\rangle, \langle k, 0\rangle\} \mid \{j, k\} \in A\}$. It can be observed that Γ is not bipartite if and only if there is a j such that $\langle j, 0\rangle$ is reachable from $\langle j, 1\rangle$.

Step 2. Construct Γ_2 from Γ_1 by duplicating Γ_1 for n copies. Nodes in k^{th} copy of Γ_2 will be labeled $\langle j, d, k\rangle$ if $\langle j, d\rangle$ is a node in Γ_1 where $d = 0$ or 1 and $1 \leq j \leq n$.

Step 3. Construct Γ_3 by adding two nodes s and t to Γ_2 with the following arcs: $\{\{s, \langle j, 0, j\rangle\} \mid 1 \leq j \leq n\}$ and $\{\{\langle j, 1, j\rangle, t\} \mid 1 \leq j \leq n\}$. If there is a path between s and t, then there must be at least one copy of Γ_1 and Γ_2 which contains a path from $\langle j, 0, j\rangle$ to $\langle j, 1, j\rangle$. The converse is also true. Therefore Γ is not bipartite if and only if there is a path between nodes s and t in Γ_3.

It is straightforward to show that NBG $\leq$ $\text{UNSAT}_\times$ and $\text{UNSAT}_\times$ $\leq$ NBG. The rest of the proof can be obtained by using the corresponding reduction processes in [K]. □

REFERENCES

[AU] AHO, A. V.; ULLMAN, J. D. The theory of parsing, translation and compiling. Vol. 1: Parsing. Prentice Hall, Englewood Cliffs, New Jersey. 1972.

[C] COOK, S. A. The complexity of theorem proving procedures. *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pp. 151–158, 1971.

[CL] CHANG, C. L.; LEE, R. C. T. *Symbolic logic and Mechanical theorem proving*. Academic Press, New York, 1973.

[G] GRIFFITHS, T. V. The unsolvability of the equivalence problem for λ-free nondeterministic generalized sequential machines. *Journal of the ACM* **15** (1968), 409–413.

[HH] HARTMANIS, J.; HUNT, H. B. III. The LBA problem and its importance in the theory of computing. Computer Science Technical Report TR-171, Cornell University, 1973.

[HS] HARTMANIS, J.; SIMON, J. On the structure of feasible computations. Computer Science Technical Report TR-74-210, Cornell University, 1974.

[HSU] HUNT, H. B. III; SZYMANSKI, T.; ULLMAN, J. D. On the complexity of *LR(k)* testing. *Proceedings of 2nd Annual ACM Symposium on Principles of Programming Languages*, pp. 130–138, 1975.

[J] JONES, N. D. Space-bounded reducibility among combinatorial problems. *Journal of Computer and Systems Sciences* **11** (1975), 68–75.

[JL] JONES, N. D.; LAASER, W. T. Problems complete for deterministic polynomial time. To appear in *Theoretical Computer Science*.

[K] KARP, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, edited by R. E. Miller and J. Thatcher, Plenum Press, New York, pp. 85–104, 1972.

[L] LYNCH, N. Personal communication.

[S] SAVITCH, W. J. Relations between nondeterministic and deterministic tape complexities. *Journal of Computer and Systems Sciences* **4** (1970), 177–192.

[S1] SAVITCH, W. J. Nondeterministic Log N space. *Proceedings of 8th Princeton Conference on Information Sciences and Systems*. Princeton University, pp. 21–23, 1974.

[Su] SUDBOROUGH, I. H. Personal Communication.

(Received May 13, 1975 and in revised form December 31, 1975.)