INFORMATION AND CONTROL **64**, 2–22 (1985)

A Taxonomy of Problems with Fast Parallel Algorithms*

STEPHEN A. COOK[†]

Department of Computer Science, University of Toronto, Toronto, Canada M5S 1A4

The class *NC* consists of problems solvable very fast (in time polynomial in log n) in parallel with a feasible (polynomial) number of processors. Many natural problems in *NC* are known; in this paper an attempt is made to identify important subclasses of *NC* and give interesting examples in each subclass. The notion of NC^1-reducibility is introduced and used throughout (problem R is NC^1-reducible to problem S if R can be solved with uniform log-depth circuits using oracles for S). Problems complete with respect to this reducibility are given for many of the subclasses of *NC*. A general technique, the "parallel greedy algorithm," is identified and used to show that finding a minimum spanning forest of a graph is reducible to the graph accessibility problem and hence is in NC^2 (solvable by uniform Boolean circuits of depth $O(\log^2 n)$ and polynomial size). The class LOGCFL is given a new characterization in terms of circuit families. The class DET of problems reducible to integer determinants is defined and many examples given. A new problem complete for deterministic polynomial time is given, namely, finding the lexicographically first maximal clique in a graph. This paper is a revised version of S. A. Cook, (1983, *in* "Proceedings 1983 Intl. Found. Comut. Sci. Conf.," Lecture Notes in Computer Science Vol. 158, pp. 78–93, Springer-Verlag, Berlin/New York). © 1985 Academic Press, Inc.

1. INTRODUCTION

In this paper we are concerned with the class of problems solvable very rapidly (in time polynomial in log n) by a parallel computer with a feasible (i.e., polynomial) number of processors. This class was first identified and characterized by Pippenger [1979], and is now commonly called NC for "Nick's Class" (see Cook, 1981; Dymond and Cook, 1980; Ruzzo, 1981). Since then the class has been shown to include a large and interesting variety of problems. Our task here is to give examples of these problems and classify them according to the methods applicable for demonstrating their inclusion in NC.

* This paper is a revised and expanded version of a paper presented at the International Conference on "Foundations of Computation Theory" held in Borgholm, Sweden, August 21–27, 1983.

[†] Research supported by the Killam Foundation of Canada and the National Sciences and Engineering Research Council of Canada.

A great many formal parallel computer models have appeared in the literature (see Cook, 1981; Vishkin, 1983 for surveys). One common kind is the shared memory computer, in which a number of processors work together synchronously and communicate with a common random access memory. In the event of read or write conflicts in this shared memory several conventions are possible. The particular variation we mention here is the SIMDAG introduced in Goldschlager (1977; 1978; 1982) in which both read and write conflicts are allowed, and the lowest numbered processor succeeds in the case of a write conflict. One reason for favoring these conventions is that the circuitry needed to implement them seems not to be substantially more complicated than that needed to implement a machine which disallows read and write conflicts. A more important reason is that the complexity classes defined in terms of SIMDAG time have nice characterizations in terms of the alternion depth required on circuits (Chandra, Stockmeyer, and Vishkin, 1982) or alternating Turing machines (Ruzzo and Tompa, 1982) (see Propositions 4.6 and 4.7 below).

At the present time no large scale general purpose parallel computers have been built. Although the shared memory model seems like a good way to go (see Schwartz, 1980a) the arbitrariness in the detailed definition makes it unappealing for an enduring mathematical theory. A more attractive model for such a theory is uniform Boolean circuit families (Borodin, 1977). It seems that all real computers will be built from circuits, and hence circuits represent a more fundamental model than the others usually considered. Also the circuit complexity of Boolean functions is an appealing mathematical subject that has been studied since Shannon (1949). Finally, the complexity classes defined in terms of circuit families have a precise characterization in terms of alternating Turing machines (Ruzzo, 1981). Furtunately, the parallel class NC remains the same whether uniform circuit families or shared memory computers are used to define it, although the subclasses NC^k may be different.

One criticism sometimes heard of this general theory of parallel computers is that real circuits must exist in 3-dimensional space and therefore the communication time for one parallel step must be $\Omega(n^{1/3})$. Thus it seems to make no sense to talk about solving problems in time $O(\log^k n)$. The obvious answer to this criticism is that by the same reasoning, even sequential random access memories should have access time $\Omega(n^{1/3})$, and yet it has proven very useful to assume access time $O(1)$ or $O(\log n)$ in mathematical models of such machines.

In this paper we present the following sequence of class inclusions (names to be defined) between NC^1 (the problems solvable by the fastest parallel algorithms) and FP (or "function P," the problems solvable by sequential polynomial time algorithms):

$$NC^1 \subseteq FL \subseteq NL^* \subseteq \begin{matrix} CFL^* \subseteq AC^1 \\ \text{DET} \end{matrix} \subseteq NC^2 \subseteq NC \subseteq FP. \tag{1.1}$$

In Section 2, uniform circuit families and the classes NC^k are defined. In Section 3, examples in the fundamental class NC^1 are given and the notion of NC^1 reduction is introduced. In Section 4 the classes FL, NL^*, CFL^*, and AC^1 are defined with examples, and the "parallel greedy algorithm" is explained. In Section 5 the class DET of problems reducible to computing integer determinants is introduced with examples. In Section 6, examples of problems in FP which are likely not in NC because they are complete for FP are given. In Section 7 the classes random NC^k are defined with examples. Finally, in Section 8 some general remarks and open questions are presented.

2. Uniform Circuit Families

A (*Boolean*) *circuit* α with n inputs and m outputs is a finite directed acyclic graph with nodes (called *gates*) labelled as follows. The circuit α has k "input nodes" with indegree zero labelled $x_1, ..., x_k$, respectively. All other nodes of indegree zero are labelled either 0 or 1. All nodes of indegree one are labelled $\neg$. All other nodes have indegree two and are labelled either $\bigwedge$ or $\bigvee$. Exactly m nodes are labelled output nodes and have labels $y_1, ..., y_m$, respectively. Every input node has at least one path from it to some output node. We use $c(\alpha)$, complexity of α, to denote the number of nodes of α, and $d(\alpha)$, depth of α, to denote the length of the longest path from some input to some output. The circuit α computes a function $f\colon \{0, 1\}^k \to \{0, 1\}^m$ in the obvious way.

In general, we are interested in computing a function $f = \langle f_n \rangle$, where $f_n\colon \{0, 1\}^{g(n)} \to \{0, 1\}^{h(n)}$, and $g(n)$ is monotone strictly increasing and $g(n) = n^{O(1)}$(i.e., $g(n) = O(n^c)$ for some constant c). (The function f will be treated as the union over n of f_n.) A *circuit family* with input size g and output size h is a sequence $\langle \alpha_n \rangle$, where α_n is a circuit with $g(n)$ inputs and $h(n)$ outputs. The family $\langle \alpha_n \rangle$ *computes* the function f iff α_n computes f_n for all n.

For example, the function "directed graph transitive closure" has $g(n) = h(n) = n^2$. The argument x for $f_n(x)$ is a string of n^2 bits representing the $n \times n$ adjacency matrix row by row for an n-node digraph G. The value $f_n(x)$ is a string of n^2 bits representing the transitive closure of G.

In many cases, we do not want our circuit to compute a single-valued function, but rather to find one value of a multiple-valued function. For example, if we want to find a spanning forest in an undirected graph, the solution may not be unique, and any correct answer will do. This motivates the following:

DEFINITION. A *problem* R (with size parameters g and h) (similar to *search problem* in Garey and Johnson, 1979) is a family $\langle R_n \rangle$ of binary relations such that $R_n \subseteq \{0, 1\}^{g(n)} \times \{0, 1\}^{h(n)}$. (The problem R will also be treated as the binary relation which is the union over n of R_n.) A circuit family $\langle \alpha_n \rangle$ *solves* the problem R iff the function $\langle f_n \rangle$ computed by $\langle \alpha_n \rangle$ *realizes* R in the following sense: For each n and each x in $\{0, 1\}^{g(n)}$, if $R_n(x, y)$ holds for some y, then $R_n(x, f_n(x))$ holds.

Note that if we identify a function f with the problem which is the graph of f, then in particular a circuit family solves the function it computes.

To illustrate the above definition, if the problem R is to find a spanning forest, then $R_n(x, y)$ holds whenever x codes an n-node undirected graph G and y codes a spanning forest for G. The family $\langle \alpha_n \rangle$ solves R iff for all n, when the inputs to α_n code an n-node graph G, the outputs to α_n code a spanning forest for G.

In this paper we restrict our attention to "uniform" circuit families; that is, families $\langle \alpha_n \rangle$ for which some algorithm, given n, easily generates the nth circuit α_n. There are several reasons for requiring uniformity. First, one may want to exhibit α_n for various values of n, and this will not always be feasible without some uniformity condition. Second, realistic parallel machine models such as SIMDAGs are naturally uniform and their computing power can be compared to uniform circuit families, but not very well with non-uniform families. Finally, uniform circuit families define complexity classes which have interesting relationships with traditional classes defined by time and space.

Postponing for the moment the exact definition of uniform, we will now define NC.

DEFINITION. NC^k is the set of all problems R solvable by a uniform circuit family $\langle \alpha_n \rangle$ with $c(\alpha_n) = n^{O(1)}$ (i.e., $c(\alpha_n)$ is bounded by some polynomial in n) and $d(\alpha_n) = O(\log^k n)$. $NC = \bigcup_k NC^k$.

The definition of *uniform* we adopt here is the one introduced and called U_{E^*} uniform by Ruzzo (1981). The reason for this choice is that it is the weakest definition for which we have a proof of Proposition 2.1 below for all $k \geqslant 1$. We will not define U_{E^*} uniform here, since the discussion in the following paragraphs explains it sufficiently for our purposes.

Proposition 2.1 concerns *alternating Turing machines* (ATMs) (see Chandra, Kozen, and Stockmeyer, 1981; Ruzzo, 1980). (The reader unfamiliar with ATMs could skip Proposition 2.1.) Our ATMs differ from the usual ones because they compute functions and solve problems instead of recognizing sets. To explain how they do this, let us say that the set A_f associated with the function f: $\{0, 1\}^* \to \{0, 1\}^*$ is $A_f = \{\langle x, i \rangle \mid \text{the } i\text{th bit of } f(x) \text{ is } 1\}$. (Note that a function f is in NC^k iff the characteristic function of A_f is in NC^k and is polynomially bounded; i.e., $|f(x)| = |x|^{O(1)}$.) We say that an ATM M *computes* f iff M recognizes A_f and f is polynomially bounded. Finally, M *solves* a problem R iff M computes some function f which realizes R (see the definition of "$\langle\alpha_n\rangle$ solves R").

PROPOSITION 2.1 (Ruzzo, 1981). *For all $k \geqslant 1$, R is in NC^k iff R is solved by some ATM in time $O(\log^k n)$ and space $O(\log n)$.*

A more common and simpler (though perhaps worse) definition of uniformity is the following: The family $\langle\alpha_n\rangle$ is *log-space uniform* iff some deterministic Turing machine will for all n, when presented with n in binary on its input tape, generate a description $\overline{\alpha_n}$ of α_n on its output tape using work-tape space $O(\log c(\alpha_n))$. We use the notation NC^k (log-space uniform) to refer to NC^k when "uniform" means "log-space uniform."

PROPOSITION 2.2 (Ruzzo, 1981). *$NC^1 \subseteq NC^1$(log-space uniform) and for $k \geqslant 2$, $NC^k = NC^k$(log-space uniform).*

Note that according to Proposition 2.2, $NC = NC$ (log-space uniform). In fact, NC can also be characterized in terms of shared memory computers as those problems solvable in time polynomial in $\log n$ by a polynomial number of processors. Thus the class NC is to a large extent independent of the exact parallel computer model used to define it.

When showing that a problem is in NC^k for $k \geqslant 2$ by constructing a circuit family $\langle\alpha_n\rangle$ which solves it, log-space uniform is a sufficient uniformity condition. However, for $k = 1$, the apparently stronger condition of U_{E^*} uniform is needed. This stronger condition is usually easily met, although in some cases such as the divisibility predicate (see Sect. 3) we do not know how to meet it. The definition of U_{E^*} uniform demands that the so-called "extended connection language" (Ruzzo, 1981) for $\langle\alpha_n\rangle$ can be recognized by an ATM in time $O(\log n)$. A sufficient condition for this is that some deterministic Turing machine, given n in binary, a gate number g, a path $p \in \{L, R\}^*$ of length $O(\log n)$, and parameter y, can determine in space $\sqrt{\log n}$ whether y describes the gate reached in α_n by tracing the path p back towards the inputs from fate g. Since a Turing machine which is allowed space up to the square root of its input length is very powerful, there is usually no difficulty in verifying this condition for families $\langle\alpha_n\rangle$ which are intuitively uniform.

We will have occasion to use a third and weaker notion of uniformity. Let us say that $\langle\alpha_n\rangle$ is *P-uniform* iff the transformation $n \to \overline{\alpha_n}$ is computable by a deterministic Turing machine in time bounded by a polynomial in $c(\alpha_n)$. Then $NC^k \subseteq NC^k$ (log-space uniform) $\subseteq NC^k$ (P-uniform) for all $k \geqslant 1$. Probably P-uniform is the most general notion of uniformity consistent with our earlier criterion that one should be able to feasibly exhibit α_n given n. The integer division problem is in NC^1 (P-uniform) (see Sect. 3). In general, we conjecture that NC is a proper subset of NC (P-uniform).

3. The Class NC^1 and NC^1 Reducibility

Recall that NC^1 consists of all problems solvable by a uniform circuit family of depth $O(\log n)$, where n is the number of input bits (the polynomial size bound is redundant in this case). Examples of functions in NC^1 are: the sum or product of 2 integers of n bits each, the sum of n integers of n bits each, integer or Boolean matrix multiplication, and sorting n integers of n bits each. Circuits for these functions are described in (Savage, 1976; Borodin, Cook, and Pippenger, 1983; Muller and Preparata, 1975).

Integer division (finding the quotient and remainder of two n-bit integers) is not known to be in NC^1 (it is easily in NC^2). Recently it has been shown (Beame, Cook, and Hoover, 1984) to be in NC^1 (P-uniform), along with finding the product of n n-bit integers. The smallest depth known for a log-space uniform family of polynomial size circuits for division is $O(\log n \log \log n)$ (Reif, 1983) (these circuits are probably also U_{E^*} uniform). Also the divisibility relation and finding the product of n integers modulo a small (of size $O(n)$) integer are in NC^1 (log-space uniform) (see Beame, Cook, and Hoover, 1984), thus showing these problems are solvable on a deterministic Turing machine in log space.

In the study of sequential time complexity, polynomial time reducibility (in its two forms "Cook" and "Karp" (Garey and Johnson, 1979)) has become standard, and in the study of space complexity log space reducibility (usually in its "Karp" or many-one form) is used (Jones and Laaser, 1977). In the study of parallel computation, it seems to me that NC^1 reducibility is appropriate. One possible definition is to say that a set A is *many-one* NC^1 reducible to a set B iff there is an NC^1 computable function f such that for all x, $x \in A$ iff $f(x) \in B$. However, here we are interested not just in sets, but in computing functions and solving problems, so the "Turing" (or "Cook") version of reducibility is most useful.

Definition. A problem R is NC^1 *reducible to* S (written $R \leqslant S$) iff there is a U_{E^*} uniform family $\langle \alpha_n \rangle$ of circuits for solving R, where $d(\alpha_n) = O(\log n)$, and α_n is allowed to have *oracle* nodes for S. An oracle node for S is a node with some sequence $\langle y_1, \ldots, y_r \rangle$ of input edges and a sequence $\langle z_1, \ldots, z_s \rangle$ of output edges whose values satisfy

$S(y_1 \cdots y_r, z_1 \cdots z_s)$. For the purpose of defining depth in α_n, this oracle node counts as depth $\lceil \log(r+s) \rceil$.

A similar definition for the case of sets is found in (Wilson, 1983). It is not hard to check that $\leqslant$ is transitive and reflexive. The *closure* C^* of a class C of problems (under $\leqslant$) consists of all problems R such that $R \leqslant S$ for some S in C. The class C is *closed* iff $C = C^*$.

PROPOSITION 3.1. *The class NC^k is closed under $\leqslant$ for all $k \geqslant 1$.*

Proof. Suppose $R \leqslant S$ and $S \in NC^k$. Suppose $\langle \alpha_n \rangle$ realizes the reduction $R \leqslant S$, and $\langle \beta_n \rangle$ solves S in NC^k. Then a family $\langle \gamma_n \rangle$ for solving R in NC^k can be constructed by letting γ_n be α_n with each oracle node for S_m replaced by β_m. To check for example, that $d(\gamma_n) = O(\log^k n)$, consider any path p in γ_n and suppose p hits instances $\beta_{m_1}, \beta_{m_2}, \ldots,$ of circuits substituted for oracle nodes in α_n. Then the length of p is at most $\sum d(\beta_{m_j}) + O(\log n) = O(\sum \log^k m_j) + O(\log n) = O(\log^k n)$, where the last bound follows since $\sum \log m_j = O(\log n)$. The uniformity of $\langle \gamma_n \rangle$ can be proved from the uniformity of $\langle \alpha_n \rangle$ and $\langle \beta_n \rangle$.

DEFINITION. A problem R is *hard* (or *NC^1 hard*) for the class C iff $S \leqslant R$ for all S in C. Further, R is *complete* (or *NC^1complete*) for C iff R is hard for C and $R \in C$.

The following obvious proposition is an abstract analog for NC^1 reducibility of the well-known fact that if a set is NP complete, then its is in P iff $P = NP$.

PROPOSITION 3.2. *If the class C is closed under $\leqslant$ and $C \subseteq D$, and if the problem R is complete for D, then $R \in C$ iff $C = D$.*

Each of the sets C in the list of inclusions (1.1) in Section 1 is in fact closed under $\leqslant$. Hence each time we show that a problem R is complete for a class D occurring later in the list we give evidence that R is not in C. At least a proof that R is in C would solve an open question (namely, whether $C = D$) in the "wrong way."

4. OTHER CLASSES AND THE PARALLEL GREEDY ALGORITHM

Let FL be the class of problems realized by functions computable in space $O(\log n)$ on a deterministic Turing machine, where the output is placed on a special write-only tape which does not participate in the space bound. Let L be the class if sets (regarded as 0-1 functions) recognized in space $O(\log n)$ on a deterministic Turing machine.

PROPOSITION 4.1. *$FL = L^*$, and hence $NC^1 \subseteq FL$.*

To show $FL \subseteq L^*$, note that if a function f is in FL, then the set A_f (defined before Proposition 2.1) is in L, and $f \leqslant A_f$. To show $L^* \subseteq FL$, we use the simulation of depth-bounded circuits by space-bounded Turing machines given in Borodin (1977).

Two examples of problems in FL are (1) undirected graph acyclicity (Cook, 1981), and (2) finding the product of two permutations where input and output permutations are represented as products of disjoint cycles. Both of these are NC^1 complete for FL; the first by an adaptation of a proof in Hong (1980), and the second by an argument in McKenzie and Cook (in preparation). A third problem in FL not known to be in NC^1 is the Boolean formula value problem (Lynch, 1977). This problem is probably not complete for FL, because it can be solved by circuits of depth $O(\log n \log\log n)$ (Gupta, 1985).

Let NL be the class of sets accepted by a nondeterministic machine in space $O(\log n)$. Examples of sets complete for NL are the directed graph accessibility problem, directed k-connectivity, and unsatisfiability of 2-CNF Boolean formulas (Jones, Lien, and Laaser, 1976).

Let NL^* be the closure of NL (as a class of 0–1 functions) under $\leqslant$. Of course every problem complete for NL is also complete for NL^*. Examples of problems for NL^* which are more naturally expressed as functions than sets are transitive closure of a Boolean matrix (i.e., directed graph transitive closure), and the shortest path problem for graphs with positive edge weights expressed in unary notation.

An example of a problem in NL^* which is probably not complete is the knapsack problem with unary weights (Tompa, 1984). An example which may or may not be complete is computing a topological sort of a directed acyclic graph. This problem is certainly complete if one requires the algorithm to state whether or not the input graph is acyclic, but otherwise its completeness is unknown. Topological sort appears in Ruzzo's list (Ruzzo, 1980a) of NC^2 problems and has a published NC^2 algorithm in Dekel, Nassimi, and Sahni, (1981). Recently, Ruzzo (Ruzzo, 1984) devised the following simple NL^* algorithm for topological sort: Compute the transitive closure; sum the columns, giving the number of predecessors of each node; then sort nodes by these numbers.

Here is another interesting example.

PROPOSITION 4.2. *The problem of finding a minimum spanning forest for an n-node undirected graph with n-bit positive integer weights is in NL^* (and hence in NC^2).*

The proof is a parallel version of the sequential greedy algorithm (see Papadimitriou and Steiglitz, 1982, for example). Let $G = (V, E)$ be the input graph, and for any set $E' \subseteq E$ of edges let rank (E') be the number of edges in a spanning forest for $G(E')$, the graph spanned by E'. Then rank (E') is the number of vertices in $G(E')$ minus the number of connected components in $G(E')$. The number of components in $G(E')$ can be computed in NL^* by computing the transitive closure of $G(E')$ in NL^* and using this to count in NC^1 the number of vertices i in $G(E')$ which are not connected to any vertex j in $G(E')$ with $j < i$. The parallel greedy algorithm proceeds by first sorting in NC^1 the edges $\{e_1, e_2,..., e_r\} = E$ of G according to increasing weight and then outputting each edge e_i which satisfies the condition:

$$\text{rank}(e_1,..., e_i) > \text{rank}(e_1,..., e_{i-1}).$$

The fact that these edges form a minimum weight spanning forest follows from Proposition 4.3 below, and from the fact that the rank function defined above satisfies the matroid axioms in a matroid whose bases are the spanning forests of G. Note that $NL^* \subseteq NC^2$ by Borodin (1977).

Proposition 4.3 (Parallel greedy algorithm). *Let E be a finite set with a positive weight associated with each element of E, and suppose $\{e_1,..., e_r\}$ is a list of the elements of E in increasing order of weight. Suppose a function* rank(E') *is defined on the subsets of E which satisfies the matroid axioms. Then the set $B = \{e_i \mid \text{rank}(e_1,..., e_i) > \text{rank}(e_1,..., e_{i-1})\}$ is a matroid base of minimum total weight.*

The proof is similar to the justification of the sequential greedy algorithm (Papadimitriou and Steiglitz, 1982). This proposition is an abstraction of the method described in Borodin, von zur Gathen, and Hopcroft (1982) for finding a column basis for a matrix. The method yields a fast parallel algorithm whenever the rank function can be computed quickly in parallel.

The class *LOGCFL* consists of all sets log space reducible to the class *CFL* of context free languages. (Here A is *log space reducible* to B iff there is some log space computable function f such that for all x, $x \in A$ iff $f(x) \in B$.) Sudborough (1978) characterized *LOGCFL* as those sets accepted by a nondeterministic auxiliary pushdown machine in log space and polynomial time. Sudborough's proof that every set accepted by a nondeterministic auxiliary PDM in log space and polynomial time is log-space reducible to *CFL* actually shows that the reduction is via an NC^1 computable function. Thus $LOGCFL = NC^1CFL$, where the latter class is defined by replacing "log-space reducible" by "many-one NC^1 reducible" (i.e., the reducing function must be NC^1 computable) in the definition of *LOGCFL*. Note that Sudborough's characterization implies that $NL \subseteq LOGCFL$. Ruzzo (1980b) further characterized *LOGCFL* as those sets accepted by an ATM in log space and polynomial tree size, and proved $LOGCFL \subseteq NC^2$.

A third interesting characterization of *LOGCFL* comes from the work of Skyum and Valiant (1981). A Boolean circuit α with negations only at its leaves (inputs) can be regarded as computing a polynomial in its input variables and their negations over the Boolean semiring in which $+$ is $\vee$ and $\bullet$ is $\wedge$. The degree of this polynomial is then by definition the *degree* (denoted degree(α)) of the circuit.

In the following proposition, we assume the circuits α_n have negations only at their leaves.

PROPOSITION 4.4 *LOGCFL is the class of sets recognizable by a uniform family* $\langle \alpha_n \rangle$ *of Boolean circuits with degree* $(\alpha_n) = n^{O(1)}$ *and* $c(\alpha_n) = n^{O(1)}$.

Remark. If the word "uniform" is deleted the class of sets so defined is called *pdC* in Skyum and Valiant (1981).

Proof. That such families $\langle \alpha_n \rangle$ compute sets in *LOGCFL* follows from Ruzzo's (1980b) characterization of *LOGCFL* mentioned above, using the techniques for ATMs simulating circuits developed in Ruzzo (1981). Conversely, we first note that every context-free language can be recognized by such a circuit family by Example 2 p. 250 of Skyum and Valiant (1981). Hence every set many-one NC^1 reducible to *CFL* is so recognized (negations can be pushed to the leaves in the reducing NC^1 circuit and the degree of the resulting NC^1 circuit is always polynomial in n). Finally, the result follows from the earlier remark that $LOGCFL = NC^1CFL$.

The above proof gives rise to an interesting problem complete for *LOGCFL* under many-one NC^1 reducibility, namely, the circuit value problem for monotone Boolean formulas of degree at most n (the number of inputs).

A second complete problem for *LOGCFL* is Greibach's hardest context-free language (Greibach, 1973). Problems in *LOGCFL* which may not be complete are the monotone planar circuit value problem (Dymond and Cook, 1980), bounded valence subtree isomorphism (Ruzzo, 1981), and basic dynamic programming problems (Goldschlager, 1977, 1978, 1982). The latter are more naturally expressed as relations or functions than sets, so it seems that a natural class to consider is CFL^* (the closure of *CFL* under $\leqslant$).

PROPOSITION 4.5. $LOGCFL \subseteq CFL^*$.

This follows immediately from the earlier remark that $LOGCFL = NC^1CFL$.

The above inclusion is proper, because CFL^* contains functions other than 0–1 functions. In addition, it is reasonable to conjecture that not all 0–1 functions in CFL^* are in $LOGCFL$. This is because the sets (i.e., the 0–1 functions) in CFL^* are closed under complementation, but while the graph accessibility problem is in NL and therefore in $LOGCFL$, its complement does not appear to be in $LOGCFL$.

An example of a dynamic programming problem in CFL^* is computing the minimum cost order of multiplying a string of n matrices (see Aho, Hopcraft, and Ullman, 1974, for a dynamic programming solution to the problem and Goldschlager, 1977, 1978, 1982, for the method of putting such problems into CFL^*). Of course, the problems complete for $LOGCFL$ mentioned above are also complete for CFL^*.

It turns out that all functions in CFL^* can be computed on a SIMDAG (see the Introduction) in time $O(\log n)$. To state a more general form of this result we introduce the following terminology.

DEFINITION. AC^k, for $k = 1, 2, \ldots$, is the class of all problems solvable by an ATM in space $O(\log n)$ and alternation depth $O(\log^k n)$.

PROPOSITION 4.6. (Ruzzo and Tompa). *AC^k is the class of all functions computable on a* SIMDAG *in $O(\log^k n)$ time with $n^{O(1)}$ processors.*

A similar characterization of "nonuniform AC^k", defined in terms of circuits with unbounded fan-in for "and" and "or," appears in Chandra, Stockmeyer, and Vishkin (1982). In fact, AC^k itself has a characterization analogous to the definition of NC^k (see Proposition 2.1).

Let us say the *direct connection language* (DCL) (see Ruzzo, 1981), for a family $\langle \alpha_n \rangle$ of circuits with unbounded fan-in for "and" and "or" consists of codes for all triples $\langle n, u, v \rangle$ such that node u is an input to node v in α_n, together with codes for triples $\langle n, u, l \rangle$, where the label, l indicates what sort of gate node u is in α_n. We say that $\langle \alpha_n \rangle$ is *uniform* if this DCL has deterministic space complexity $O(\log n)$.

PROPOSITION 4.7 (Cook and Ruzzo, 1983). *AC^k consists of those problems solvable by uniform unbounded fan-in circuit families in $O(\log^k n)$ depth and $n^{O(1)}$size.*

It turns out that the above proposition still holds if the definition of *uniform* is weakened to simply require that the DCL is in AC^k.

By Proposition 2.1 of Ruzzo (1981) it follows that $AC^k \subseteq NC^{k+1}$. One way to see this using unbounded fan-in circuits is that each "or" gate or "and" gate has fan-in $n^{O(1)}$ and hence can be replaced by a tree of depth $O(\log n)$ of fan-in two gates. In particular, $AC^1 \subseteq NC^2$.

Ruzzo (1980b) showed $LOGCFL \subseteq AC^1$. Since AC^1 is closed under $\leqslant$ (as can be seen from Proposition 4.7), we have

PROPOSITION 4.8. $CFL^* \subseteq AC^1$

By putting Propositions 4.6 and 4.8 together we obtain

COROLLARY 4.9. *All problems described so far are solvable by* SIMDAG's *in* $O(\log n)$ *time with* $n^{O(1)}$ *processors. This applies in particular to context-free language recognition and finding a minimum spanning forest in an undirected graph* (Awerbuch and Shiloach, 1983; Reif, 1982).

An example in AC^1 not known to be in CFL^* is the shortest path problem in an undirected graph with positive integer edge weights presented in binary notation. That this is in AC^1 follows by min-plus matrix powering (Aho, Hopcroft, and Ullman, 1974). If the edge weights are presented in unary notation, the problem is in NL^*.

5. THE CLASS DET

Let *intdet* be the problem of computing $\det(A)$ given an $n \times n$ matrix A of n-bit integers, and let *matpow* be the problem of computing the powers $A^1, A^2, \ldots, A^n$, given such an A. Since integer matrix multiplication is in NC^1 it is easy to see that *matpow* is in NC^2.

Csansky (1976) was the first to show that the problem of computing the determinant of an $n \times n$ matrix over a field of characteristic zero can be solved using an *algebraic* circuit of depth $O(\log^2 n)$ and polynomial size. This statement does not imply the fact that *intdet* is in NC^2 because the algebraic circuit charges depth one for plus and times, whereas these operations require log depth for integers using Boolean circuits. However, a study of Csansky's method, using the fact that iterated integer addition is in NC^1, does show that *intdet* is in NC^2. Recently Berkowitz (1984) gives an alternative algebraic circuit for determinant which makes it clear that $intdet \leqslant matpow$ (recall $\leqslant$ means "is NC^1 reducible to"). Borodin *et al.* (1983) gives an explicit construction showing how to adopt Berkowitz's algebraic circuits for determinant to give NC^2 Boolean circuits not only for *intdet*, but for bit representations of determinants over other rings, such as the polynomials with integer coefficients. (For a general discussion of the algebraic versus bit points of view in complexity theory, see Borodin, 1982.)

Since many problems in NC^2 are reducible to *intdet*, we make the following

DEFINITION. $DET = \{intdet\}^* = \{R \mid R \leqslant intdet\}$.

It is clear that the problem "iterated integer product" (given n-bit integers $a_1, a_2, ..., a_n$ compute their product $a_1 a_2 \cdots a_n$) is in *DET* by computing the determinant of the matrix with $a_1, ..., a_n$ on the diagonal and zeroes elsewhere. Since integer division $\leqslant$ *intdet* (see Hoover, 1979; Reif, 1983; or Beame, Cook, and Hoover, 1984), it follows that the former problem is in *DET*.

A large class of problems in *DET* comes from the following:

PROPOSITION 5.1. $NL^* \subseteq DET$.

Proof. It suffices to show that the graph accessibility problem is reducible to *intdet*, since the former is complete for *NL*. Let A be the adjacency matrix of an n-node digraph G and assume A has zeros on the diagonal. Then the i, jth element of A^k, where A is treated as an integer matrix, is the number of paths of length k from i to j in G. For any ε with $0 < \varepsilon < \|A\|^{-1}$ (where $\|A\|$ is the norm of A) we have, setting $M = I - \varepsilon A$,

$$M^{-1} = (I - \varepsilon A)^{-1} = I + \varepsilon A + (\varepsilon A)^2 + \cdots.$$

Therefore, the i, jth element of M^{-1} is nonzero iff there is a path from i to j in G. Since $M^{-1} = \operatorname{adj}(M)/\det(M)$, and we can take $\varepsilon = 1/n$ (since $\|A\| < n$), we have, multiplying M by $1/\varepsilon$,

$$\exists \text{ path in } G \text{ from } 1 \text{ to } n \text{ iff } \det((nI - A)[n \mid 1]) \neq 0,$$

where $[n \mid 1]$ indicates the deletion of the nth row and first column.

PROPOSITION 5.2. *The following problems are complete for DET*:

(1) *intdet* (*integer determinant*)

(2) *matpow* (matrix powering)

(3) *itmatprod* (*iterated matrix product*)

Input: *n $n \times n$ matrices with n-bit integer entries*
Output: *their product*

(4) *matinv* (*integer matrix inverse*)

Input: *$n \times n$ matrix A with n-bit integer entries*
Output: *A^{-1} in the form $\langle \operatorname{adj}(A), det(A) \rangle$, where all entries are integers with $n^2 + \lceil \log_2 n \rceil + 1$ bits.*

Proof. (a) *intdet* $\leqslant$ *matpow*: (see Berkowitz, 1984).

(b) *matpow* $\leqslant$ *matinv*: (see Borodin, 1982). Let N be the $n^2 \times n^2$ matrix consisting of $n \times n$ blocks which are all zero except for $n-1$ copies of A above the diagonal of zero blocks. Then $N^n = 0$ and $(I-N)^{-1} = I + N + N^2 + \cdots + N^{n-1} =$

$$\begin{bmatrix} I & A & A^2 & \cdots & A^{n-1} \\ 0 & I & A & \cdots & A^{n-2} \\ \vdots & & & & \vdots \\ 0 & & & \cdots & I \end{bmatrix}.$$

(c) *matinv* $\leqslant$ *intdet*: All entries of adj(A) are determinants of minors of A with appropriate sign.

(d) *matpow* $\leqslant$ *itmatprod*: Obvious.

(e) *itmatprod* $\leqslant$ *matpow*: Let $A_1,..., A_n$ be $n \times n$ matrices, and let B be an $(n^2+n) \times (n^2+n)$ matrix consisting of $n \times n$ blocks which are all zero except for $A_1,..., A_n$ appearing above the diagonal of zero blocks. Then B^n has the product $A_1 A_2 \cdots A_n$ in the upper right corner.

This completes the proof of Proposition 5.2. Clearly the inverse of matrices over the rationals (expressed as integer pairs $\langle$numerator, denominator$\rangle$) and solutions of nonsingular systems of linear equations over the rationals are also in *DET*. It is not clear that these problems are complete for *DET* (unless they are artificially formulated) because although the determinant of an integer matrix can be expressed as a quotient of integers using solutions to these problems, I do not see how to reduce integer division to them.

One can add many other problems to *DET*. The method in Berkowitz (1984) actually shows how to compute the coefficients of the characteristic polynomial $\det(\lambda I - A)$ given an oracle for matrix powering, so this problem is in *DET* (and complete). This shows how to compute the coefficients of a polynomial $(x-a_1)(x-a_2)\cdots(x-a_n)$ in *DET*, so the Lagrange interpolation polynomials are in *DET*. Hence polynomial interpolation over the rationals is in *DET*. This allows us to solve all the problems listed in Proposition 5.2 in *DET* even when the matrix entries are polynomials with a fixed number of variables (or rational functions) over the integers or rationals by polynomial evaluation and interpolation. Hence by Borodin, Cook, and Pippenger (1983) the problems of computing the completion of a stochastic matrix and simulating a $\log n$ space-bounded probabilistic Turing machine are in *DET*.

Another source of problems in *DET* is Borodin *et al* (1982) and von zur Gathen (1983) in which algebraic reductions to computing determinants are given. From these we can conclude, for example, that computing the greatest common divisor of n univariate polynomials over the rationals is in *DET*. Also according to Ibarra, Moran, and Rosier (1980) computing the rank of a matrix over the rationals can be reduced to computing characteristic polynomials, so this problem is in *DET*. In Borodin *et al.* (1982) it is shown how to find a column basis for a matrix (using the parallel greedy algorithm: see Proposition 4.3) once the rank can be computed, and show how this can be used to find a general solution to a singular system of linear equations. Hence this problem is in *DET*, in case the ground field is the rationals.

6. Problems NC^1 Complete for FP

Let FP be the class of all problems realized (see Sect. 2) by functions computable in polynomial time on a deterministic Turing machine. Since at present we cannot prove $FP \neq NC^1$ the best way to indicate that a problem in FP is probably not in NC is to prove that it is complete for FP. (Such arguments are usually stated with respect to log space reducibility, but in fact the proof usually shows that NC^1 reducibility applies as well.) If R is NC^1 complete for FP and R is in NC, then $FP = NC$, an unlikely result (see Proposition 3.2).

Examples of problems complete for FP are the circuit value problem (Ladner, 1975) (either monotone or planar Goldschlager, 1977), linear programming (Khachian, 1979; Dobkin, Lipton, and Reiss, 1979), and maximum network flow (with capacities given in binary notation) (Goldschlager, Shaw, and Staples, 1982). See Jones and Laaser (1977) for several others. Here is one new one:

Proposition 6.1. *Finding the lexicographically first maximal clique in an undirected graph is NC^1 complete for FP.*

Proof. We show how to reduce the monotone circuit value problem to the above problem. Given a monotone Boolean circuit α (with each input assigned 0 or 1) we construct a graph G such that each gate v of α is associated with node v' and v'' of G. This will be done in such a way that the lexicographically first clique C of G includes v' iff the value $\nu(v)$ of v in α is 1, and C includes v'' if $\nu(v) = 0$. We assume that the gates and inputs of α are ordered topologically with the inputs first and the output last. If the gates and inputs are ordered $\langle v_1, v_2, ..., v_k \rangle$ then the nodes of G are ordered $\langle v_1', v_1'', v_2', v_2'', ..., v_k', v_k'' \rangle$, except the order of (v_i', v_i'') may be reversed. If v is an input labelled 1 then v' is adjacent to all precding nodes and v'' is adjacent to no preceding nodes. If v is an input labelled 0 then these conditions are reversed, except v' and v'' are never adjacent. If v is an "and" gate with inputs u and w, then v' is adjacent to all preceding nodes except u'' and w'', and v'' is adjacent to all preceding nodes except v'. If v is an "or" gate, then v'' precedes v', v'' is adjacent to all preceding nodes except u' and w', and v' is adjacent to all preceding nodes except v''.

Proposition 6.1 suggests that finding the first maximal clique in a graph is not solvable in *NC*. An interesting contrast to this result is provided by the recent result of Karp and Wigderson (1984) who show that finding *some* maximal clique is in fact in *NC*.

7. Random *NC*

The class *BPP* (Gill, 1977) can be defined as the class of all sets recognizable in polynomial time by a probabilistic Turing machine with error probability at most $\frac{1}{4}$. Similarly one can define *RNC* (random *NC*) to be the class of problems solvable by probabilistic circuits in polylog depth and polynomial time. More precisely, a problem R is in RNC^k iff it is realized by a function f computed by a uniform family $\langle \alpha_n \rangle$ of probabilistic circuits with bitwise error probability at most $\frac{1}{4}$, where $d(\alpha_n) = O(\log^k n)$ and $c(\alpha_n) = n^{O(1)}$. Here a *probabilistic circuit* is a Boolean circuit with ordinary inputs x and "coin tossing" inputs y. The probability that a particular output bit v is 1 is defined to be the fraction of input strings y such that the value of v is 1 when α_n has inputs (x, y). If each bit of $f(x)$ is computed correctly with probability at least $\frac{3}{4}$, then one can arrange many circuits in parallel each computing the same bit, and a majority vote can be taken to obtain a reliable value. Thus if R is in RNC^k under the above definition, then for each l some other uniform family $\langle \beta_n \rangle$ of probabilistic circuits with polynomial size and $O(\log^k n)$ depth computes all bits of the function f realizing R correctly with the probability of one or more errors at most 2^{-n^l}.

If R is in RNC^k, then using the techniques of Adleman (1978) one can show that R is in "nonuniform NC^k", that is, NC^k with the uniformity restriction removed.

NC is contained in *RNC* almost by definition, but it is of course not clear whether *RNC* is a subclass of *NC*, or even of *FP*, although certainly $RNC \subseteq BPP$. Since it seems unlikely that $FP \subseteq RNC$, a proof that R is hard for *FP* (under NC^1, NC^k, RNC^k, or log-space reducibility) is a strong indication that R is not in *RNC*. (Equivalently a proof that R is in *RNC* is a strong indication that R is not hard for *FP*.)

A number of problems in RNC^2 appear in Borodin *et al.* (1982, see also von zur Gathen, 1983). Examples are finding the rank of a matrix (and solving possibly singular systems of linear equations) over a finite field, and finding the size of a maximum matching in a bipartite graph (or an arbitrary undirected graph (Feather, 1984). These methods are extended in Feather to show that the size of the maximum flow in a network with edge capacities expressed in unary is in RNC^2. (When edge capacities are expressed in binary the problem is complete for FP (Golschlager *et al.*, 1982). In Schwartz (1980b) it is shown that testing the singularity of a matrix of polynomials in many variables is in RNC^2. Recently it has been shown (McKenzie, 1984; McKenzie and Cook, 1983) that the abelian permutation group membership problem and related problems are in RNC^3.

8. Conclusion and Open Questions

The title of this paper advertises more than I have delivered, since I have mainly discussed problem in NC^2 (and a few in RNC^2) as opposed to problems in general with fast parallel algorithms. We might define the latter class to be those problems which are solvable in parallel in time polynomial in $\log n$ ("polylog time") with no restriction on the number of processors (or circuit size, in the circuit model). This class is, by the "parallel computation thesis" (Goldschlager, 1977, 1978, 1982; Pratt and Stockmeyer, 1976; Borodin, 1977), equal to polylog space. I find it interesting that very few natural problems in the last class have come to my attention which are not in NC. One notable exception is the problem of determining whether two groups, presented by their multiplication tables, are isomorphic. This can be solved in space $O(\log^2 n)$ by taking advantage of the fact that a group with n elements has a set of generators of size at most $\log_2 n$ (Lipton, Snyder, and Zalcstein, 1976; Miller, 1978). I know of no NC solution to this problem, or even any polynomial time solution.

Within NC (and RNC) I have stuck to NC^2 (and RNC^2), partly because when I wrote the earlier version (Cook, 1983) of this paper there were few natural examples known to be RNC and not known to be in RNC^2. Recently more examples have come to light, and it is worth mentioning some of these (and some earlier ones). First are problems of the form find a maximal (or minimal) subset subject to restrictions. For example, Lev (1980) shows that finding a maximal (not maximum) matching in a bipartite graph is in NC^5, and recently Karp and Wigderson (1984) show that finding a maximal independent set in a graph is in NC^5.[1] The techniques in the last paper may well apply to other maximality problems and open up a new field of research: Classifying maximality problems according to their parallel complexity.

Other examples in RNC but maybe not in RNC^2 are the Abelian per-

[1] Improved to NC^2 in Luby, M. (1985), A simple algorithm for the maximal independent set problem, *in* "Proc. 17th ACM Sympos. Theory of Comput." pp. 1–10.

mutation group membership problem and related problems (shown to be in RNC^3 in McKenzie and Cook (1983), and the problem of recognizing whether a permutation group is nilpotent (shown to be in NC^4 in McKenzie, 1984). In fact, these two references give other such problems, and suggest there may be a rich class of examples.

We summarize below the class inclusions mentioned in this paper:

$$NC^1 \subseteq FL \subseteq NL^* \subseteq \begin{array}{c} CFL^* \subseteq AC^1 \\ DET \end{array} \subseteq NC^2 \subseteq NC$$

$$\subseteq \begin{array}{c} DSPACE((\log n)^{O(1)}) \\ FP \end{array}.$$

Natural examples complete for each of the above classes (suggesting that the inclusion immediately to the left of the class might be proper) have been given, with the exceptions of AC^1, NC^2, NC, and $DSPACE((\log n)^{O(1)})$. There is provably no complete problem for this last class, and there is none either for NC unless $NC = NC^k$ for some k. An intriguing open question is to find natural complete problems for AC^1 and NC^2. (The word "natural" precludes having "$\log n$" or "$\log^2 n$" appear in the statement of a problem. Of course the circuit value problem for circuits of depth at most $\log^2 n$ is complete for NC^2). It is interesting to note that all our examples in NC^2 are in fact either in AC^1 or in DET. The question of whether $DET = NC^2$ has an interesting algebraic analog (see Valiant, 1979; Valiant, Skyum. Berkowitz, and Rackoff, 1983).

It would be nice to show that DET and CFL^* are comparable. It seems unlikely that $DET \subseteq AC^1$ (and hence unlikely that $DET \subseteq CFL^*$), since interger matrix powering is in DET, and if A^n is computed by repeated squaring then $\log n$ stages are required and each stage requires unbounded alternation depth by Furst, Saxe, and Sipser (1981).

Of course it would require a breakthrough in complexity theory to prove $NC^1 \neq FP$, and hence a breakthrough to prove any two of the above classes are unequal (excluding $DSPACE$ $((\log n)^{O(1)})$).

It would be nice to show that the problems in RNC mentioned in Section 7 are in NC. Among the interesting problems in FP not known to be either complete for FP or in (random) NC are integer greatest common divisors, computing $a^b \bmod c$ (a, b, c positive integers presented in binary) and testing membership in an arbitrary permutation group (McKenzie, 1984). Another such problem mentioned in Cook (1983) was solved recently by Karp, Upfal, and Wigderson, (1985). They showed how to find a maximum matching in a graph in RNC.

Acknowledgments

I am indebted to Larry Ruzzo, not only for circulating a list of problems in NC^2 several years ago, but for carefully reading the earlier version of this paper and suggesting many illuminating additions and improvements. My thanks also to Allan Borodin and Martin Tompa who each supplied a good list of improvements, and to Patrick Dymond for helpful discussions, and to Mike Luby for suggesting improved notation.

References

Adleman, L. (1978), Two theorems on random polynomial time, *in* "Proc. 19th IEEE Found. Comput. Sci.," pp. 75–83.

Aho, A. V., Hopcroft, J. E., and Ullman, J. D., (1974), "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass.

Awerbuch, B., and Shiloach, Y. (1983), New connectivity and MSF algorithm for ultracomputer and PRAM, preprint, IBM-Israel Scientific Center, Technion, Haifa.

Berkowitz, S. J. (1984), On computing the determinant in small parallel time using a small number of processors, *Inform. Process. Lett.* **18**, 147–150.

Borodin, A. (1977), On relating time and space to size and depth, *SIAM J. Comput.* **6**, 733–744.

Borodin, A. (1982) Structured vs. general models in computational complexity, *in* "Logic and Algorithmic," Enseign. Math. (No. 30) pp. 47–65, Univ. Geneva, Geneva.

Beame, P. W., Cook, S. A., and Hoover, H. J. (1984), Log depth circuits for division and related problems, *in* "Proc. 17th IEEE Found. Comput. Sci."

Borodin, A., Cook, S. A., and Pippenger, N. (1983), Parallel computation for well-endowed rings and space-bounded probabilistic machines, *Inform. and Control* **58**, 113–136.

Borodin, A., von zur Gathen, J. and Hopcroft, J. (1982), Fast parallel matrix and GCD computations, *Inform. and Control* **52**, 241–256.

Chandra, A. K., Kozen, D. C., and Stockmeyer, L. J. (1981), Alternion, *J. Assoc. Comput. Mach.* **28**, No. 1, 114–133.

Chandra, A. K., Stockmeyer, L. J., and Vishkin, U. (1982), Complexity theory for unbounded fan-in parallelism, *in* Proc. 23rd IEEE Found. Comput. Sci.," pp. 1–13.

Cook, S. A., (1981), Towards a complexity theory of synchronous parallel computation, *Enseign. Math.* **27**, 99–124.

Cook, S. A. (1983), The classification of problems which have fast parallel algorithms, *in* Proc. 1983 International FCT Conference," Lecture notes in Computer Science Vol. 158, pp. 78–93, Springer-Verlag, Berlin/New York.

Cook, S. A., and Ruzzo, W. L. (1983), unpublished theorem.

Csanky, L. (1976), Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5**, 618–623.

Dymond, P. W., and Cook, S. A. (1980), Hardware complexity and parallel computation, *in* "Proc. 21st IEEE Found. Comput. Sci.," 360–372.

Dobkin, D., Lipton, R. J. and Reiss, (1979), Linear programming is log–space hard for *P*, *Inform. Process. Lett.* **8** 96–97.

Dekel, E., Nassimi, D., and Sahni, S. (1981), Parallel matrix and graph algorithms, *SIAM J. Comput.* **10** 657–675.

Feather, T., (1984), M.Sc. thesis, Department of Computer Science, University of Toronto.

Furst, M., Saxe, J., and Sipser, M. (1981), Parity, circuits, and the polynomial-time hierarchy, *in* "Proc. 22nd IEEE Found. Comput. Sci.," pp. 260–270.

GAREY, M. R., AND JOHNSON, D. S. (1979), Computers and Intractability: A Guide to the Theory of *NP*-Completeness," Freeman, San Francisco.

VON ZUR GATHEN, J. (1983), Parallel algorithms for algebraic problems, *in* "Proc. 15th ACM Sympos. Theory of Comput.," pp. 17–23.

GILL, J. (1977), Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6**, 675–695.

GOLDSCHLAGER, L. M. (1977; 1978; 1982) "Synchronous Parallel Computation," Ph.D. thesis, University of Toronto; *in* "Proc. ACM Sympos. Theory of Comput.," pp. 89–94; *Assoc. Comput. Mach.* **29**, No. 4, 1073–1086.

GOLDSCHLAGER, L. M. (1977), The monotone and planar circuit value problems are log space complete for *P*, *SIGACT News* **9**, No. 2, 25–29.

GREIBACH, S. A. (1973), The hardest context-free language, *SIAM J. Comput.* **2**, 304–310.

GOLDSCHLAGER, L. M., SHAW, R. A., AND STAPLES, J. (1982), The maximum flow problem is log space complete for *P*, *Theoret. Comput. Sci.* **21**, 105–111.

GUPTA, A. (1985), M.Sc. thesis, Dept. of Computer Science, University of Toronto.

HONG, J. W. (1980), On some space complexity problems about the set of assignments satisfying a boolean cormula, *in* "Proc. 12th ACM Sympos. Theory of Comput.," pp. 310–317.

HOOVER, H. J. (1979), "Some Topics in Cicrcuit Complexity," M.Sc. thesis, University of Toronto, Department of Computer Science, Department of Computer Science Technical Report 139/80.

IBARRA, O. H., MORAN, S., AND ROSIER, L. E. (1980), A note on the parallel complexity of computing the rank of order *n* matrices, *Inform. Process. Lett.* **11**, 162.

JA' JA', J., AND SIMON, J. (1982), Parallel algorithms in graph theory: Planarity testing, *SIAM J. Comput.* **11**, 314–328.

JONES, N. D., AND LAASER, W. T. (1977), Complete problems for deterministic polynomial time, Theoretical Computer Science **3**, 105–117.

JONES, N. D., LIEN, Y. E., AND LAASER, W. T. (1976), New problems complete for nondeterministic log space, *Math. Systems Theory* **10** 1–17.

KHACHIAN, L. G. (1979), A polynomial time algorithm for linear programming, *Dokl. Akad. Nauk SSSR* **244** No. 5 1093–96; transl. in *Soviet Math. Dokl.* **20**, 191–194.

KARP, R. M., UPFAL, E., AND WIGDERSON, A. (1985), Constructing a perfect matching is in random NC, *in* "Proc. 17th ACM Sympos. Theory of Comput.," pp. 22–32.

KARP, R. M., AND WIGDERSON, A. (1984), A fast parallel algorithm for the maximal independent set problem, *in* "Proc. 16th ACM Sympos. Theory of Comput.," pp. 266–272.

LADNER, R. E. (1975), The circuit value problem is log space complete for *P*, *SIGACT News* **7**, No. 1, 18–20.

LEV., G. (1980), "Size Bounds and Parallel Algorithms for Networks," Doctoral thesis, Report CST-8-80, Dept. of Computer Science, University of Edinburgh.

LYNCH, N. (1977), Log space recognition and translation of parenthesis languages, *J. Assoc. Comput. Mach.* **24**, No. 4, 583–590.

LIPTON, R. J., SNYDER, L., AND ZALCSTEIN, Y. (1976), "The Complexity of the Word and Isomorphism Problems for Finite Groups," Tech. Rep. 91/76, Yale University.

MCKENZIE, P. (1984), "Parallel Complexity and Permutation Groups," Ph.D. thesis, University of Toronto, Department of Computer Science.

MILLER, G. L. (1978), On the $n^{\log n}$ isomorphism technique, *in* "Proc. 10th Sympos. Theory of Comput.," pp. 51–58.

MCKENZIE, P., AND COOK, S. A. (1983), The parallel complexity of the Abelian permutation group membership problem, *in* "Proc. 24th IEEE Found of Comput. Sci.," pp. 154–161.

MCKENZIE, P., AND COOK, S. A. (1985), The parallel complexity of Abelian Permutation group problems. University of Toronto, Dept. of Computer Science, Technical Report No. 181/85.

MULLER, D. E., AND PREPARATA, F. P. (1975), Bounds to complexities of networks for sorting and switching, *J. Assoc. Comput. Mach.* **22** No. 2, 195–201.

PIPPENGER, N. (1979), On simultaneous resource bounds (preliminary version), *in* "Proc. 20th IEEE Found. of Comput. Sci.," pp. 307–311.

PAPADIMITRIOU, C. H., AND STEIGLITZ, K. (1982), "Combinatorial Optimization: Algorithms and Complexity," Prentice–Hall, Englewood Cliffs, N. J.

PRATT, V. R., AND STOCKMEYER, L. J. (1976), A characterization of the power of vector machines, *J. Comput. System Sci.* **12** 198–221.

REIF, J. H. (1982), Symmetric complementation, *in* "Proc. 14th ACM Sympos. Theory of Comput." pp. 201–214.

REIF, J. H. (1983), Logarithmic depth circuits for algebraic functions, *in* "24th IEEE Found. of Comput. Sci.," pp. 138–145; revised version: Preprint (1984).

RUZZO, W. L. (1980a), unpublished list of problems in NC^2.

RUZZO, W. L. (1980b), Tree-size bounded alternation, *J. Comput. System Sci.* **21** No. 2, 218–235.

RUZZO, W. L. (1981), On uniform circuit complexity, *J. Comput. System Sci.* **22** No. 3 365–383.

RUZZO, R. L. (1984), private communication.

RUZZO, W. L., AND TOMPA, M. (1982), unpublished result. See Stockmeyer, L., and Viskin, I., "Simulation of Parallel Random Access Machines by Circuits," Report RC-9362. IBM Research, Yorktown Heights, N.Y.

SAVAGE, J. E. (1976), "The Complexity of Computing," Wiley, New York.

SKYUM, S., AND VALIANT, L. G. (1981), A complexity theory based on Boolean algebra, *in* "Proc. 22nd IEEE Found. of Comput. Sci.," pp. 244–253.

SCHWARTZ, J. T. (1980a), Ultracomputers, *ACM Trans. Program. Lang. Systems* **2**, No. 4 484–521.

SCHWARTZ, J. T. (1980b), Probabilistic algorithms for verification of polynomial identities, *J. Assoc. Comput. Mach.*, **27**, No. 4 701–717.

SHANNON, C. E. (1949), The synthesis of two terminal switching circuits, *BSTJ* **28** 59–98.

SUDBOROUGH, I. H. (1978), On the tape complexity of deterministic context-free languages, *J. Assoc. Comput. Mach.* **25** No. 3, 405–414.

TOMPA, M. (1984), private communication.

VALIANT, L. G. (1979), Completeness classes in algebra, *in* "Proc. 11th ACM Sympos. Theory of Comput.," pp. 249–261.

VALIANT, L. G., SKYUM, S., BERKOWITZ, S., AND RACKOFF, C. (1983), Fast parallel computation of polynomials using few processors, *SIAM J. Comput.* **12** No. 4, 641–644.

VISHKIN, U. (1983), Synchronous parallel computation—A survey, preprint Courant Institute, New York University.

WILSON, C. (1983), Relativized circuit complexity, *in* "Proc. 24th IEEE Found. of Comput. Sci.," pp. 329–342.