SIAM J. COMPUT.
Vol. 13, No. 2, May 1984

013

CONSTANT DEPTH REDUCIBILITY*

ASHOK K. CHANDRA†, LARRY STOCKMEYER‡ AND UZI VISHKIN§

Abstract. The purpose of this paper is to study reducibilities that can be computed by combinational logic networks of polynomial size and constant depth containing AND's, OR's and NOT's, with no bound placed on the fan-in of AND-gates and OR-gates. Two such reducibilities are defined, and reductions and equivalences among several common problems such as parity, sorting, integer multiplication, graph connectivity, bipartite matching and network flow are given. Certain problems are shown to be complete, with respect to these reducibilities, in the complexity classes deterministic logarithmic space, nondeterministic logarithmic space, and deterministic polynomial time. New upper bounds on the size-depth (unbounded fan-in) circuit complexity of symmetric Boolean functions are established.

Key words. circuit complexity, unbounded fan-in circuits, reducibility, complete problems

1. Introduction. Reducibility is a key concept in the theory of computation. In recursion theory, effective reducibility is useful in proving problems decidable or undecidable. In complexity theory, polynomial-time reducibility is central to the concept of NP-completeness [5], [14], [9] and permits one to show that the complexities of different problems are related even though the exact complexities of the problems are unknown. Similarly, reducibility can be used to establish upper or lower bounds on computational complexity: If problem A is reducible to problem B, then a lower bound on the complexity of A translates to a similar lower bound on the complexity of B, and an upper bound on B translates to an upper bound on A; this requires that the computational resources used in computing the reducibility function be negligible compared to the upper or lower bounds being proved. For investigating certain problems such as the relationship between deterministic and nondeterministic logarithmic space, polynomial-time reducibility is too weak so it has been strengthened to logspace reducibility [13], [21].

The purpose of this paper is to study reducibilities that can be computed by combinational logic circuits containing AND's, OR's and NOT's with no bound placed on the fan-in of AND-gates and OR-gates (call these simply *circuits*) where the circuit has polynomial size and constant depth. We define two such reducibilities and investigate how several common problems such as parity,

*Received by the editors September 14, 1982, and in revised form July 25, 1983. Portions of this paper have been reprinted, with permission, from "A complexity theory for unbounded fan-in parallelism" by A. K. Chandra, L. J. Stockmeyer and U. Vishkin, appearing in the Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, 1982, pp.1-13, © 1982 IEEE. This paper was typeset at the IBM San Jose Research Laboratory.

†Mathematical Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.

‡Computer Science Department, IBM Research Lab. K51/281, San Jose, California 95193. Work performed in part at the IBM Thomas J. Watson Research Center.

§Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, New York 10012. Work performed while the author was visiting the IBM Thomas J. Watson Research Center, while on leave from the Department of Computer Science, Technion, Haifa, Israel.

sorting, integer multiplication, graph connectivity, bipartite matching and network flow, are related by these reducibilities. There are two motivations for studying constant-depth reducibility. Interest was drawn to size-depth complexity for circuits by a recent result of Furst, Saxe and Sipser [8] showing that the parity function cannot be computed by any polynomial-size constant-depth circuit. Furst, Saxe and Sipser also give a few polynomial-size constant-depth reductions from parity to other problems, thus showing that these other problems cannot be computed by polynomial-size constant-depth circuits. By giving a more detailed classification of a larger collection of problems, our results contribute toward the theory of size-depth complexity for circuits. It should be expected that constant-depth reducibility will refine and expose more structure in low-level complexity classes such as deterministic polynomial time and nondeterministic logarithmic space than do the commonly used polynomial-time and log-space reducibilities.

Another motivation is that circuits with unbounded fan-in can be viewed as a model for unbounded fan-in parallelism, where circuit depth corresponds to parallel time and circuit size corresponds to the number of processors in the parallel machine. This view was strengthened by a recent result of Stockmeyer and Vishkin [22] giving a correspondence between circuits and the concurrent-read concurrent-write parallel random-access machines of Goldschlager [10] and Shiloach and Vishkin [19], [23] (*CRCW PRAM* or, for short, *WRAM*) which have many random access machines operating in parallel and communicating via a shared common memory, with appropriate conventions for resolving writing conflicts into the common memory. Stockmeyer and Vishkin show that time and number of processors for WRAM's correspond respectively (and simultaneously) to depth and size for circuits, where the time-depth correspondence is to within a constant factor and the size-processors correspondence is to within a polynomial. (Technically, for the simulation of circuits by WRAM's, the WRAM is allowed to be nonuniform.) By this correspondence, our results can also be viewed as contributing toward the theory of time-processors complexity for (nonuniform) WRAM's.

Section 2 of the paper contains definitions, including definitions of our constant-depth reducibilities. In §3, we establish several reductions and equivalences among problems using these reducibilities. We also show certain problems to be complete, with respect to these reducibilities, in the complexity classes deterministic logarithmic space, nondeterministic logarithmic space, and deterministic polynomial time. In §4 we give upper bounds on the size-depth circuit complexity of symmetric functions and graph transitive closure that improve the obvious upper bounds.

2. Definitions. A *Boolean function* is a function of the form $f:\{0,1\}^n \to \{0,1\}^m$ where 0 and 1 denote Boolean values *false* and *true*, respectively; n is the number of *variables* or *inputs*, and m is the number of *outputs*. A *problem* is an infinite sequence of Boolean functions $\{f_n \mid n \geq 1\}$ such that f_n has n variables. For some of the specific problems discussed herein, we define f_n only for certain values of n; in this case we assume that any f_n not defined explicitly is identically zero.

A *circuit* is an acyclic directed graph. Each node of the graph is labeled as either an input node, an AND-gate or an OR-gate. Input nodes must have fan-in (i.e., in-degree) zero. In addition, certain nodes are designated as output nodes.

An assignment of Boolean values to the input nodes extends, in the obvious way, to Boolean values associated with all nodes; an AND-gate (resp., OR-gate) with fan-in zero is assigned value 1 (resp., 0). The *size* of a circuit is the number of edges (i.e., wires). The *depth* is the length of a longest path from some input to some output. The circuit C *computes* the function $f:\{0,1\}^n \to \{0,1\}^m$ if C has $2n$ input nodes and m output nodes and there is a 1-1 correspondence between input nodes and $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ ($\bar{x}_i$ denotes the negation of x_i) and a 1-1 correspondence between output nodes and $y_1, \ldots, y_m$ such that, for all $(x_1, \ldots, x_n) \in \{0,1\}^n$, the assignment $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ to the input nodes extends to the assignment $y_1, \ldots, y_m$ to the output nodes where $f(x_1, \ldots, x_n) = (y_1, \ldots, y_m)$. (When input variables and their negations are available as inputs, it is well known that NOT-gates can be eliminated from circuits without increasing depth and at most doubling size, so we do not need NOT-gates in our formal definition of circuits. However, for convenience we sometimes use negations in describing circuits.)

DEFINITION. Let $S(n)$ and $D(n)$ be functions from positive integers to positive reals. The problem $F = \{f_n\}$ is in the complexity class SIZE-DEPTH$(S(n), D(n))$ if, for every $n \geq 1$, f_n is computed by a circuit with both size $\leq S(n)$ and depth $\leq D(n)$. Also define

$$\text{SIZE-DEPTH(poly, constant)} = \bigcup_{c,d,k \geq 0} \text{SIZE-DEPTH}(cn^k, d).$$

Similarly, SIZE-DEPTH(poly, $D(n)$) is the union of SIZE-DEPTH$(cn^k, D(n))$ over all constants c and k, and SIZE-DEPTH$(S(n)$, constant) is the union of SIZE-DEPTH$(S(n), d)$ over all constants d.

A *fan-in 2 circuit* is a circuit such that all AND-gates and OR-gates have fan-in 2.

Our first reducibility is the projection reducibility studied by Skyum and Valiant [20]. A problem $F = \{f_n\}$ is *projection reducible* to a problem $G = \{g_n\}$ ($F \leq_{\text{proj}} G$) if there is a function $p(n)$ bounded above by a polynomial in n, and for each $f_n \in F$ a mapping

$$\sigma_n: \{y_1, \ldots, y_{p(n)}\} \to \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, 0, 1\}$$

such that $f_n(x_1, \ldots, x_n) = g_{p(n)}(\sigma_n(y_1), \ldots, \sigma_n(y_{p(n)}))$. We also use a weaker reducibility, *constant-depth truth-table reducibility*: $F \leq_{\text{cd-tt}} G$ if there is a polynomial $p(n)$ and a constant c such that each f_n is computed by a circuit of depth $\leq c$ and size $\leq p(n)$ containing "black boxes" which compute members g_j of G or their negations $\bar{g}_j$ with $j \leq p(n)$, where the size and depth of black boxes are counted as unity, and such that there is no path in the circuit from an output of one black box to an input of another black box.

We write $F \equiv_{\text{x}} G$ if both $F \leq_{\text{x}} G$ and $G \leq_{\text{x}} F$.

The following proposition is obvious.

PROPOSITION 2.1. (1) $\leq_{\text{proj}}$ *and* $\leq_{\text{cd-tt}}$ *are transitive relations.*

(2) $F \leq_{\text{proj}} G \Rightarrow F \leq_{\text{cd-tt}} G$.

(3) *If* $F \leq_{\text{proj}} G$ *or* $F \leq_{\text{cd-tt}} G$, *and* $G \in$ SIZE-DEPTH$(S(n), D(n))$ *where* S *and* D *are monotone nondecreasing, then*

$$F \in \text{SIZE-DEPTH}(p(n) \cdot S(p(n)), c \cdot D(p(n)))$$

for some polynomial p *and constant* c. *In particular,*

$$G \in \text{SIZE-DEPTH(poly, constant)} \Rightarrow F \in \text{SIZE-DEPTH(poly, constant)}.$$

(We are interested primarily in the case $D(n) = O((\log n)^k)$ for some constant k, and $c \cdot D(p(n)) = O((\log n)^k)$ in this case, justifying the name "constant depth reducibility".)

We use certain complexity classes defined by time or space bounded Turing machines (see, for example, [9], [12]). Let $\mathscr{L}$ (resp., $\mathscr{NL}$) be the class of languages $L \subseteq \{0,1\}^*$ which are accepted by deterministic (resp., nondeterministic) Turing machines with a read-only input tape and a $\log n$ bounded worktape. A language $L \subseteq \{0,1\}^*$ is in the class NONUNIF-$\mathscr{L}$ (NONUNIF-$\mathscr{NL}$) if there is a polynomial $p(n)$ and a $\log n$ space-bounded deterministic (nondeterministic) Turing machine M, such that for every n there is a binary word α_n with $|\alpha_n| \leq p(n)$, such that for all x with $|x| = n$, M accepts $x\#\alpha_n$ iff $x \in L$; α_n is called the *advice* (for inputs of length n). Let $\mathscr{P}$ be the class of languages $L \subseteq \{0,1\}^*$ accepted by polynomial time-bounded deterministic Turing machines. A problem F is a *one-output problem* if every $f_n \in F$ has one output. The complexity classes $\mathscr{L}$, $\mathscr{NL}$, NONUNIF-$\mathscr{L}$, etc. can be viewed as classes of one-output problems by making the obvious correspondence to binary languages, namely, $x \in L$ iff $f_{|x|}(x) = 1$. If $\leq$ is a reducibility, F is a one-output problem, and $\mathscr{C}$ is a class of binary languages, then $\mathscr{C} \leq F$ if $G \leq F$ for all $G \in \mathscr{C}$; F is *$\leq$-complete in $\mathscr{C}$* if both $\mathscr{C} \leq F$ and $F \in \mathscr{C}$.

For several of the specific problems considered, inputs are natural numbers or graphs. Natural numbers are sometimes represented in unary notation and sometimes in binary. The *m-bit unary representation* of k $(0 \leq k \leq m)$ is the binary word 1^k0^{m-k}. Graphs are represented by adjacency matrices. An undirected graph with m vertices $v_1, \ldots, v_m$ is represented by $m(m-1)/2$ Boolean variables a_{ij} for $1 \leq i, j \leq m$ and $i < j$ such that $a_{ij} = 1$ iff there is an edge between v_i and v_j. A directed graph is represented by m^2 variables a_{ij} for $1 \leq i, j \leq m$ such that $a_{ij} = 1$ iff there is a directed edge from v_i to v_j. An undirected bipartite graph with $2m$ vertices $u_1, \ldots, u_m, v_1, \ldots, v_m$ is represented by m^2 variables a_{ij} for $1 \leq i, j \leq m$ such that $a_{ij} = 1$ iff there is an edge between u_i and v_j. In describing graph constructions we let $\{u,v\}$ denote the undirected edge between vertices u and v and we let (u,v) denote the directed edge from u to v.

Define the function

$$\lg(n) = \lceil \log_2(n+1) \rceil;$$

note that $\lg(n)$ is the length of the binary representation of n.

3. Constant depth reducibility among problems. Our results are summarized in Fig. 1; see the Appendix for precise definitions of the various problems.

THEOREM 3.1. *All the problems enclosed as a group in Fig.* 1 *are in the same* $\equiv_{\text{cd-tt}}$ *equivalence class. Groups with* $\equiv_{\text{proj}}$ *written to the left are in the same* $\equiv_{\text{proj}}$ *equivalence class. A problem in a group $\mathscr{G}$ is reducible to all problems in the group above $\mathscr{G}$ via the indicated reducibility. Problems in the lowest group in Fig.* 1 *are in the class* SIZE-DEPTH(poly, constant).

(We note that Furst, Saxe and Sipser [8] previously showed that PARITY is $\leq_{\text{cd-tt}}$ to THRESHOLD, MULTIPLICATION, and UNDIR-ST-CONNECTIVITY.) Problems that are $\equiv_{\text{proj}}$ are, in a fairly strong sense, the "same" problem. Since Furst, Saxe and Sipser [8] show that PARITY is not in the class SIZE-DEPTH(poly, constant), we know that this class lies strictly below the class containing PARITY in the $\leq_{\text{cd-tt}}$ ordering. We do not know that any of the

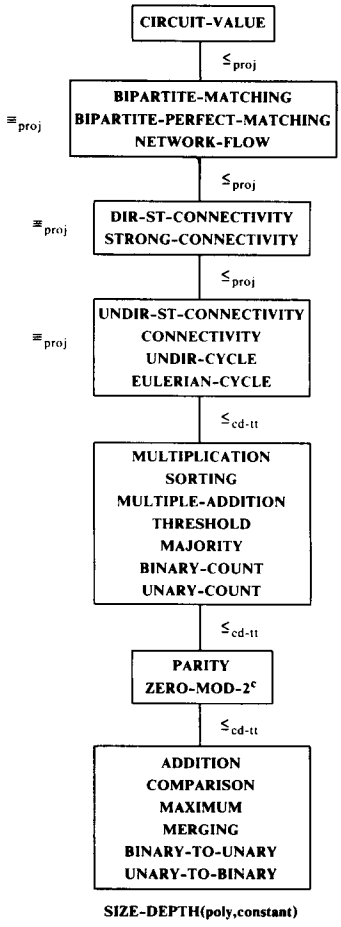

FIG. 1.

other relationships in Fig. 1 are strict in the $\leq_{cd\text{-}tt}$ ordering. Several of the reductions in Fig. 1 are not surprising. However, we can single out

$$\text{UNDIR-ST-CONNECTIVITY} \equiv_{proj} \text{CONNECTIVITY}$$

$$\text{MULTIPLICATION} \equiv_{cd\text{-}tt} \text{SORTING}$$

as being nonobvious.

Shortly we prove Theorem 3.1 by giving a series of reductions. We first prove the following completeness results; these results immediately yield most of the projection-equivalences for graph connectivity problems.

THEOREM 3.2.

(1) UNDIR-ST-CONNECTIVITY, CONNECTIVITY, *and* UNDIR-CYCLE *are* $\leq_{proj}$*-complete in* NONUNIF-$\mathscr{L}$.

(2) UNDIR-CYCLE *is* $\leq_{proj}$*-complete in* $\mathscr{L}$.

(3) DIR-ST-CONNECTIVITY *and* STRONG-CONNECTIVITY *are* $\leq_{proj}$*-complete in* NONUNIF-$\mathscr{NL}$ *and in* $\mathscr{NL}$.

(4) CIRCUIT-VALUE *is* $\leq_{proj}$*-complete in* $\mathscr{P}$.

Proof. The reductions are very similar to the known log-space reductions [11], [13], [15], [17].

(1) By a result of Aleliunas et al. [1], it easily follows that all three problems mentioned in part (1) belong to NONUNIF-$\mathscr{L}$; for inputs of length $n = m(m-1)/2$ corresponding to adjacency matrices for undirected graphs with m vertices, the advice α_n encodes a universal traversal sequence for such graphs. For the other direction we first show that

$$\text{NONUNIF-}\mathscr{L} \leq_{\text{proj}} \text{UNDIR-ST-CONNECTIVITY}.$$

Let M be a deterministic $\lg(n)$ space-bounded Turing machine. Fix an input length n, let α_n be the advice, and let $n' = |\alpha_n|$. The *configurations* of M are 4-tuples (q, i, β, j) where q indicates the state of the finite control, i with $0 \leq i \leq n+n'+2$ is the position of the input head in the string $\mathrm{¢}x\#\alpha_n\$$ (where ¢ and \$ are left and right endmarkers), the word β with $|\beta| = \lg(n)$ is the content of the worktape, and j is the position of the head on the worktape. The machine has a unique initial configuration $c_0 = (q_0, 0, \#^{\lg(n)}, 1)$ where q_0 is the unique initial state and # is the blank tape symbol. Assume that M has a unique accepting configuration $c_a = (q_a, 0, \#^{\lg(n)}, 1)$ where q_a is the unique accepting state. Modify M so that it never halts, and that if configuration c_a is entered then M cycles forever in c_a. Let p be the number of configurations of M; p is bounded above by a polynomial in n. The number p can also be taken as an upper bound on the time for M to accept; i.e., if M accepts x of length n then M will be in configuration c_a at step p.

For each input x of length n we describe a graph $G(x)$ such that the adjacency matrix of $G(x)$ is obtained from x by a projection reduction. The vertices of $G(x)$ are all pairs (c, t) where c is a configuration of M and $0 \leq t \leq p$. Consider a vertex (c, t) with $c = (q, i, \beta, j)$ and $0 \leq t < p$. If $i = 0$ or $i > n$, then the configuration c' reached in one move of M is determined and $G(x)$ will have the edge $\{(c, t), (c', t+1)\}$. If $1 \leq i \leq n$, there will be two configurations c' and c'' such that c' (c'') is the configuration reached in one move if $x_i = 0$ ($x_i = 1$). Thus, $\bar{x}_i$ is the entry in the adjacency matrix corresponding to the edge $\{(c, t), (c', t+1)\}$, and x_i is the entry corresponding to the edge $\{(c, t), (c'', t+1)\}$. If $c' = c''$ then the entry corresponding to edge $\{(c, t), (c', t+1)\}$ is 1. Let $m = p(p+1)$ be the number of vertices of $G(x)$. Number the vertices so that $v_1 = (c_0, 0)$ and $v_m = (c_a, p)$. Since M is deterministic, for each fixed x and each vertex v of the form $(\cdot, p)$, the connected component of $G(x)$ containing v is a tree which can be rooted at v such that the sons of $(\cdot, t)$ in the tree are all of the form $(\cdot, t-1)$ for all t. Therefore, M accepts x iff there is a path from $v_1 = (c_0, 0)$ to $v_m = (c_a, p)$.

For CONNECTIVITY, we form $G'(x)$ from $G(x)$ by adding the edges of a spanning tree among the vertices

$$\{ (c, p) \mid c \neq c_a \} \cup \{(c_0, 0)\}.$$

Recalling that M never halts, it is easy to see that $G'(x)$ is connected iff there is a path from $(c_0, 0)$ to (c_a, p). For UNDIR-CYCLE, we add to $G(x)$ the edge between $(c_0, 0)$ and (c_a, p).

(2) Since Hong [11] shows that UNDIR-CYCLE $\in \mathscr{L}$, the result is immediate from part (1).

(3) Obviously DIR-ST-CONNECTIVITY and STRONG-CONNECTIVITY belong to $\mathscr{NL}$. The proof that

$$\text{NONUNIF-}\mathcal{NL} \leq_{\text{proj}} \text{DIR-ST-CONNECTIVITY}$$

is very similar to the undirected case in (1). For example, if from configuration $c = (q, i, \beta, j)$ with $x_i = 0$ ($x_i = 1$) M can reach any configuration in the set C_0 (C_1) in one move, and if $B = C_0 \cap C_1$, then the entry of the adjacency matrix corresponding to the directed edge $((c,t),(c',t+1))$ is $\bar{x}_i$ if $c' \in C_0 - B$, x_i if $c' \in C_1 - B$, 1 if $c' \in B$, or 0 otherwise. So there is a directed path in $G(x)$ from $v_1 = (c_0, 0)$ to $v_m = (c_a, p)$ iff M accepts x.

For STRONG-CONNECTIVITY, form $G'(x)$ from $G(x)$ by adding the directed edges

$$\{ (v_m, u), (u, v_1) \mid u \neq v_1 \text{ and } u \neq v_m \}.$$

Then $G'(x)$ is strongly connected iff there is a directed path from v_1 to v_m in $G(x)$.

(4) As Ladner [15] points out, it is well known that for any deterministic Turing machine M there is a polynomial-size fan-in 2 circuit which takes as input a (binary representation of a) configuration of M and outputs the next configuration after one step. If M is $p(n)$ time-bounded, one places $p(n)$ copies of this circuit in series. By choosing the binary representation of tape symbols properly, the bits of the binary representation of the initial configuration are either constants or depend directly on the bits of the input x to M. Therefore, this is a projection reduction. □

The following useful lemma is proved by computing f from its disjunctive normal form.

LEMMA 3.3. *If f is a Boolean function with n inputs and m outputs, then f is computed by a circuit of size $\leq (m+n)\cdot 2^n$ and depth 2.*

Proof of Theorem 3.1. We first show that the problems in the lowest group in Fig. 1 are in SIZE-DEPTH(poly, constant).

ADDITION

The well known carry look-ahead method for the addition of m-bit numbers can be implemented as a circuit of size $O(m^3)$ and constant depth (see, e.g., the proof of Theorem 1 in [22]). This size bound is greatly improved in [4].

COMPARISON

Given two m-bit binary numbers $y = y_m \ldots y_2 y_1$ and $z = z_m \ldots z_2 z_1$, $y < z$ iff there is an i such that $y_i = 0$, $z_i = 1$, and $y_j = z_j$ for all $j > i$. This can be computed by a circuit of size $O(m^2)$ and depth $O(1)$.

MAXIMUM

Given m m-bit binary numbers $a_1, \ldots, a_m$, let $c_{ij} = 1$ if $a_i \geq a_j$, or 0 otherwise. Using COMPARISON, all the c_{ij} can be computed in size $O(m^4)$ and depth $O(1)$. Letting d_i be the AND of c_{ij} over all $j \neq i$, a_i is the maximum iff $d_i = 1$. The maximum number can be computed as

$$\bigvee_{1 \leq i \leq m} a_i \wedge d_i$$

where the $\wedge$ is computed bitwise over the bits of a_i.

MERGING

Let $a_1, ..., a_m$ and $b_1, ..., b_m$ be the lists of m-bit binary numbers, each list sorted in nondecreasing order. (Using COMPARISON, a polynomial-size constant-depth circuit can check that the lists are indeed sorted and can set all bits of the output to zero if not.) Let $c_{ij} = 1$ iff $b_i < a_j$. For each j, $c_{1j}c_{2j}...c_{mj}$ is a unary representation of the number of b's less than a_j. Let c_j be this unary representation with j 1's appended on the left and $m-j$ 0's appended on the right, so c_j is a $2m$-bit unary representation of a_j's position in the merged list. Similarly, let $d_{ij} = 1$ iff $a_i \leq b_j$. Appending j 1's to the left of $d_{1j}d_{2j}...d_{mj}$ and $m-j$ 0's to the right gives b_j's position in the merged list as a unary number d_j. For each k with $1 \leq k \leq 2m$, let $EQ_k(z)$ be a one-output circuit whose output is 1 iff z is a unary representation of k; letting $z = z_1z_2...z_{2m}$ and $z_{2m+1} = 0$, $EQ_k(z) = z_k \wedge \bar{z}_{k+1}$. Finally, the k^{th} number in the merged list is

$$\bigvee_{1 \leq j \leq 2m} (EQ_k(c_j) \wedge a_j) \vee \bigvee_{1 \leq j \leq 2m} (EQ_k(d_j) \wedge b_j).$$

Remark. Shiloach and Vishkin [19] show that MAXIMUM and MERGING can be computed by a WRAM in constant time with a polynomial number of processors. Therefore, an alternate proof that MAXIMUM and MERGING are in SIZE-DEPTH(poly, constant) follows from this result and the Stockmeyer-Vishkin [22] simulation of WRAM's by circuits mentioned in the Introduction.

UNARY-TO-BINARY

Let $x = x_1...x_n$ be the given unary representation. (A polynomial-size constant-depth circuit can check whether $x_1...x_n \in 1^*0^*$ and set the output to zero if not.) Let $x_0 = 1$, $x_{n+1} = 0$ and $d_i = x_i \wedge \bar{x}_{i+1}$ for $0 \leq i \leq n$, so $d_i = 1$ iff x is the unary representation of i. For $1 \leq j \leq \lg(n)$, the j^{th} bit of the binary representation is the OR of d_i over all i such that the j^{th} bit of i is 1.

BINARY-TO-UNARY

Since the output depends only on the first $\lg(n)$ inputs, the polynomial-size constant-depth circuit is immediate from Lemma 3.3.

The remainder of the proof of Theorem 3.1 is a series of reductions.

PARITY $\leq_{\text{proj}}$ ZERO-MOD-2^c

PARITY on n variables $x_1, ..., x_n$ is reduced to ZERO-MOD-2^c on $2^{c-1}n$ variables y_{ij} for $1 \leq i \leq n$ and $1 \leq j \leq 2^{c-1}$. The projection reduction is $\sigma_n(y_{ij}) = x_i$ for all j.

ZERO-MOD-2^c $\leq_{\text{cd-tt}}$ PARITY

We show that ZERO-MOD-2^c $\leq_{\text{cd-tt}}$ ZERO-MOD-2^{c-1} for all $c \geq 2$. This suffices since c is constant and $\leq_{\text{cd-tt}}$ is transitive. Let $x_1, ..., x_n$ be the input bits and let $s = \Sigma x_i$. Compute $y_{ij} = x_i \wedge x_j$ for all i and j with $i < j$. Let $t = \Sigma y_{ij}$ and note that $t = s(s-1)/2$. It is easy to verify that $s \equiv 0 \pmod{2^c}$ iff both $s \equiv 0 \pmod{2^{c-1}}$ and $t \equiv 0 \pmod{2^{c-1}}$.

PARITY $\leq_{\text{proj}}$ MULTIPLICATION

This reduction is given in [8]. Given $x_1, ..., x_n$, let $k = \lg(n)$ and form the two binary numbers

$$A = \sum_{i=1}^{n} x_i 2^{k(i-1)}, \quad B = \sum_{i=1}^{n} 2^{k(i-1)}.$$

Writing $AB = \Sigma c_i 2^{k(i-1)}$, where the c_i are k-bit numbers, the parity of Σx_i is the low order bit of c_n.

The next seven reductions form a cycle showing $\equiv_{\text{cd-tt}}$ for the seven problems in the group containing MULTIPLICATION in Fig. 1.

MAJORITY $\leq_{\text{cd-tt}}$ MULTIPLICATION

The first stage of this reduction is identical to PARITY $\leq_{\text{proj}}$ MULTIPLICATION described above. The binary representation of Σx_i is c_n. These k bits are then compared with the k-bit binary representation of $\lceil n/2 \rceil$.

MULTIPLICATION $\leq_{\text{cd-tt}}$ MULTIPLE-ADDITION

Multiplication of m-bit numbers y and $z = \Sigma z_i 2^i$ is reduced to addition of m $2m$-bit numbers $a_0, \ldots, a_{m-1}$. For each i, $a_i = z_i 2^i y$, so each bit of each a_i is the AND of a bit of y and a bit of z.

MULTIPLE-ADDITION $\leq_{\text{cd-tt}}$ BINARY-COUNT

Let $A_1, \ldots, A_m$ be the m m-bit binary numbers to be added, and for $1 \leq i \leq m$ let

$$A_i = \sum_{j=0}^{m-1} a_{ij} 2^j \quad \text{where } a_{ij} \in \{0,1\}.$$

Let $L = m + \lg(m)$ and let $\ell = \lg(m)$. Note that L bits are sufficient for the sum $S = \Sigma A_i$. We describe the reduction as a sequence of stages.

The first stage. For $0 \leq j < m$, let b_j be the ℓ-bit result of applying BINARY-COUNT with inputs $a_{1j}, a_{2j}, \ldots, a_{mj}$. Note that $S = \Sigma b_j 2^j$. Since each b_j has only ℓ bits, the numbers $b_j 2^j$ can be "packed" into ℓ L-bit numbers $B_1, \ldots, B_\ell$ whose sum is S; that is for each i with $1 \leq i \leq \ell$ let

$$B_i = \sum_{k \geq 0} b_{i-1+k\ell} 2^{i-1+k\ell}.$$

The second stage. Let $\ell(2) = \lg(\ell)$ $(= \lg(\lg(m)))$. Similarly, by applying BINARY-COUNT to each bit position in $B_1, \ldots, B_\ell$ and then packing the $\ell(2)$-bit results into $\ell(2)$ L-bit numbers $C_1, \ldots, C_{\ell(2)}$, we have reduced the original problem to the problem of adding $\ell(2)$ L-bit numbers. Since each use of BINARY-COUNT in the second stage has only $\lg(m)$ inputs, each application of BINARY-COUNT is implemented directly by a circuit of polynomial size and constant depth (see Lemma 3.3) rather than by a "black box" as in the first stage.

The remaining stages. For $i \geq 3$, let $\ell(i) = \lg(\ell(i-1))$. By the same method of counting followed by packing, the i^{th} stage produces $\ell(i)$ L-bit numbers whose sum is S. Let i^* be the smallest i such that $\ell(i) = 2$. The i^* stage produces two L-bit numbers, say Y and Z, which can be added directly by a circuit of polynomial size and constant depth to obtain S. If implemented as described, this would give a circuit of depth $> i^*$ (not constant). Actually, only the first two stages are implemented as described. For $i \geq 3$, note that each bit of each L-bit number produced at the i^{th} stage depends on at most $\ell(i-1)$ bits of the numbers produced at the $(i-1)^{\text{th}}$ stage. Therefore, each bit of Y and Z depends on at most $t = \ell(2)\ell(3)\ldots\ell(i^*-1)$ bits of the numbers $C_1, \ldots, C_{\ell(2)}$ produced at the

second stage. Since $t = O(\lg n)$, Lemma 3.3 says that Y and Z can be computed from the C_i by a circuit of polynomial size and constant depth.

BINARY-COUNT $\leq_{\text{cd-tt}}$ SORTING

Let $x_1, \ldots, x_n$ be the input bits. Let a_i be an n-bit number whose highest order bit is x_i (the lower order bits of the a_i can be chosen to make the a_i distinct if desired). The highest order bits of the a_i in sorted order give an n-bit unary representation of Σx_i, from which the binary representation of Σx_i can be computed in polynomial size and constant depth.

SORTING $\leq_{\text{cd-tt}}$ UNARY-COUNT

Let $a_1, \ldots, a_m$ be the numbers to be sorted. Let $c_{ij} = 1$ iff either $a_i < a_j$ or $(a_i = a_j$ and $i \leq j)$. For each fixed j, applying UNARY COUNT with inputs c_{ij} for all i gives a unary representation of a_j's position in the sorted list. The sorted list is then computed as in the proof that MERGING is in SIZE-DEPTH(poly, constant).

UNARY-COUNT $\leq_{\text{cd-tt}}$ THRESHOLD

Given $x_1, \ldots, x_n$, the i^{th} bit of the unary representation of Σx_i is the result of applying THRESHOLD to $x_1, \ldots, x_n$ with threshold i.

THRESHOLD $\leq_{\text{cd-tt}}$ MAJORITY

Let $x_1, \ldots, x_m$ be the input bits to the threshold function and let $k_1 \ldots k_m$ be the m-bit unary representation of the threshold k. Let $y_i = x_i$ and $y_{m+i} = \bar{k}_i$ for $1 \leq i \leq m$. Now

$$\sum_{i=1}^{m} x_i \geq k \quad \text{iff} \quad \sum_{i=1}^{2m} y_i \geq m.$$

This completes the cycle for the $\equiv_{\text{cd-tt}}$-class containing MULTIPLICATION.

MAJORITY $\leq_{\text{proj}}$ CONNECTIVITY

Since MAJORITY $\in \mathscr{L}$, this is immediate from Theorem 3.2(1).

UNDIR-ST-CONNECTIVITY $\equiv_{\text{proj}}$ CONNECTIVITY

UNDIR-ST-CONNECTIVITY $\equiv_{\text{proj}}$ UNDIR-CYCLE

These are immediate from Theorem 3.2(1).

EULERIAN CYCLE $\leq_{\text{proj}}$ CONNECTIVITY

An undirected graph G has an Eulerian cycle iff G is connected except for isolated vertices and every vertex of G has even degree [2]. Since [1] shows that the connectivity test is in NONUNIF-$\mathscr{L}$, it follows that EULERIAN-CYCLE is in NONUNIF-$\mathscr{L}$, so the reduction follows from Theorem 3.2(1).

CONNECTIVITY $\leq_{\text{proj}}$ EULERIAN-CYCLE

Given an undirected graph G, we describe an undirected graph G' such that G is connected iff G' has an Eulerian cycle. If G has vertices $V = \{v_1, \ldots, v_m\}$, then G' has vertices

$$V \cup \{ u_{ij} \mid 1 \leq i < j \leq m \} \cup \{ y_i, z_i \mid 1 \leq i \leq m \}.$$

The edges $\{v_i, y_i\}$, $\{v_i, z_i\}$ and $\{y_i, z_i\}$ are in G' for all i. The edges $\{v_i, u_{ij}\}$, $\{u_{ij}, v_j\}$ and $\{v_i, v_j\}$ are present in G' iff the edge $\{v_i, v_j\}$ is present in G. Note that every vertex of G' has even degree, and G' is connected (except for isolated vertices) iff G is connected.

UNDIR-ST-CONNECTIVITY $\leq_{\text{proj}}$ DIR-ST-CONNECTIVITY

DIR-ST-CONNECTIVITY $\equiv_{\text{proj}}$ STRONG-CONNECTIVITY

These are immediate from Theorem 3.2(3).

DIR-ST-CONNECTIVITY $\leq_{\text{proj}}$ NETWORK-FLOW

There is a directed path from vertex v_1 to vertex v_m in G iff there is a flow of ≥ 1 from v_1 to v_m when all edges of G have unit capacity.

The next three reductions form a cycle.

BIPARTITE-PERFECT-MATCHING $\leq_{\text{proj}}$ BIPARTITE-MATCHING

This is trivial.

BIPARTITE-MATCHING $\leq_{\text{proj}}$ NETWORK-FLOW

This reduction is given in [6, Thm. 6.11]. Let G be the given bipartite graph with vertices $u_1, \ldots, u_m, v_1, \ldots, v_m$. Form a directed flow network G' with vertices $s, t, u_1, \ldots, u_m, v_1, \ldots, v_m$. The directed edges (s, u_i) and (v_i, t) in G' have capacity 1 for all $1 \leq i \leq m$. The directed edge (u_i, v_j) in G' has capacity 1 (resp., 0) if the edge $\{u_i, v_j\}$ is present (resp., not present) in G. All other edges of G' have capacity 0. There is a flow of f from s to t iff G has a matching of size f.

NETWORK-FLOW $\leq_{\text{proj}}$ BIPARTITE-PERFECT-MATCHING

Many of the details of this reduction were pointed out to us by N. Pippenger; the reduction was observed independently by T. Feather. A few additional details needed to make the reduction a projection are due to the authors. Let G be the given directed flow network with vertices $v_1, \ldots, v_m$. Assume for the moment that the edge capacities $c(i, j)$ and the flow f are valid unary representations. It is convenient to describe the reduction as a composition of two projection reductions. In the first reduction, we transform G to a directed graph G' with $2m^3 - m^2$ vertices. For each vertex v_i of G, G' has m^2 "copies" of v_i, namely v_{ip} for $1 \leq p \leq m^2$. For all $1 \leq p \leq m^2$, the vertices v_{1p} are called *sources* of G' and the vertices v_{mp} are called *targets* of G'. For each pair (i, j) with $1 \leq i, j \leq m$ and $i \neq j$, G' also has vertices e_{ijk} for $1 \leq k \leq m$. For each i, j, k, p and q with $1 \leq i, j, k \leq m$ and $1 \leq p, q \leq m^2$, the directed edges (v_{ip}, e_{ijk}) and (e_{ijk}, v_{jq}) are present in G' iff $k \leq c(i, j)$. Since $c(i, j)$ is represented in unary, this is a projection reduction; that is, the entries of the adjacency matrix corresponding to the edges (v_{ip}, e_{ijk}) and (e_{ijk}, v_{jq}) are simply the k^{th} bit of $c(i, j)$. There is a flow of f from v_1 to v_m in G iff

(*) There are f vertex disjoint paths in G', each path being from some source to some target, with the sources and targets of the paths also being disjoint.

In the second step we transform G' to an undirected bipartite graph G''. Replace each source u of G' by a vertex u_b. Replace each target u of G' by a vertex u_g. Replace each other vertex u of G' by the pair of vertices u_b and u_g with an edge between u_b and u_g. The edge $\{u_b, w_g\}$ is present in G'' iff the directed edge (u, w) is present in G'. Let $f_1 f_2 \ldots f_r$ be the unary representation of the flow f, where $r = m^2$. If u_b is the replacement for the source v_{1p} and w_g is the replacement for the target v_{mp}, then the edge $\{u_b, w_g\}$ is present in G'' iff $f_p = 0$. Thus, for each p with $1 \leq p \leq f$, the replacement u_b for v_{1p} cannot be matched with the replacement w_g for v_{mp}, but for $f < p \leq m^2$, u_b can be matched with w_g.

It is not difficult to verify that (*) holds in G' iff there is a perfect matching in G''. The correspondence between vertex disjoint directed paths in G' and perfect matchings in G'' is as follows: If (u, w) is not a source-target pair, the edge (u, w) of G' is used in one of the directed paths iff the edge $\{u_b, w_g\}$ of G'' is used in the matching; if the vertex u of G' is neither a source nor a target, then u does not appear on any of the directed paths iff the edge $\{u_b, u_g\}$ is used in the matching. Details are left to the reader.

Finally, for each word $c(i,j)$ and f we add a new component to G'' such that the added components all have perfect matchings iff all $c(i,j)$ and f are valid unary representations. In general, let $w = w_1 \ldots w_m$ be a binary word. We describe a bipartite graph $H(w)$ with vertices $a_1, \ldots, a_{m+1}$, $b_1, \ldots, b_{m+1}$ such that $H(w)$ is obtained from w by a projection reduction and $H(w)$ has a perfect matching iff $w \in 1^*0^*$. The edge $\{a_{m+1}, b_{m+1}\}$ and the edges $\{a_i, b_1\}$ are in $H(w)$ for all i with $2 \le i \le m+1$. For $1 \le i \le m$, the edge $\{a_i, b_{i+1}\}$ is in $H(w)$ iff $w_i = 1$, and the edge $\{a_i, b_i\}$ is in $H(w)$ iff $w_i = 0$. If $w \notin 1^*0^*$ then $w_i = 0$ and $w_{i+1} = 1$ for some i; this means that the vertex b_{i+1} has no edges adjacent to it, so $H(w)$ cannot have a perfect matching. Conversely, if $w = 1^p0^q$ then the perfect matching is $\{a_i, b_{i+1}\}$ for $1 \le i \le p$, $\{a_{p+1}, b_1\}$, and $\{a_i, b_i\}$ for $p+2 \le i \le m+1$.

BIPARTITE-MATCHING $\le_{\text{proj}}$ CIRCUIT-VALUE

Since BIPARTITE-MATCHING $\in \mathscr{P}$, this is immediate from Theorem 3.2(4).

This completes the proof of Theorem 3.1.

4. Upper bounds. It is not hard to show that BINARY-COUNT is in SIZE-DEPTH(poly, $O(\log n)$) and that DIR-ST-CONNECTIVITY is in SIZE-DEPTH($O(2^n)$, constant). These upper bounds can be improved. Regarding DIR-ST-CONNECTIVITY, Schorr [18] gives a simulation of nondeterministic Turing machines by circuits; the following Theorem 4.1 is implicit in the simulation.

THEOREM 4.1 (Schorr). *There is a constant b such that for any integer $k > 0$ there is a constant c such that*

$$\text{DIR-ST-CONNECTIVITY} \in \text{SIZE-DEPTH}(2^{cn^{1/k}}, bk).$$

Proof. We sketch the proof for completeness. Let G be the given directed graph with m vertices. Let $n = m^2$ be the number of bits in G's adjacency matrix. A circuit of constant depth b and size exponential in $n^{1/k}$ can compute the adjacency matrix of a new m-vertex directed graph G' such that, for each pair of vertices v_i and v_j, the edge (v_i, v_j) is present in G' iff there is a directed path of length $\le \lceil m^{1/k} \rceil$ from v_i to v_j in G. Fix an i and j, and let $r = \lceil m^{1/k} \rceil$. A *possible path* is any sequence $w_0, \ldots, w_r$ of $r+1$ vertices such that $w_0 = v_i$ and $w_r = v_j$. For each possible path $\rho = (w_0, \ldots, w_r)$, let $\tau(\rho)$ be the AND of the entries in G's adjacency matrix corresponding to the edges (w_t, w_{t+1}) for $0 \le t < r$. Assume that all (i,i)-entries have been set to 1. Now the (i,j)-entry in the adjacency matrix of G' is the OR of $\tau(\rho)$ over all possible paths ρ from v_i to v_j. By placing k copies of this adjacency matrix transformation in series, the $(1,m)$-entry in the final matrix is 1 iff there is a path (necessarily of length $\le m$) from v_1 to v_m in G. □

THEOREM 4.2. *If* F = BINARY-COUNT *or* F *is a family of one-output symmetric Boolean functions, then for any* $\varepsilon > 0$,

$$F \in \text{SIZE-DEPTH}(O(n \cdot 2^{(\log n)^{\varepsilon}}), O(\log n/\varepsilon \cdot \operatorname{loglog} n)).$$

Define 2-SIZE-DEPTH($S(n), D(n)$) to be the class of problems F such that each $f_n \in F$ is computed by a fan-in 2 circuit of size $\leq S(n)$ and depth $\leq D(n)$. Since Muller and Preparata [16] show that BINARY-COUNT and all families of one-output symmetric Boolean functions are in 2-SIZE-DEPTH($O(n), O(\log n)$), Theorem 4.2 is immediate from the following:

THEOREM 4.3. *For any* $\varepsilon > 0$,

$$\text{2-SIZE-DEPTH}(S(n), D(n)) \subseteq \text{SIZE-DEPTH}(O(2^{(\log n)^{\varepsilon}} \cdot S(n)), O(D(n)/\varepsilon \cdot \operatorname{loglog} n)).$$

Proof. Any fan-in 2 circuit with one output and depth $\leq (\varepsilon/2) \cdot \operatorname{loglog} n$ can depend on at most $(\log n)^{\varepsilon/2}$ inputs, so by Lemma 3.3 it can be replaced by a circuit of depth 2 and size $O(2^{(\log n)^{\varepsilon}})$. Given a fan-in 2 circuit of depth D, partition its nodes into levels $L_1, L_2, \ldots$ such that level L_i has nodes whose depth is between $(i-1)(\varepsilon/2)\operatorname{loglog} n$ and $i(\varepsilon/2)\operatorname{loglog} n$. View each node in L_1 as the output node of a fan-in 2 circuit. By the observation above, each such circuit can be replaced by a circuit of size $O(2^{(\log n)^{\varepsilon}})$ and depth ≤ 2. Regard the output nodes of these new circuits as the "inputs" to level L_2. Apply this process to $L_2, L_3, \ldots$ in sequence, so each level is compressed to depth 2. □

By our reductions, the upper bound SIZE-DEPTH($2^{cn^{1/k}}$, constant) holds for any problem $F \leq_{\text{cd-tt}}$ DIR-ST-CONNECTIVITY, and the upper bound SIZE-DEPTH(poly, $O(\log n/\operatorname{loglog} n)$) holds for any problem $F \leq_{\text{cd-tt}}$ BINARY-COUNT (e.g., MULTIPLICATION and SORTING). Theorems 4.1 and 4.2 could serve as an avenue for proving that certain of the $\equiv_{\text{cd-tt}}$-classes in Fig. 1 are distinct. For example, if

$$\text{DIR-ST-CONNECTIVITY} \notin \text{SIZE-DEPTH}(\text{poly}, O(\log n/\operatorname{loglog} n))$$

then the class containing BINARY-COUNT and the class containing DIR-ST-CONNECTIVITY are different. The following result shows that if these two classes are equal then Savitch's theorem [17] can be improved in the nonuniform case.

THEOREM 4.4. *If* BINARY COUNT $\equiv_{\text{cd-tt}}$ DIR-ST-CONNECTIVITY, *then*

$$\text{NONUNIF-}\mathcal{NL} = \text{NONUNIF-}\mathcal{L}.$$

Proof. In the constant-depth circuit that performs the reduction DIR-ST-CONNECTIVITY $\leq_{\text{cd-tt}}$ BINARY-COUNT, replace each gate with fan-in $r > 2$ by a tree of depth $O(\log r)$; since r is bounded above by a polynomial in n, $\log r = O(\log n)$. Now replace each "black box" computing some member of BINARY-COUNT by a Muller-Preparata [16] fan-in 2 circuit of depth $O(\log n)$ and polynomial size. Thus

$$\text{DIR-ST-CONNECTIVITY} \in \text{2-SIZE-DEPTH}(\text{poly}, O(\log n)).$$

The conclusion NONUNIF-$\mathcal{NL}$ = NONUNIF-$\mathcal{L}$ now follows easily from two facts:

(1) NONUNIF-$\mathcal{NL} \leq_{\text{proj}}$ DIR-ST-CONNECTIVITY (Theorem 3.2(3));

(2) The circuit value problem for fan-in 2 circuits of polynomial size and

depth $D(n)$ ($\geq \log n$) can be solved by a deterministic Turing machine in space $O(D(n))$ (see [3, Lemma 1]).

Details are left to the reader. □

5. Conclusion. One direction for future work is to add other problems to the classification begun in Fig. 1. A more fundamental open question is whether any two of the groups in Fig. 1 can be collapsed into one $\equiv_{\text{cd-tt}}$-class, or conversely whether the $\leq_{\text{cd-tt}}$ relation between the two groups is strict. In proving the latter, a stronger result would be obtained by replacing $\leq_{\text{cd-tt}}$ with constant-depth Turing reducibility, $\leq_{\text{cd-T}}$, defined like $\leq_{\text{cd-tt}}$ except that now in the circuit that performs the reduction there can be directed paths from the output of one black box to the input of another black box. (We did not define $\leq_{\text{cd-T}}$ above since we needed at most the power of $\leq_{\text{cd-tt}}$ for our specific reductions.)

Another interesting area is to improve the known lower bounds on size-depth circuit complexity. Furst, Saxe and Sipser [8] show that unbounded fan-in circuits can possibly be used as a tool to separate the polynomial-time hierarchy from PSPACE by an oracle. However, the lower bound for PARITY proved in [8], that any constant depth circuit must have size at least $n^{c \cdot \log^* n}$ for some constant c, is not sufficient to imply the existence of such an oracle. Our reducibilities suggest that an improved lower bound might be easier for a problem, such as CONNECTIVITY, that lies above PARITY in the $\leq_{\text{cd-tt}}$ ordering.

Appendix. In several of the following problems we require the input to be of a particular form. For example, in MERGING, the two input lists must be sorted, and in BIPARTITE-MATCHING, NETWORK-FLOW, THRESHOLD and UNARY-TO-BINARY, certain parts of the input must be valid unary representations, i.e., words in 1^*0^*. If the input is not of the correct form, all bits of the output are defined to be zero.

ADDITION (MULTIPLICATION)
INPUT: Two m-bit binary numbers ($n = 2m$).
OUTPUT: Their sum (product).

BINARY-COUNT (UNARY-COUNT)
INPUT: n bits $x_1, \ldots, x_n$.
OUTPUT: $\lg(n)$-bit binary representation of their sum Σx_i (n-bit unary representation of Σx_i).

BINARY-TO-UNARY
INPUT: n bits $x_1, \ldots, x_n$.
OUTPUT: $2^{\lg(n)}$-bit unary representation of the number whose binary representation is $x_1 x_2 \ldots x_{\lg(n)}$.

BIPARTITE-MATCHING
INPUT: Adjacency matrix of a $2m$-vertex bipartite graph and an m-bit unary number k.
OUTPUT: Does the graph have a matching with $\geq k$ edges?

BIPARTITE-PERFECT-MATCHING
INPUT: Adjacency matrix of a $2m$-vertex bipartite graph.
OUTPUT: Does the graph have a perfect matching?

CIRCUIT-VALUE
INPUT: Binary representation of a fan-in 2 circuit (i.e., every gate has fan-in at most 2) with one output node, and an assignment of Boolean values to all input nodes (the details of the binary representation are not crucial).
OUTPUT: The Boolean value computed by the output node.

COMPARISON
INPUT: Two m-bit binary numbers z_1 and z_2 $(n = 2m)$.
OUTPUT: Is $z_1 > z_2$?

CONNECTIVITY (STRONG-CONNECTIVITY)
INPUT: Adjacency matrix of an undirected (directed) graph.
OUTPUT: Is the graph connected (strongly connected)?

DIR-ST-CONNECTIVITY (UNDIR-ST-CONNECTIVITY)
INPUT: Adjacency matrix of an m-vertex directed (undirected) graph.
OUTPUT: Is there a directed path (a path) from v_1 to v_m?

EULERIAN-CYCLE
INPUT: Adjacency matrix of an undirected graph.
OUTPUT: Is there a cycle that traverses every edge exactly once?

MAJORITY
INPUT: n bits $x_1, \ldots, x_n$
OUTPUT: Is $\Sigma x_i \geq n/2$?

MAXIMUM
INPUT: A list of m m-bit binary numbers $(n = m^2)$.
OUTPUT: The largest number in the list.

MERGING
INPUT: Two lists of m m-bit binary numbers, each list sorted in nondecreasing order $(n = 2m^2)$.
OUTPUT: The merged list (a number which appears k times in the input lists will appear duplicated k times in the output list).

MULTIPLE-ADDITION
INPUT: A list of m m-bit binary numbers $A_1, \ldots, A_m$.
OUTPUT: The $(m + \lg(m))$-bit binary representation of the sum ΣA_i.

MULTIPLICATION (see ADDITION)

NETWORK-FLOW
INPUT: m-bit unary numbers $c(i, j)$ for each $1 \leq i, j \leq m$ with $i \neq j$, and an m^2-bit unary number f $(n = m^3)$.
OUTPUT: Is there an integral flow of $\geq f$ from v_1 to v_m in the directed flow network with m vertices and capacity $c(i, j)$ on edge (v_i, v_j) for each i and j?

PARITY (ZERO-MOD-2^c, for constant c)
INPUT: n bits $x_1, \ldots, x_n$.
OUTPUT: Is $\Sigma x_i \not\equiv 0 \bmod 2 \pmod{2^c}$?

SORTING
INPUT: A list of m m-bit binary numbers ($n = m^2$).
OUTPUT: The same list sorted in nondecreasing order (a number which appears k times in the input list will be duplicated k times in the output list).

STRONG-CONNECTIVITY (see CONNECTIVITY)

THRESHOLD
INPUT: m bits $x_1, \ldots, x_m$, and m-bit unary number k.
OUTPUT: Is $\Sigma x_i \geq k$?

UNARY-COUNT (see BINARY-COUNT)

UNARY-TO-BINARY
INPUT: n-bit unary representation of a number k.
OUTPUT: $\lg(n)$-bit binary representation of k.

UNDIR-CYCLE
INPUT: Adjacency matrix of an undirected graph.
OUTPUT: Is there a cycle in the graph?

UNDIR-ST-CONNECTIVITY (see DIR-ST-CONNECTIVITY)

ZERO-MOD-2^c (see PARITY)

Acknowledgment. We thank Amir Schorr for allowing us to include his proof of Theorem 4.1, and we thank Nicholas Pippenger for pointing out the reduction of network flow to bipartite perfect matching.

REFERENCES

[1] R. ALELIUNAS, R. M. KARP, R. J. LIPTON, L. LOVASZ AND C. RACKOFF, *Random walks, universal traversal sequences, and the complexity of maze problems*, Proc. 20th IEEE Symposium on Foundations of Computer Science, 1979, pp. 218-223.
[2] C. BERGE, *Graphs and Hypergraphs*, North-Holland, New York, 1973.
[3] A. BORODIN, *On relating time and space to size and depth*, this Journal, 6 (1977), pp. 733-744.
[4] A. K. CHANDRA, S. FORTUNE AND R. J. LIPTON, *Unbounded fan-in circuits and associative functions*, Proc. 15th ACM Symposium on Theory of Computing, 1983, pp. 52-60.
[5] S. A. COOK, *The complexity of theorem proving procedures*, Proc. Third ACM Symposium on Theory of Computing, 1971, pp. 151-158.
[6] S. EVEN, *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.
[7] S. FORTUNE AND J. WYLLIE, *Parallelism in random access machines*, Proc. Tenth ACM Symposium on Theory of Computing, 1978, pp. 114-118.
[8] M. FURST, J. B. SAXE AND M. SIPSER, *Parity, circuits and the polynomial-time hierarchy*, Proc. 22nd IEEE Symposium on Foundations of Computer Science, 1981, pp. 260-270.
[9] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.
[10] L. M. GOLDSCHLAGER, *A universal interconnection pattern for parallel computers*, J. Assoc. Comput. Mach., 29 (1982), pp. 1073-1086.
[11] J.-W. HONG, *On some deterministic space complexity problems*, this Journal, 11 (1982), pp. 591-601.
[12] J. E. HOPCROFT AND J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[13] N. D. JONES, *Space-bounded reducibility among combinatorial problems*, J. Comput. Sys. Sci., 11 (1975), pp. 68-85.

[14] R. M. KARP, *Reducibility among combinatorial problems*, in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher eds., Plenum Press, New York, 1972, pp. 85-104.

[15] R. E. LADNER, *The circuit value problem is log space complete for* P, SIGACT News, 7 (1975), pp. 18-20.

[16] D. E. MULLER AND F. P. PREPARATA, *Bounds to complexities of networks for sorting and for switching*, J. Assoc. Comput. Mach., 22 (1975), pp. 195-201.

[17] W. J. SAVITCH, *Relationships between nondeterministic and deterministic tape complexities*, J. Comput. Sys. Sci., 4 (1970), pp. 177-192.

[18] A. SCHORR, *Super-circuits*, manuscript, Computer Science Dept., Tel Aviv Univ., Tel Aviv, Israel, Sept. 1981.

[19] Y. SHILOACH AND U. VISHKIN, *Finding the maximum, merging and sorting in a parallel computation model*, J. Algorithms, 2 (1981), pp. 88-102.

[20] S. SKYUM AND L. G. VALIANT, *A complexity theory based on Boolean algebra*, Proc. 22nd IEEE Symposium on Foundations of Computer Science, 1981, pp. 244-253.

[21] L. J. STOCKMEYER AND A. R. MEYER, *Word problems requiring exponential time: preliminary report*, Proc. 5th ACM Symposium on Theory of Computing, 1973, pp. 1-9.

[22] L. STOCKMEYER AND U. VISHKIN, *Simulation of parallel random access machines by circuits*, this Journal, this issue, pp. xxx-xxx.

[23] U. VISHKIN, *Implementation of simultaneous memory address access in models that forbid it*, J. Algorithms, 4 (1983), pp. 45-50.