Math. Systems Theory 17, 13–27 (1984)

Mathematical
Systems Theory

Parity, Circuits, and the Polynomial-Time Hierarchy*

Merrick Furst,[1] James B. Saxe,[2] and Michael Sipser[3]

[1]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., 15213

[2]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., 15213

[3]Department of Mathematics, Massachusetts Institute of Technology, Boston, Mass., 02139

Abstract. A super-polynomial lower bound is given for the size of circuits of fixed depth computing the parity function. Introducing the notion of polynomial-size, constant-depth reduction, similar results are shown for the majority, multiplication, and transitive closure functions. Connections are given to the theory of programmable logic arrays and to the relativization of the polynomial-time hierarchy.

1. Introduction

It is of both practical and theoretical interest to know the size of boolean circuits necessary to compute common boolean functions. Despite considerable effort, the techniques for proving combinatorial lower bounds on circuit size remain primitive. To date, even for circuits restricted to be monotone, linear lower bounds are the best that can be shown for naturally arising functions [14, 2, 16].

Lupanov studied bounded depth circuits in 1961 and showed that parity circuits of depth 2 must have an exponential number of gates [11]. Krapchenko, investigating the complexity of the parity function, proved an n^2 lower bound on the size of parity formulas [9, 10]. Considering a different restriction of the circuit model we prove a superpolynomial lower bound: constant-depth circuits of $\neg$-gates and arbitrary fan-in $\wedge$ and $\vee$-gates must use more than a polynomial number of gates to compute parity, majority, multiplication, or transitive closure.

*Research partially funded by NSF Grant MCS-81-05555 and ONR Grant N00014-76-C-0370.

Constant-depth circuits elegantly model programmable logic arrays (*PLA's*), a type of integrated circuit used inside microprocessors to compactly represent many functions [14]. According to folklore, some common functions, e.g., parity, multiplication, and transitive closure cannot be implemented with small PLA's. The new lower bound establishes a basis for this belief by showing that any PLA implementing these functions must be using more than a polynomial amount of chip area.

Lower bounds for constant-depth circuits relate to questions about the relativization of the polynomial-time hierarchy. We show that an $\Omega(n^{\mathrm{poly}(\log n)})$ lower bound on the size of constant-depth circuits computing parity would imply the existence of an oracle A that separates $\cup_i \Sigma_i^{P,A}$ from PSPACE^A. Readers not interested in this connection may wish to skip Section 2 and proceed directly to Section 3 where the circuit lower bound is proved.

2. The Polynomial-Time Hierarchy

The Meyer-Stockmeyer polynomial-time hierarchy is a ladder of language classes lying between P and PSPACE, as depicted in Figure 1 [15, 20].

A language L is in the class Σ_i^P if and only if the strings y in L can be characterized as satisfying a sentence

$$\exists^P x_1 \forall^P x_2 \cdots Q_i^P x_i R(y, x_1, \ldots, x_i),$$

in which

(i) $\exists^P x_j$ quantify over strings of length polynomial in $|y|$,
(ii) Q_i^P is $\forall^P$, or $\exists^P$ as i is even or odd, and
(iii) R is a deterministic polynomial-time predicate.

$$\begin{tikzcd}
\text{PSPACE} & & \\
\vdots & & \\
\Sigma_2^P \arrow[d, dash] \arrow[rrd, dash] & & \Pi_2^P \arrow[d, dash] \arrow[lld, dash] \\
\Sigma_1^P \arrow[rd, dash] & & \Pi_1^P \arrow[ld, dash] \\
& \Sigma_0^P = P = \Pi_0^P &
\end{tikzcd}$$

Fig. 1. The Polynomial-Time Hierarchy

The class Π_i^P is defined to be co-Σ_i^P. Thus, $P = \Sigma_0^P = \Pi_0^P$, $NP = \Sigma_1^P$, and co$NP = \Pi_1^P$.

Aside from the obvious trivial inclusions, nothing is known about the relationships among these classes. Separating Σ_i^P from Σ_j^P, for some i, j with $0 < i < j$, would separate P, NP, and PSPACE. Baker, Gill and Solovay [3] posed the following question for language classes in the polynomial hierarchy relativized to oracles; does there exist an oracle X such that, for some values of i, $\Sigma_i^{P,X} \subsetneq \Sigma_{i+1}^{P,X}$? The strongest separation theorem known gives an oracle B such that, $\Sigma_2^{P,B} \neq \Pi_2^{P,B}$, which implies $\Sigma_2^{P,B} \subsetneq \Sigma_3^{P,B}$ [4]. The technique used to obtain this result is not strong enough to separate any higher levels.

Parity and the Relativized Polynomial Hierarchy

There is more than one notion of relativized space bounded computation [12]. Permitting machines to make exponentially long oracle queries—by not accounting for the space used on the query tape—makes it quite easy to construct an A that separates $\cup_i \Sigma_i^{P,A}$ and PSPACE^A. The separation problem becomes interesting only if oracle queries must fit within the polynomial-space bound and we adopt this convention.

Definition. The polynomial hierarchy relativized by an oracle A is PH$^A = \cup_i \Sigma_i^{P,A}$.

In what follows we prove that if parity of n inputs cannot, for any k, be computed by constant-depth, $O(n^{\log^k n})$-size circuits, then there exists an oracle A that separates PSPACE^A and PH^A. The proof requires several lemmas.

First, consider the oracle property,

$$L^A = \{1^n |\, \text{the number of strings of length } n \text{ in } A \text{ is odd}\}, \tag{1}$$

as defined by Angluin [1]. It is certainly in PSPACE^A. Can it be in $\Sigma_i^{P,A}$, for some fixed i?

Lemma 2.1. *Let*

$$\sigma(y) = \exists^P x_1 \forall^P x_2 \cdots Q_i^P x_i R^A(x_1,\dots,x_i, y)$$

be a sentence in $\Sigma_i^{P,A}$. There is an equivalent sentence

$$\bar\sigma(y) = \exists^P x_1 \forall^P x_2 \dots Q_{i+2}^P x_{i+2} \bar R^A(x_1,\dots,x_{i+2}, y)$$

in $\Sigma_{i+2}^{P,A}$, where $\bar R^A(x_1,\dots,x_{i+2}, y)$ is a predicate that can be computed by a machine that makes at most one oracle query and runs for a polynomial amount of time.

Proof. We show how $R^A(x_1,\dots,x_i, y)$ can be expressed as

$$\exists^P r \forall^P s \bar R^A(x_1,\dots,x_i, r, s, y),$$

where $\bar R^A(x_1,\dots,x_i, r, s, y)$ can be computed by a machine that makes only one oracle query and runs for a polynomial amount of time. Then, substituting this two quantifier sentence for R^A in σ we obtain $\bar\sigma$.

Let $M_R^A(x_1,\ldots,x_i, y)$ be a polynomial-time machine that computes the predicate $R^A(x_1,\ldots,x_i, y)$. If specific values of $x_1,\ldots,x_i$, and y are given, M runs for a polynomial number of steps, makes some oracle query q_j, receives a response r_j ("$q_j \in A$", or "$q_j \notin A$"), branches one of two ways based on the outcome, and iterates this process a polynomial number of times before finally accepting or rejecting. This computation (whose value is really a function of the oracle A) may be viewed as a binary tree T with internal nodes labeled with oracle queries q_j, edges labeled "$q_j \in A$", or "$q_j \notin A$", and leaves labeled either "accept" or "reject". The sentence

$$\exists^P r \forall^P s \bar{R}^A(x_1,\ldots,x_i, r, s, y)$$

must characterize those oracles A that lead M to accept. Informally, the sentence should say: "there is an accepting path in T along which every query response is consistent with A." More formally, let r range over all possible polynomially long sequences of (query q_j, response r_j) pairs, and let s range over all possible queries q_s. Let $\bar{R}^A(x_1,\ldots,x_i, r, s, y)$ be the predicate

"$r = (q_{j_1}, r_{j_1})\ldots(q_{j_l}, r_{j_l})$ is an accepting branch in T
  and
if q_s appears as query q_{j_t} along branch r then
  r_{j_t} is the correct answer to the question $q_{j_t} \overset{?}{\in} A$".

The first part of this predicate can be computed in polynomial time by simply simulating M. The second part can be determined by making at most one oracle query. □

Actually, a technically stronger result can be shown.

Corollary 2.2. *Let*

$$\sigma(y) = \exists^P x_1 \forall^P x_2 \ldots Q_i^P x_i R^A(x_1,\ldots,x_i, y)$$

be a sentence in $\Sigma_i^{P,A}$. *There is an equivalent sentence*

$$\bar{\sigma}(y) = \exists^P x_1 \forall^P x_2 \ldots Q_{i+1}^P x_{i+1} \bar{R}^A(x_1,\ldots,x_{i+1}, y),$$

in $\Sigma_{i+1}^{P,A}$, *where* $\bar{R}^A(x_1,\ldots,x_{i+1}, y)$ *can be computed by a machine that makes at most one oracle query and runs for a polynomial amount of time.*

Proof. In the proof of Lemma 2.1, the two quantifier sentence equivalent to R could just as easily have been constructed to be a $\forall\exists$ sentence whose polynomial-time predicate requires only one oracle query to evaluate. If quantifier Q_i had been existential (universal), the first quantifier of the two quantifier sentence used to replace R could have been chosen to be existential (universal). This could have allowed a merging of two adjacent levels of like quantifiers, giving a $\Sigma_{i+1}^{P,A}$ sentence $\bar{\sigma}$. □

Lemma 2.3. *If the oracle property* L^A, *defined by* (1) *is in* $\Sigma_i^{P,A}$ *then, for some fixed* k, *there exists a depth* $i+1$, $O(n^{\log^k n})$*-size circuit to compute parity of* n *variables.*

Proof. Suppose 1^n is in L^A if and only if $\exists^P x_1 \forall^P x_2 \ldots Q_i^P x_i R^A(x_1,\ldots,x_i,1^n)$. Let $\bar{\sigma}$ be as in Corollary 2.2. We can think of the sentence

$$\bar{\sigma}(1^n) = \exists^P x_1 \forall^P x_2 \ldots Q_{i+1}^P x_{i+1} \bar{R}^A(x_1,\ldots,x_{i+1},1^n)$$

as describing a height $i+1$ tree with root labeled "$\exists$", internal nodes labeled "$\exists$" or "$\forall$", and leaves labeled "$\bar{R}^A(x_1,\ldots,x_{i+1},1^n)$", for specific values of $x_1,\ldots,x_{i+1}$. Each internal node has $2^{(n^k)}$ sons, one for each possible value of an x_i. The predicate $\bar{R}^A(x_1,\ldots,x_{i+1},1^n)$, for fixed values of $x_1,\ldots,x_{i+1}$, is determined by at most one oracle query q_j. Thinking of the 2^n length-n oracle queries as variables $q_1,\ldots,q_{2^n}$, we see that $\bar{R}^A(x_1,\ldots,x_{i+1},1^n)$ is equivalent to either the literal q_j, or the literal $\bar{q}_j$, for some j. Replacing $\forall$-nodes by $\wedge$-gates, $\exists$-nodes by $\vee$-gates, and the $\bar{R}^A(x_1,\ldots,x_{i+1},1^n)$ leaves by q_j or $\bar{q}_j$ as appropriate, we obtain, for some c, a depth $i+1$, $c^{(n^k)}$-size circuit computing parity of the 2^n variables q_j. Noticing that $c^{(n^k)}$ is $O((2^n)^{\log^k(2^n)})$, the result follows. □

Theorem 2.4. *If the parity function on n variables cannot be computed by constant-depth, $O(n^{\log^k n})$-size circuits, then there exists an oracle A such that* PSPACE$^A \supsetneq$ PH^A.

Proof. Using Lemma 2.3 and the assumption that parity cannot be computed by constant-depth, $\Omega(n^{\log^k n})$-size circuits, as in [4], we can diagonalize away from PH and construct an oracle A' such that $L^{A'} \notin$ PH$^{A'}$. On the other hand, for any A, $L^A \in$ PSPACE^A since a machine need only run through all strings of length n and count how many are in A to determine "$1^n \overset{?}{\in} L^A$". Thus, $L^{A'} \in$ PSPACE$^{A'}$ − PH$^{A'}$. □

We conjecture that there is no k such that parity can be computed by constant-depth, $\Omega(n^{\log^k n})$-size circuits.

3. The Parity Lower Bound

We begin by defining terms.

Definition. For each n, $X_n = \{x_1,\ldots,x_n\}$ is the set of *inputs*, and $\bar{X}_n = \{x_1, \bar{x}_1,\ldots,x_n, \bar{x}_n\}$ is the set of *literals*.

Definition. A *0-circuit* is a literal; an *i-circuit* is a nonempty collection of $(i-1)$-circuits. An i-circuit is an $\wedge$-circuit if i is even and an $\vee$-circuit if i is odd. There are two *constant circuits*, 0 and 1. The function computed by an $\wedge$-circuit ($\vee$-circuit) C is the $\wedge$ ($\vee$) of the functions computed by C's members. The 0-circuit $x_j(\bar{x}_j)$ computes the (negation of the) jth projection function. The constant circuits compute the constant functions. For each circuit C, the function

$$f_C:\{0,1\}^n \to \{0,1\},$$

is the function that C computes.

Definition. For circuits B and C, *B belongs to C*, if B is hereditarily a member of C, i.e., "belongs to" is the transitive closure of "member of". The *depth* of an i-circuit is i, the *size* of an i-circuit is the number of circuits belonging to it. Constant and 0-circuits have size 0 and depth 0.

The circuits defined in this way are equivalent to conventional circuits except they are organized into levels of $\wedge$'s and $\vee$'s that connect only to adjacent levels. An arbitrary circuit of depth d and size s on n inputs can be converted to one of this form having depth d and size at most $4s+2n$. Much of this increase comes from the necessary introduction of trivial gates having only one input. If these are counted as wires, the size of the new circuit is only $2s$.

Definition. A *restriction* is a function $\rho\colon X_n \to \{0,1,*\}$. The natural extension of ρ to $\overline{X}_n$ is made by assigning

$$\rho(\bar{x}_i) = \begin{cases} *, & \text{if } \rho(x_i) = *; \\ \neg\,\rho(x_i), & \text{if } \rho(x_i) \neq *. \end{cases}$$

Restrictions fix some of the inputs of a function while leaving others, those assigned $*$, variable. For any restriction ρ, and function $f\colon \{0,1\}^n \to \{0,1\}$, ρ *induces* a function

$$f^\rho\colon \{0,1\}^k \to \{0,1\},$$

where k is the number of literals assigned $*$. If $\alpha \in \{0,1\}^k$, $f^\rho(\alpha) = f(\alpha^\rho)$, where α^ρ is $\rho(x_1)\rho(x_2)\ldots\rho(x_n)$ with the k $*$'s replaced by the k elements of α.

In the same spirit that restrictions induce new functions, they also induce new circuits. A restriction ρ *forces* an $\wedge$-circuit ($\vee$-circuit) to be 0 if ρ forces any (all) of its members to be 0. Dually, ρ *forces* an $\wedge$-circuit ($\vee$-circuit) to be 1 if ρ forces all (any) of its members to be 1. Also, ρ forces a 0-circuit $x \in \overline{X}_n$ to be a 0 (1) if $\rho(x) = 0$ (1). A restriction ρ induces a circuit C^ρ from the circuit C in the following sense. If ρ forces C to be 0 (1), then C^ρ is the constant circuit 0 (1). If C is not forced by ρ, then C^ρ is recursively defined to be $\{B^\rho \mid B$ is an unforced member of $C\}$.

Lemma 3.1. *For any circuit C, function f, and restriction ρ, if C computes f then C^ρ computes f^ρ.*

Proof. Straightforward. □

Definition. An *n-parity function* is $x_1 + \cdots + x_n \pmod 2$, or the negation of this. A *parity circuit* is any circuit that computes a parity function.

Lemma 3.2. *For any parity function f and restriction ρ, f^ρ is a parity function.*

The following is the main theorem. Its proof occupies most of the rest of this section.

Theorem 3.3. *Parity cannot be computed by constant-depth, polynomial-size circuits.*

Proof. (By contradiction) Let d be the smallest depth admitting polynomial size parity d-circuits. Lupanov proves that parity 2-circuits are exponentially large [12], therefore, d must be at least 3. We proceed in three steps. First we construct, using suitably chosen restrictions of the polynomial-size parity d-circuits, polynomial-size parity d-circuits all of whose 1-circuits are of constant size. Then we apply a second set of restrictions to these circuits obtaining polynomial-size parity d-circuits all of whose 2-circuits are of constant size. Finally, we modify *these* circuits to get polynomial-size parity $(d-1)$-circuits, thereby contradicting the minimality of d.

Step 1. Let $C_1, C_2, \ldots$ be parity d-circuits, where C_n computes an n-parity function and has size $\leqslant n^k$. We show that, for each sufficiently large n, a restriction ρ chosen at random from a suitable distribution has a non-zero probability of inducing from C_n a circuit C_n^ρ such that,

(i) C_n^ρ computes an m-parity function, for some $m \geqslant \sqrt{n}/2$, and

(ii) all the 1-circuits of C_n^ρ are bounded in size, independently of n.

The random restriction $\rho: X^n \to \{0,1,*\}$ is chosen from a distribution that independently assigns, for each i, the probabilities:

$$\Pr[\rho(x_i) \text{ is a } *] = 1/\sqrt{n},$$

$$\Pr[\rho(x_i) \text{ is a } 1] = \Pr[\rho(x_i) \text{ is a } 0] = \frac{1-1/\sqrt{n}}{2}.$$

Fixing a constant c, whose value we exhibit later, ρ *fails* if either C_n^ρ contains a 1-circuit of size $> c$ or ρ assigns $*$ to fewer than $\sqrt{n}/2$ inputs. Since C_n has at most n^k 1-circuits,

$$\Pr[\rho \text{ fails}] \leqslant \Pr[\rho \text{ assigns} < \sqrt{n}/2 *\text{'s}] + n^k \cdot \Pr[\text{a given induced 1-circuit in } C_n^\rho \text{ has size} > c]. \tag{2}$$

Chebyshev's inequality shows that the first term of (2) is bounded above by $O(n^{-1/2})$. (Note that all our inequalities hold for sufficiently large n.) To bound the second term of (2), consider a 1-circuit B and the probability that B^ρ has size $> c$. Either B is *wide*, *i.e*, has size $\geqslant c \ln n$, or B is *narrow*, *i.e.*, has size $< c \ln n$.

Case 1. B is wide.

$$\begin{aligned}
\Pr[B \text{ is not forced}] &\leqslant \Pr[\text{no member of } B \text{ is assigned } 1] \\
&\leqslant (3/4)^{c \ln n}, \qquad (\text{for } n \geqslant 4) \\
&\leqslant n^{c \ln(3/4)} \\
&= o(n^{-c/4}).
\end{aligned}$$

Case 2. B is narrow.

$$\begin{aligned}\Pr[\text{size of } B^\rho > c] &\leqslant \Pr[B \text{ contains} \geqslant c\,*\text{'ed inputs}]\\ &\leqslant \binom{c\ln n}{c}(1/\sqrt{n})^c\\ &\leqslant (c\ln n)^c n^{-c/2}\\ &= o(n^{-c/4}).\end{aligned}$$

If $c = 8k$, then

$$\begin{aligned}\Pr[C_n^\rho \text{ contains a 1-circuit of size} > c] &\leqslant n^k \cdot o(n^{-c/4}) = n^k \cdot o(n^{-k})\\ &= o(n^{-k}).\end{aligned}$$

Hence,

$$\Pr[\rho \text{ fails}] \leqslant n^{-1/2} + o(n^{-k}) = o(1).$$

Thus, for every sufficiently large n, there exists a restriction ρ such that,

(i) C_n^ρ computes an m-parity function, for some $m \geqslant \sqrt{n}/2$,

(ii) the size of $C_n^\rho \leqslant n^k$ is polynomial in m, and

(iii) C_n^ρ contains no 1-circuits of size greater than c.

From these induced circuits it is easy to obtain a sequence $D_1, D_2, \ldots$ of polynomial-size parity d-circuits in which every 1-circuit has size $\leqslant c$.

Step 2. Let $D_1, D_2, \ldots$ be d-circuits, such that, for each n, D_n computes an n-parity function, has at most n^k gates, and contains no 1-circuits of size greater than c. We show that, for each sufficiently large n, a restriction ρ chosen from the distribution used in Step 1 has a non-zero probability of inducing from D_n a circuit equivalent to one which

(i) computes an m-parity function, for some $m \geqslant \sqrt{n}/2$, and

(ii) C_n^ρ has 2-circuits bounded in size independently of n.

Fixing a constant b_c, ρ *fails* if either

(i) ρ assigns fewer than $\sqrt{n}/2\,*$'s, or

(ii) if some 2-circuit of D_n^ρ *depends upon* $\geqslant b_c$ inputs.

As in Step 1, the probability that ρ assigns too few $*$'s is very small. To show that there is some b_c for which the second condition is also unlikely we make the following claim.

Claim. For every c, there is a constant b_c, such that, for any 2-circuit A all of whose 1-circuits are of size at most c, $\Pr[A^\rho$ depends upon $> b_c$ inputs$] = o(n^{-k})$. The claim is proved by induction on c.

Basis. ($c = 1$) The 2-circuit A computes the $\wedge$ function and hence by an argument dual to that used in Step 1, we see that b_1 exists.

Induction. (Show b_c exists given b_{c-1}.) Let $b = k \cdot 4^c$ and consider two cases, one in which A is *wide, i.e*, has $\geqslant b\ln n$ *disjoint* 1-circuits (no common inputs), and the other in which A is *narrow.*

Case 1. A is wide.

$$\begin{aligned}\Pr[A \text{ is not forced}] &\leqslant \Pr[\text{no member of } A \text{ is forced to } 0]\\ &\leqslant \Pr[\text{a 1-circuit of size} \leqslant c \text{ is not forced to } 0]^{b\ln n}\\ &\leqslant (1-4^{-c})^{b\ln n}\\ &= n^{b\ln(1-4^{-c})}\\ &\leqslant n^{-b4^{-c}}.\end{aligned}$$

For $b = k\cdot 4^c$ this is $o(n^{-k})$.

Case 2. A is narrow. Choose a maximal collection of disjoint 1-circuits in A and let H be the set of inputs appearing in this collection. Since A is narrow $|H| \leqslant bc\log n$, and H *hits* (contains at least one input appearing in) each of A's 1-circuits. Let h be the number of $*$'d inputs in H, and let $l = 2^h$. Let $\rho_1, \rho_2, \ldots, \rho_l$ be the 2^h restrictions obtained by setting these $*$'s to 0 or 1 in all possible ways. The value computed by A^ρ can be determined from the values of $A^{\rho_1}, \ldots, A^{\rho_l}$ and the h $*$'d inputs. Furthermore, since H hits every 1-circuit, the 1-circuits in each A^{ρ_i} are of size at most $c-1$. Thus, by the induction hypothesis, $\Pr[A^{\rho_i}$ depends upon $> b_{c-1}$ inputs$] = o(n^{-k})$. By an argument similar to that in Step 1, Case 2, $\Pr[h > 4k] \leqslant o(n^{-k})$. If A^ρ depends upon $> 4k + l\cdot b_{c-1}$ inputs, then either $h > 4k$ or one of the A^{ρ_i} depends upon more than b_{c-1} inputs. Since $l \leqslant 2^{4k}$ whenever $h \leqslant 4k$, letting $b_c = 4k + 2^{4k}\cdot b_{c-1}$, we have

$$\begin{aligned}\Pr[A^\rho \text{ depends upon} > b_c \text{ inputs}] &\leqslant o(n^{-k}) + 2^{4k}\cdot o(n^{-k})\\ &= o(n^{-k}).\end{aligned}$$

Claim proved. Hence, for the value of b_c given by the claim, a non-failing ρ exists. Since any 2-circuit which depends upon only b_c inputs is equivalent to a 2-circuit of size at most $b_c\cdot 2^{b_c}$, we can construct from D_n^ρ an equivalent d-circuit that,

(i) computes an m-parity function, for some $m > \sqrt{n}/2$, and
(ii) has 2-circuits that are of size at most $b_c\cdot 2^{b_c}$.

We thus obtain a sequence $E_1, E_2, \ldots$ of polynomial-size parity d-circuits all of whose 2-circuits are of constant size.

Step 3. The parity d-circuits E_i can be converted to parity $(d-1)$-circuits by rewriting their 2-circuits of size a, using the distributive law, as $\vee$-$\wedge$-circuits of size at most $a\cdot 2^a$. Replacing all 2-circuits in the E_i by these, merging the two now adjacent levels of $\vee$'s (recall that $d \geqslant 3$), and then applying DeMorgan's laws to exchange $\vee$'s and $\wedge$'s, we obtain parity $(d-1)$-circuits at most a constant factor larger.

Thus, we conclude that there exists a sequence $F_1, F_2, \ldots$ of polynomial size parity $(d-1)$-circuits, which is impossible. □

A more careful analysis slightly improves the lower bound. Instead of finding an equivalent $\vee$-$\wedge$ 2-circuit at the end of Step 2, it is sufficient to proceed directly to Step 3 and find an equivalent $\vee$-$\wedge$ circuit there. Parity d-circuits of size $O(n^{f(n)})$ are converted in Step 1 to parity d-circuits of size $O(n^{2f(n)})$ whose 1-circuits are of size $O(f(n))$. Each repetition of Step 2 at most triply exponentiates the size of the 1-circuits. Denoting the kth iterate of the log function by $\log^{(k)}$, and the inverse of the function $g(0)=1$, $g(n)=2^{g(n-1)}$, by $\log^* n$, we have:

Corollary 3.4. *Parity d-circuits must be of size* $\Omega(n^{\log^{(k)} n})$, *where* $k=3(d-2)$.

Corollary 3.5. *Polynomial-size parity circuits must have depth* $\Omega(\log^* n)$.

Corollary 3.6. *The boolean function* "$\equiv k \pmod p$", *for any* k *and* $p \geqslant 2$, *cannot be computed by constant-depth, polynomial-size circuits.*

Proof. An argument exactly analogous to that in the proof of the main theorem confirms this result. □

4. Consequences

In the previous section we showed that parity cannot be computed by constant-depth, polynomial-size circuits. The parity function seems so simple that one immediately wonders if other, seemingly more complicated, functions also cannot be realized in constant depth and polynomial size. Here we prove that majority, multiplication of binary integers, and transitive closure share this property with parity.

Definition. A function f is *constant-depth, polynomial-size reducible* to a function g ($f \leqslant_{cp} g$) if f can be realized with constant-depth, polynomial-size circuits on literals, made up of $\vee$-gates, $\wedge$-gates, $\neg$-gates, and gates computing the function g.

Lemma 4.1. *If Parity is* $\leqslant_{cp}$ *reducible to a function* g, *then* g *is not realizable in constant depth and polynomial size.*

Proof. Straightforward. □

Definition. The Majority predicate on n binary variables is defined to be 1 if and only if more than half of the inputs are 1.

Lemma 4.2. *Parity* $\leqslant_{cp}$ *Majority.*

Proof. (outline) Let $x_1, \ldots, x_n$ be the variables for which we wish to construct a constant-depth, polynomial-size circuit using $\wedge$-gates, $\vee$-gates, and Majority gates. Using a Majority gate and a constant 0-circuit (0 or 1) we can compute the predicate P_k = "at least k bits are on" in depth one, and polynomial size. We compute parity as

$$(P_0 \wedge \overline{P_1}) \vee (P_2 \wedge \overline{P_3}) \vee (P_4 \wedge \overline{P_5}) \vee \cdots . \qquad \square$$

This lemma and Lemma 4.1 yield the following.

Theorem 4.3. *Majority cannot by realized by constant-depth, polynomial-size circuits.*

We now show that circuits performing multiplication of binary integers in constant depth require more than a polynomial number of gates. This proves that multiplication cannot be implemented by polynomial-size program logic arrays (PLA's.) We point out, for contrast, that addition can be realized in constant depth and polynomial size, and hence can be implemented by polynomial-size PLA's.

Definition. Let $a = (a_i)$, $b = (b_i)$ be two n-bit binary numbers. A Multiplication gate has $2n$ inputs (a_i), (b_i), and $2n$ outputs, where the output bits are the $2n$ bits of the product of a and b.

Lemma 4.5. *Parity $\leqslant_{cp}$ Multiplication.*

Proof. (outline) Let $x_0, \ldots, x_{n-1}$ be the variables for which we wish to construct a constant-depth, polynomial-size circuit using $\vee$-gates, $\wedge$-gates, and multiplication gates. Let $k = [\log n]$. Define two kn-bit binary numbers a and b as follows:

$$a = \sum_{i=0}^{n-1} a_i 2^{ki}, \text{ and}$$

$$b = \sum_{i=0}^{n-1} b_i 2^{ki},$$

where for all i, $b_i = 1$ and $a_i = x_i$. The $2kn$ bits of these numbers can easily be computed from the x_j with a circuit of depth 1.

Consider the product

$$ab = \sum_{i=0}^{2n-1} c_i 2^{ki},$$

where each c_i is a k-bit binary number. The low order bit of the number

$$c_{n-1} = \sum_{i=0}^{n-1} a_i b_{n-1-i} = \sum_{i=0}^{n-1} x_i$$

is the parity of the x_i, Figure 2. Therefore, given a multiplication gate, we can compute parity with constant-depth, polynomial-size circuits $\square$

Theorem 4.6. *Multiplication cannot be computed by constant-depth, polynomial-size circuits.*

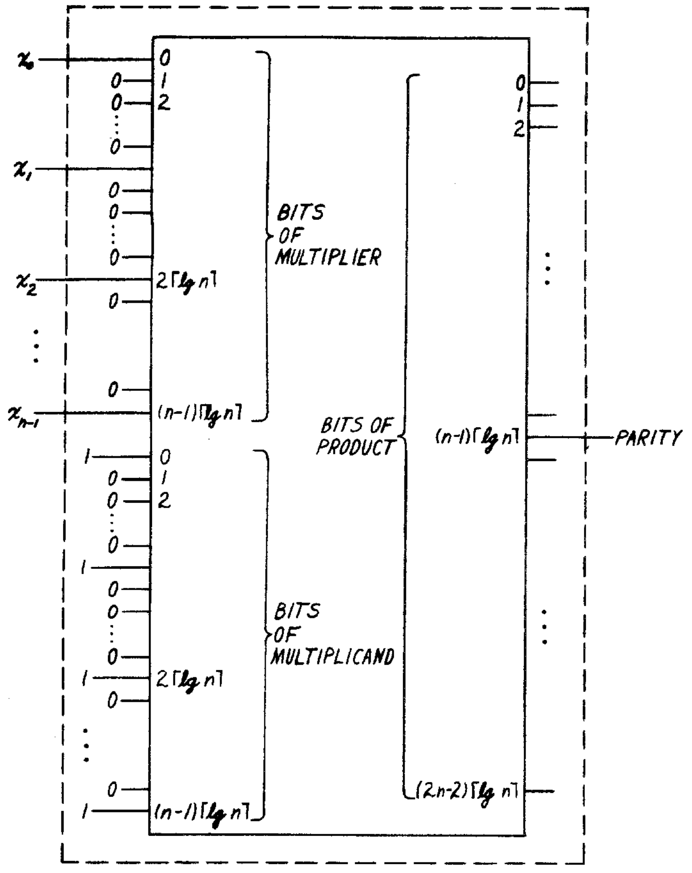

Fig. 2. Parity $\leqslant_{cp}$ Multiplication.

Proof. From Lemmas 4.1 and 4.2. [6] □

Definition. Let $A=(a_{ij})$ be the n by n adjacency matrix for a graph G. A Transitive Closure gate has n^2 inputs a_{ij} and n^2 outputs a^*_{ij} such that $A^*=(a^*_{ij})$ is the adjacency matrix for the transitive closure of G.

Lemma 4.7. (J. Byrd) *Parity* $\leqslant_{cp}$ *Transitive Closure.*

Proof. Let $x_1,\dots,x_n$ be the variables for which we want to construct a constant-depth, polynomial-size parity circuit using $\vee$-gates, $\wedge$-gates, and transitive closure gates. Let G be a graph with $n+2$ vertices defined by a 0-1 assignment to the x_i in the following way. The vertices of G are labeled

$$v_{\text{start}}, v_{x_1},\dots,v_{x_n}, v_{\text{end}}.$$

There is an edge between v_{start} and the first v_{x_i}, for which $x_i=1$. There is an edge between v_{end} and the last v_{x_j} for which $x_j=1$. There is also an edge between v_{x_i}

and v_{x_j} if $i < j$, and

(i) $x_i = 1$,

(ii) $x_{i+1} = x_{i+2} = \cdots = x_{j-1} = 0$, and

(iii) $x_j = 1$.

Thus G contains a single path linking v_{start} to v_{end} passing through the "on" variables from left to right, Figure 3.

From the literals $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ we can compute the n^2 bits of the adjacency matrix $A = (a_{ij})$ for G with $\wedge$-gates as follows.

$$\begin{aligned} a_{\text{start},i} &= \bar{x}_1 \wedge \cdots \wedge \bar{x}_{i-1} \wedge x_i, \\ a_{i,\text{end}} &= x_i \wedge \bar{x}_{i+1} \wedge \cdots \wedge \bar{x}_n, \\ a_{ij} &= x_i \wedge \bar{x}_{i+1} \wedge \cdots \wedge \bar{x}_{j-1} \wedge x_j. \end{aligned}$$

Consider the graph G^2 on the vertices

$$v'_{\text{start}}, v'_{x_1}, \ldots, v'_{x_n}, v'_{\text{end}}$$

in which there is an edge from v'_r to v'_s if and only if there is a path of length 2 in G from v_r to v_s, Figure 3. The n^2 bits of the adjacency matrix $B = (b_{ij})$ for G^2 can be computed from the a_{ij} with $\vee$-gates and $\wedge$-gates as follows,

$$b_{rs} = (a_{r1} \wedge a_{1s}) \vee (a_{r2} \wedge a_{2s}) \vee \cdots \vee (a_{rn} \wedge a_{ns}).$$

Therefore, the b_{ij} can be computed in constant depth and polynomial size from the x_i. Take the transitive closure $B^* = (b^*_{ij})$ of B with a transitive closure gate. The bit $b^*_{\text{start,end}}$ is a 1 if and only if the sum of $x_1, \ldots, x_n$ is odd. Thus Parity $\leqslant_{cp}$ Transitive Closure. □

Skyum and Valiant have shown that any function computable by polynomial-size formulas can be reduced by projections to transitive closure [18]. This is an alternative proof of Lemma 4.7.

Theorem 4.8. *Transitive Closure cannot be realized in constant depth and polynomial size.*

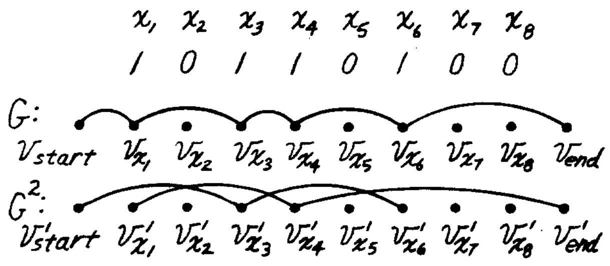
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
1 0 1 1 0 1 0 0
G:
v_{start} v_{x_1} v_{x_2} v_{x_3} v_{x_4} v_{x_5} v_{x_6} v_{x_7} v_{x_8} v_{end}
G^2:
v'_{start} v'_{x_1} v'_{x_2} v'_{x_3} v'_{x_4} v'_{x_5} v'_{x_6} v'_{x_7} v'_{x_8} v'_{end}

Fig. 3. The graphs G and G^2.

5. Areas for Additional Research

(1) Better lower bounds: It is straightforward to give $O(2^{n^{1/(d-1)}})$ sized parity d-circuits. How can the gap between upper and lower bounds be tightened?

(2) Polynomial lower bounds: The bits of Addition can be computed by polynomial-size 3-circuits. Is it possible to compute them with linear size d-circuits for some d? Can any size-depth trade-off be given?

(3) Polynomial-size, constant-depth reduction: Elucidate the structure of this reducibility. We conjecture that the majority function does not reduce to the parity function.

(4) Connections with the infinitary version: The proofs of the main theorem of this paper and the main theorem in [20] are structurally similar. Is there any formal connection between them? Is there a finitary version of the measure theoretic proof of the infinitary version of the theorem presented in [8]?

(5) More powerful models: Polynomial-size, bounded-width decision dags can simulate polynomial-size, bounded-depth circuits, as well as compute functions such as parity and "$\equiv 0 \pmod{k}$" [7]. Can they compute the majority function?

6. Acknowledgments

The authors would like to thank Jim Byrd, Steven Fortune, and Charles Leiserson for many helpful conversations. Steve Fortune first suggested that the depth 3 case might be tractable, Jim Byrd found the proof showing that parity is constant-depth, polynomial-size reducible to transitive closure, and Charles Leiserson pointed out the connection to PLA's. Our special thanks go to Hania Gajewska, Dan Hoey, Martin Tompa, and the referees for carefully reading earlier versions of this manuscript and suggesting many valuable improvements.

IBM, at Yorktown Heights and San Jose, provided two of the authors with congenial atmospheres while this research was in progress. The authors were partially funded by NSF grant MCS-81-05555, ONR grant N00014-76-C-0370, and an IBM Graduate Fellowship.

References

1. D. Angluin, Counting problems and the polynomial-time hierarchy. *Theoretical Computer Science*, to appear.
2. N. Blum, A $2.75n$ lower bound for the combinational complexity of boolean functions. University of Saarbrucken, Technical Report.
3. T. Baker, J. Gill, and R. Solovay, Relativizations of the $P =^? NP$ question. *SIAM Journal of Computing*, 4, 4, 1975.
4. T. Baker and A. Selman, A second step toward the polynomial hierarchy. *Theoretical Computer Science*, 8, 2, 1979, pp. 177–187.
5. A. Chandra, D. Kozen, and L. Stockmeyer, Alternation. *Journal of the ACM*, 28, 1, January 1981.
6. Digital Equipment Corporation , *Decsystem 10 Assembly Language Handbook*. Third Edition, 1973, pp. 51–52.
7. M. Furst, Bounded width computation DAG's. In preparation, 1982.

8. M. Furst, J. B. Saxe, M. Sipser, Parity, circuits and the polynomial-time hierarchy. 22ND *Symposium on the Foundations of Computer Science*, 1981, pp. 260–270.
9. M. Furst, J. B. Saxe, M. Sipser, Depth 3 circuits require $\Omega(n^{c \log n})$ gates to compute parity: a geometric argument. In preparation.
10. V. Krapchenko, Complexity of the realization of a linear function in the class of II-circuits. English translation in *Math. Notes Acad. Sci., USSR*, 1971, pp. 21–23; orig. in *Mat. Zamet*, 9, 1, pp. 35–40.
11. V. Krapchenko, A method of obtaining lower bounds for the complexity of II-schemes. English translation in *Math. Notes Acad. Sci USSR*, 1972, pp. 474–479; orig. in *Mat. Zamet*, 10, 1, pp. 83–92.
12. O. Lupanov, Implementing the algebra of logic functions in terms of constant-depth formulas in the basis $+, *, -$. English translation in *Sov. Phys.-Dokl.*, 6, 2, 1961; orig. in *Dokla. Akad. Nauk SSSR*, 136, 5.
13. R. Ladner and N. Lynch, Relativization of questions about log space computability. *Mathematical Systems Theory*, 10, 1, 1976.
14. C. Mead and L. Conway, *Introduction to VLSI Systems*. Addison-Wesley, Reading, Mass. 1980.
15. W. Paul, A $2.5N$ lower bound for the combinational complexity of boolean functions. 7th *Annual ACM Symposium on Theory of Computing*, 1975, pp. 27–36.
16. J. Savage, *The Complexity of Computing*. John Wiley and Sons, New York, 1976, Sect. 2.4.
17. C. P. Schnorr, A $3n$ lower bound on the network complexity of boolean functions. *Theoretical Computer Science*, 10, 1, 1980, p. 83.
18. L. J. Stockmeyer, The polynomial-time hierarchy. *Theoretical Computer Science*, 3, 1, 1976, pp. 1–22.
19. S. Skyum and L. G. Valiant, A complexity theory based on boolean algebra. 22nd *Symposium on the Foundations of Computer Science*, 1981, pp. 244–253.
20. M. Sipser, On polynomial vs exponential growth. In preparation.
21. L. Stockmeyer and A. Meyer, Word problems requiring exponential time, preliminary report. 5th *Annual ACM Symposium on Theory of Computing*, 1973.

Received May 1982 and in revised form July 13, 1983 and August 22, 1983.