A $2.5n$ -LOWER BOUND ON THE COMBINATIONAL COMPLEXITY OF BOOLEAN FUNCTIONS*

WOLFGANG J. PAUL†

Abstract. Consider the combinational complexity $L(f)$ of Boolean functions over the basis $\Omega = \{f|f: \{0, 1\}^2 \rightarrow \{0, 1\}\}$. A new method for proving linear lower bounds of size $2n$ is presented. Combining it with methods presented in Savage [13, (1974)] and Schnorr [18, (1974)], we establish for a special sequence of functions $f_n: \{0, 1\}^{n+2 \log(n)+1} \rightarrow \{0, 1\}$: $2.5n \leq L(f) \leq 6n$. Also a trade-off result between circuit complexity and formula size is derived.

Key words. combinational complexity, circuit size

1. Introduction. The interest in lower bounds for the combinational complexity of Boolean functions stems from two facts:

- (i) practical interest: the hardware of a computer consists largely of switching networks;
- (ii) theory of algorithms: proving lower bounds for the run time of algorithms has been reduced to the problem of proving lower bounds for the combinational complexity of Boolean functions [2], [15], [17]. However, for this purpose lower bounds of size greater than $O(n \log(n))$ would be needed.

Several functions with exponential combinational complexity are known [8], [20]. If one measures the complexity of the definition of a sequence of functions $f_n: \{0, 1\}^n \rightarrow \{0, 1\}$ by the complexity of a Turing machine which accepts $\{x | f_{\text{length}(x)}(x) = 1\}$, then all functions which can presently be shown to have exponential complexity are defined by Turing machines of exponential tape complexity.

The proof relies on the fact that (for reasonable functions $s: \mathbb{N} \rightarrow \mathbb{N}$) a Turing machine with tape complexity $s(n) \log s(n)$ —or a formal system with equivalent descriptive power—can construct an optimal circuit with complexity $s(n)$.

There is however a great interest in nonlinear lower bounds for functions which are definable by Turing machines with polynomial time complexity or even with complexity around $n \log n$ (e.g., multiplication). Also, if by proving a lower bound on circuit size one wishes, say, to settle the $P = NP?$ question [1], [4] in a negative sense, one has to prove a lower bound which grows faster than any polynomial for a sequence of functions definable in NP . The best lower bounds known for sequences of functions definable in NP and relative to the full basis of 2-input 1-output gates are of size $2n$ [18].

We first list some basic notations and facts. In Theorem 1, with known techniques a $2n$ -lower bound is derived for a function which describes reading in a storage of n cells of 1 bit each.

To prove a lower bound better than $2n$, new methods are needed which—unlike the known ones—assert the existence of gates which are not necessarily near the inputs of the circuit. Theorem 2 is proven with such a method. Applied alone, the method yields $2n$ -lower bounds.

* Received by the editors April 28, 1975, and in final revised form June 1976.

† Department of Computer Science, Cornell University, Ithaca, New York 14853, and Universität Saarbrücken, Saarbrücken, West Germany. This research was supported by DAAD (German Academic Exchange Service) under Grant 430/402/563/5.

In Theorem 3, extension of the known techniques together with the new method are applied to prove a $2.5n$ -lower bound. This proof turns out to be considerably more difficult than proofs of $2n$ -lower bounds.

In the last section we point out a relation to Neciporuk's test [9] and prove a trade-off result between formula size and circuit complexity.

2. Basic notations and facts. Let $K = \{0, 1\}$.

$\Omega = \{f \mid f: K^2 \rightarrow K\}$ is the set of *basic operations*. A *switching circuit* N is a directed, acyclic graph, each node having indegree 0 or 2. A node with indegree 0 is an *input node*. A node with outdegree 0 is an *output node*. $X_n = \{x_1, \dots, x_n\}$ is the set of input nodes of N .

Each node with indegree 2 is labeled by an $f \in \Omega$. Associate with each node C of N a function $\text{res}_N(C): K^n \rightarrow K$ in the obvious way. In the same way associate functions $\text{res}_N(e)$ with the edges e of N . If it is clear, which circuit is meant, the subscript N in $\text{res}_N(\cdot)$ is dropped.

For $f: K^s \rightarrow K$, N *computes* f if there is a node n in N such that $\text{res}(n) = f$. Define the *combinational complexity* of f by $L(f) = \min\{\# \text{ nodes in } N \text{ which are no input nodes} \mid N \text{ computes } f\}$.

Assigning values 0 or 1 to the variables of a subset $S \subset X_n$ allows us to concentrate on subfunctions as well as on the subcircuit of N , which consists only of paths from an $x_i \in S$ to a terminal node.

A partial assignment (P.A.) is a set of assignments $\alpha = \{x_i := c_i \mid c_i \in K, i \in J \subseteq [1 : n]\}$.

Given $f: K^n \rightarrow K$, a P.A. α implies in a natural way a subfunction $f_A: K^{n-\#\alpha} \rightarrow K$ of f . Functions $\text{res}(C)_\alpha$ will be of special interest.

A path $(n \Rightarrow n')$ is a path from node n to node n' . Let p be a path $(n \Rightarrow n')$. Denote by $[n, n']_p$ the set of nodes of p . If there is only one path $(n \Rightarrow n')$, the subscript p is dropped. Define

$$\begin{aligned} [n, n']_p &= [n, n']_p - \{n'\}, \\ (n, n']_p &= [n, n']_p - \{n\}, \\ (n, n')_p &= [n, n']_p - \{n, n'\}. \end{aligned}$$

For sets of nodes S, S' define

$$\exists \text{ path } (S \Rightarrow S'): \exists n \in S, n' \in S': \exists \text{ path } (n \Rightarrow n').$$

Example. In Fig. 1, $\exists \text{ path } (A \Rightarrow C)$; $\exists \text{ path } ([A, C] \Rightarrow [D, F])$.

FIG. 1

Denote by $\wedge$, $\vee$, $\oplus$ and $\neg$ logical AND, OR, $+\text{mod } 2$, and negation, respectively.

For Boolean variables or functions x and $a \in K$, denote by

$$x^a = \begin{cases} x & \text{if } a = 1, \\ \neg x & \text{if } a = 0. \end{cases}$$

We sum up some simple facts which will be used later, often without being mentioned explicitly.

LEMMA 1.

(i) $(\neg a) \oplus b = a \oplus (\neg b) = \neg(a \oplus b)$.

(ii) *In an optimal circuit all labelings of nodes are of the form*

$$(y_1^a \wedge y_2^b)^c \quad (\text{AND-type gate})$$

or

$$(y_1 \oplus y_2)^d \quad (\oplus\text{-type gate}) \quad (a, b, c, d \in K).$$

(All other gates compute trivial functions and can be “absorbed” into adjacent nontrivial gates.)

So one can consider an optimal circuit as consisting of $\wedge$, $\oplus$ and $\neg$, where L counts only the number of $\wedge$ ’s and $\oplus$ ’s. A function of two variables, which can be computed by an AND-type gate ($\oplus$ -type gate) we call an AND-like function ($\oplus$ -like function).

(iii) *No AND-like function is $\oplus$ -like and vice versa.*

(iv) *An optimal circuit for a single function f has exactly one output node t with $\text{res}(t) = f$.*

From now on the letter t stands always for the output node of the circuit under consideration.

(v) *If one output of a gate is constant, independent of the assignments of values to the input nodes, this gate can be eliminated.*

(vi) *In an optimal circuit no two edges, starting from one node, go into the same node.*

(vii) *If two functions f and g are equal, then $f_\alpha = g_\alpha$ for all P.A.’s α .*

In the various upcoming figures the symbolism of Fig. 2 will be used. Occasionally branching points of splitting paths will be treated like nodes without cost. Negations will not be drawn in figures.

3. Two $2n$ -lower bounds. The first theorem has a practical aspect. It gives an answer to the question: how much does it cost to read from storage? On the other hand, its proof introduces some techniques, which will be used later.

For $\mathbf{a} = a_1 \cdots a_r \in K^*$ denote by

$$(\mathbf{a}) = \text{binary number represented by } \mathbf{a} + 1.$$

THEOREM 1. *Define $f_n: K^{n+\log(n)} \rightarrow K$ by:*

$$f_n(a_1, \dots, a_{\log(n)}, x_1, \dots, x_n) = x_{(\mathbf{a})}.$$

Then $2n - 2 \leq L(f_n) \leq 3n - 3$.

Proof. We first prove the lower bound. Define for $1 \leq s \leq n$ the statement E_s :

E_s : For any function $f: K^{n+\log n} \rightarrow K$ with the property $[\exists S \subseteq [1:n], \#S = s$

FIG. 2

such that for all $\mathbf{a}$ with $(\mathbf{a}) \in S$: $f(\mathbf{a}, x_1, \dots, x_n) = x_{(\mathbf{a})}$] holds: $2s - 2 \leq L(f)$.

E_1 is trivially true. We show that E_s holds if E_{s-1} holds.

Consider an optimal realization N of a function f as defined in E_s :

Case 1. $\exists i \in S$: $\text{outdegree}(\text{input node } x_i) \geq 2$. Thus setting $x_i := \text{const.}$ eliminates at least two gates. The new circuit N obtained by elimination of the gates computes a function f' for which E_{s-1} applies with $S' = S \setminus \{i\}$. If N was cheaper than $2s - 2$, then N' is cheaper than $2(s-1) - 2$, thus contradicting E_{s-1} .

Case 2. $\exists i \in S$: $\text{outdegree}(\text{input node } x_i) = 1$ and the edge from x_i goes into an AND-type gate C , i.e., $\text{res}(C) = (x_i^a \wedge y^b)^c$ with $a, b, c \in K$ and y some function of the other input variables. Claim: $C \neq t$ (recall that t was defined to be the terminal node). As $s \geq 2$, $\exists j \in S$ such that $j \neq i$. Fix the variables in $\mathbf{a}$ such that $(\mathbf{a}) = j$. Call this P.A. α . By definition of f , $f_\alpha = \text{res}(t)_\alpha = x_j$. But setting $x_i := (\neg a)$ yields $(\text{res}(C)_\alpha)_{x_i := (\neg a)} = (\neg c)$ independent of x_j . Hence $C \neq t$. Fix $x_i := (\neg a)$. This eliminates C and its successor.

Case 3. $\exists i \in S$: $\text{outdegree}(\text{input node } x_i) = 1$ and the edge from x_i goes into a $\oplus$ -type gate C . For the P.A. α of Case 2, $\text{res}(t)_\alpha$ is independent of x_i . But $\text{res}(C)_\alpha$ depends on x_i , hence $C \neq t$.

Consider Fig. 3. Identify x_i with y . In the new circuit N' , $\text{res}_{N'}(C) = \text{const.}$ This eliminates C and its successor. $\text{res}_{N'}(t)$ is again a function in the sense of E_{s-1} with $S' = S \setminus \{i\}$.

This completes Case 3 and the proof of the lower bound.

In order to obtain the upper bound first observe that if n is not a power of two, f_n is not completely specified. We give a realization of a function which takes the same values as f_n where f_n is specified.

FIG. 3

Let $n = n_1 + n_2$ where n_1 is a power of two and $2n_1 \geq n$. Let $k = \lfloor \log(n_2) \rfloor$. Then

$$f_n(a_1, \dots, a_{\log(n)}, x_1, \dots, x_n) = ((\neg a_1) \wedge f_{n_1}(a_2, \dots, a_{\log(n)}, x_1, \dots, x_{n_1})) \vee (a_1 \wedge f_{n_2}(a_{\log(n)-k+1}, \dots, a_{\log(n)}, x_{n_1+1}, \dots, x_n)).$$

Thus $L(f_n) \leq L(f_{n_1}) + L(f_{n_2}) + 3$. Also $L(f_1) = 0$. By induction follows $L(f_n) \leq 3n_1 - 3 + 3n_2 - 3 + 3 = 3n - 3$. $\square$

While the argument in the proof of Theorem 1 was limited to the top of the circuit (i.e., near the input nodes), the next proof establishes the existence of branching points, which may be deep in the circuit.

THEOREM 2. *If for $f: K^n \rightarrow K \exists S \subseteq [1:n]$ such that $[\forall i, j \in S, i \neq j \exists P.A.'s \alpha_{ij}^1, \alpha_{ij}^2]$ which fix all variables except x_i and x_j such that*

$$\begin{aligned} f_{\alpha_{ij}^1}(x_i, x_j) &= (x_i^a \wedge x_j^b)^c, \\ f_{\alpha_{ij}^2}(x_i, x_j) &= (x_i \oplus x_j)^d \quad (a, b, c, d \in K), \end{aligned}$$

then $L(f) \geq 2(\#S) - 2$.

We first prove two lemmas.

LEMMA 2. *Let G be a directed acyclic graph such that:*

- (i) G has n input nodes;
- (ii) G has m output nodes;
- (iii) Any interior node (i.e., which is not an input node) has indegree ≤ 2 ;
- (iv) G contains p nodes with outdegree ≥ 2 .

Then G has at least $n - m + p$ interior nodes.

FIG. 4

Proof. Begin to construct G from the input nodes by inserting node after node. In the beginning one has n “wires hanging down” from the input nodes. One interior node decreases the “number of down hanging wires” only by one. The number is increased during the construction by at least p . In the end it is m . Hence G contains at least $n + p - m$ interior nodes. $\square$

Lemma 2 will be applied several times to certain subgraphs of circuits. The next lemma helps to establish the existence of branching points in circuits for functions f , to which the hypothesis of Theorem 2 applies:

LEMMA 3. $\forall i, j \in S, i \neq j$: before a path $(x_i \Rightarrow t)$ meets a path $(x_j \Rightarrow t)$, one of these paths splits.

Proof. Suppose the lemma is false for i and j . Consider Fig. 4. Let C be the node where the paths meet. Let e_1, e_2 be the input wires of C . Let $\text{res}(t)$ be the function computed by the circuit. We show: if C is an AND-type gate, then all subfunctions of $\text{res}(t)$, which depend only on x_i and x_j are AND-like or constant. If C is a $\oplus$ -type gate, all such subfunctions are $\oplus$ -like or constant. But f has subfunctions of both types. More precisely:

Take any P.A. α which fixes all inputs except x_i, x_j . As—by assumption—neither path $(x_i \Rightarrow C)$ nor path $(x_j \Rightarrow C)$ splits:

$$\text{res}(e_1)_\alpha = x_i^{a_1} \text{ or } 0 \text{ or } 1,$$

$$\text{res}(e_2)_\alpha = x_j^{a_2} \text{ or } 0 \text{ or } 1,$$

$$\text{res}(t)_\alpha = \text{res}(C)_\alpha^{a_3} \text{ or } 0 \text{ or } 1.$$

Suppose C is an AND-type gate. Then

$$\text{res}(t)_\alpha = (x_i^{a_4} \wedge x_j^{a_5})^{a_6} \text{ or } 0 \text{ or } 1 \neq f_\alpha$$

by Lemma 1.3.

In the same way one excludes: C is a $\oplus$ -type gate. $\square$

Proof of Theorem 2. There are $\#S - 1$ input nodes $x_i, i \in S$, such that any path $(x_i \Rightarrow t)$ splits. Suppose that this is false. Then there are $i_0, j_0 \in S$ such that there is only one path $(x_{i_0} \Rightarrow t)$ and one path $(x_{j_0} \Rightarrow t)$. This contradicts Lemma 3. Hence on $\#S - 1$ paths $(x_i \Rightarrow t)$ there are first nodes D_i with outdegree $(D_i) \geq 2$. Again by Lemma 3 these are mutually distinct. Now the theorem follows from Lemma 2. $\square$

That the branching points need not be directly under the input nodes is shown in Fig. 5.

Example. For any $c \in [1 : n - 1]$ and any

$$f_c: K^n \rightarrow K \quad \text{such that} \quad f_c(x_1, \dots, x_n) = 1 \text{ if } \sum x_i = c,$$

the following is true:

$$L(f_c) \geq 2n - 2.$$

4. A $2.5n$ -lower bound. Notation:

$$\mathbf{a}_1 = a_1, \dots, a_{\log(n)},$$

$$\mathbf{a}_2 = a_{\log(n)+1} \dots a_{2 \log(n)} \quad (a_i \in K).$$

THEOREM 3. For $f: K^{n+2 \log(n)+1} \rightarrow K$,

$$f(a_1, \dots, a_{2 \log(n)}, q, x_1, \dots, x_n) = \begin{cases} x_{(\mathbf{a}_1)} \wedge x_{(\mathbf{a}_2)} & \text{if } q = 1, \\ x_{(\mathbf{a}_1)} \oplus x_{(\mathbf{a}_2)} & \text{if } q = 0, \end{cases}$$

$$2.5n - 2 \leq L(f) \leq 6n + C \quad (C \in \mathbb{N} \text{ independent of } n).$$

Proof. Note that each of Theorem 1 and Theorem 2 implies $2n$ -lower bounds for f . The upper bound is an easy consequence of Theorem 1. As in the proof of Theorem 1, we first make f independent of all inputs x_i , which allows us to

FIG. 5

eliminate 3 gates (see Cases I–IV). When that is no longer possible, we will know quite exactly how the “top” of the circuit looks. For the remaining s inputs, we prove the existence of $5s/2 - 2$ gates without an inductive argument. Define for $1 \leq s \leq n$ the statement E_s :

E_s : For any function $f: K^{n+2 \log(n)+1} \rightarrow K$ with the property

$\left[\exists S \subseteq [1:n], \#S = s \text{ such that for } \mathbf{a}_1, \mathbf{a}_2 \text{ with } (\mathbf{a}_1), (\mathbf{a}_2) \in S: \right.$

$$f(\mathbf{a}_1, \mathbf{a}_2, q, x_1, \dots, x_n) = \begin{cases} x_{(\mathbf{a}_1)} \wedge x_{(\mathbf{a}_2)} & \text{if } q = 1, \\ x_{(\mathbf{a}_1)} \oplus x_{(\mathbf{a}_2)} & \text{if } q = 0 \end{cases}$$

holds, $2.5s - 2 \leq L(f)$.

E_1 is trivially true. To see that E_{s-1} implies E_s , consider an optimal realization N of a function f as defined in E_s :

Case I: $\exists i \in S$ such that $\text{outdegree}(\text{input node } x_i) \geq 3$. This is similar to Case 1 in the proof of Theorem 1.

Case II: $\exists i \in S$ such that $\text{outdegree}(\text{input node } x_i) = 2$ and one of the edges e_1, e_2 going out from x_i goes into an AND-type gate. This is almost like Case 2 in the proof of Theorem 1. A small difficulty occurs in the case of Fig. 6.

If the edge $(g_1 \rightarrow g_2)$ does not split, N is not optimal. Hence it splits. Now the method of Case 2 of the proof of Theorem 1 work, no matter if g_1 or g_2 is the AND-type gate.

Case III. $i \in S$: $\text{outdegree}(\text{input node } x_i) = 2$ and both edges from x_i go into $\oplus$ -type gates A_1 and A_2 . Consider Fig. 7. Let B (C) be the first place on a path $(A_1 \Rightarrow t)$ ($(A_2 \Rightarrow t)$) which is an AND-type gate or a branching point. The existence of B or C will be shown in a moment. Denote by $f_1, \dots, f_s$ ($g_1, \dots, g_t$) the functions associated with the other input wires to this path. By definition of B

FIG. 6

and C , there is only one path $(A_1 \Rightarrow B)$ and only one path $(A_2 \Rightarrow C)$. Because of Lemma 1 (part (i)) one can assume that all gates in $[A_1, B']$ and in $[A_2, C']$ are $\oplus$ -gates (all occurring negations can be removed by possibly negating f_1 or g_1 or both).

LEMMA 4. $[A_1, B'] \cap [A_2, C'] = \emptyset$.

Proof. Suppose not. Then the situation is like that in Fig. 8. Hence

$$\text{res}(D) = f_1 \oplus \cdots \oplus g_q \oplus x_i \oplus x_i = f_1 \oplus \cdots \oplus g_q.$$

As no other path goes out of $[A_1, D)$ and $[A_2, D)$, $f_1, \dots, g_q$ are independent of x_i . Hence $\text{res}(D)$ is independent of x_i . By the same reasoning, $\text{res}(t)$ depends on x_i only via $\text{res}(D)$. Hence $\text{res}(t)$ is independent of x_i . But as $i \in S$, f depends on x_i .

COROLLARY 1. B' and C' are different.

COROLLARY 2. B or C exists.

Proof. Suppose not; then $B' = C' = t$, contradicting Lemma 4.

LEMMA 5. If B and C exist:

$$(\neg \exists \text{ path } (B \Rightarrow [A_2, C'])) \vee (\neg \exists \text{ path } (C \Rightarrow [A_1, B']));$$

if without loss of generality only C exists:

$$\neg \exists \text{ path } (B' \Rightarrow [A_2, C']).$$

FIG. 7

FIG. 8

Proof. If B and C and both paths exist, the graph N contains a cycle.

If only C exists, the existence of a path $(B' \Rightarrow [A_2, C'])$ implies the existence of B .

From now on without loss of generality C exists and if B exists, then $\neg \exists$ path $(B \Rightarrow [A_2, C'])$.

LEMMA 6. $C \neq t$.

Proof. Suppose $C = t$. Then C is an AND-type gate and $\exists$ path $(A_1 \Rightarrow C)$. This implies the existence of B . Hence if B does not exist, the lemma is true. If B exists, proceed as follows: Since $\#S \geq 2$, $\exists j \in S$, $j \neq i$. Fix $\mathbf{a}_1, \mathbf{a}_2$ such that $(\mathbf{a}_1) = (\mathbf{a}_2) = j$, $q = 1$. Fix all other variables except x_i and x_j arbitrarily. Call this P.A. α . Then $f_\alpha(x_i, x_j) = x_j$ independent of x_i . We prove that the output $\text{res}(t)_\alpha$ still depends on x_i and that, by fixing x_i to the right constant, one can produce a false output:

$\neg \exists \text{ path } (B \Rightarrow [A_2, C'])$ implies $g_1, \dots, g_t$ do not depend on x_i .

Hence

$$\text{res}_\alpha(C) = x_i \oplus h(x_j)$$

where h is some Boolean function of one variable. Furthermore because C is an AND-type gate:

$$\text{res}_\alpha(C) = (\text{res}_\alpha(C')^a \wedge y(x_i, x_j)^b)^c$$

for some constants $a, b, c \in K$ and some Boolean function y . Fix $x_j := c$ and then

$x_i := d$ such that $(x_i \oplus h(c))^a = 0$. Then

$$\text{res}_\alpha(C)(d, c) = (0 \wedge y(d, c)^b)^c = 0^c = (\neg c) \neq f_\alpha(d, c).$$

Elimination step. Change the circuit N . Compute $(g_1 \oplus \dots \oplus g_t)^a$ for some $a \in K$ and connect the result with x_i . This increases the costs by $t-1$. The new graph N' is cycle free because $\neg \exists$ path $(B \Rightarrow [A_2, C'])$. But now $\text{res}_{N'}(C') = \neg a$. Hence delete $[A_2, C']$, decreasing the costs by t . If C is a branching point, this eliminates at least two more gates. If C is an AND-type gate, choose a such that $\text{res}_{N'}(C) = \text{const}$. This eliminates C and a successor which must exist by Lemma 6.

Case IV. $\exists i \in S$: outdegree(input node x_i) = 1 and the edge from x_i goes into a $\oplus$ -type gate g . This case is slightly easier than Case III: treat g and the $\oplus$'s immediately following it like $[A_2, C']$ and don't worry about $[A_1, B']$.

If none of the Cases I-IV are applicable, we cannot eliminate enough gates by decreasing S . But we know: $\forall i \in S$: input node x_i has exactly one outgoing edge. This goes into an AND-type gate. Call this gate G_i . Define $\mathbf{G} = \{G_i | i \in S\}$.

LEMMA 7. $\forall i, j, i \neq j: G_i \neq G_j$.

Proof. The proof follows immediately from Lemma 3. $\square$

DEFINITION. A path $n \rightarrow n_1 \rightarrow \dots \rightarrow n_t \rightarrow n'$ is *free* if $\forall i: n_i \notin \mathbf{G}$. A superscript f in expressions like $[n, n']^f$ will denote that the path, which defines this set of nodes, is free.

LEMMA 8. $\forall i \in S: \exists$ free path $(G_i \Rightarrow t)$.

Proof. Suppose for $i \in S: \neg \exists$ free path $(G_i \Rightarrow t)$. Then all paths $(G_i \Rightarrow t)$ go through a $G_j, j \neq i$ (Fig. 9).

FIG. 9

Fix all variables except x_i such that $(\mathbf{a}_1) = (\mathbf{a}_2) = i$, $q = 1$ and such that by the induced P.A. α , $\text{res}(G_j)_\alpha = \text{const.}$ for all $j \neq i$. Then $f_\alpha = x_i$ but, as there is no free path $(G_i \Rightarrow t)$: $\text{res}(t)_\alpha$ independent of x_i . $\square$

DEFINITION. Let p be a free path ($n \Rightarrow n'$). p splits into a free path to n'' if $\exists C \in [n, n']_p$, $D \notin [n, n']_p \cup \mathbf{G}$, such that there is an edge from C to D and a free path ($D \Rightarrow n''$).

DEFINITION. $\mathbf{E} = \{e | \exists i \text{ such that } e \text{ is node on a free path } (G_i \Rightarrow t), e \notin \mathbf{G}\}$.

Clearly $E \cap G = \emptyset$. If we would show that there are $\#S - 1$ nodes in E where a free path $(G_i \Rightarrow t)$ splits into another free path to t , an application of Lemma 2 to determine the size of E would yield a $3n$ -lower bound. But Fig. 10 shows that it might be the case that no free path $(G_i \Rightarrow t)$ splits into another free path to t .

By placing gates on places like α , whose other input is an appropriate function of the a_i 's and q , one would still be able to compute $x_i \wedge x_j$ as well as $x_i \oplus x_j$ for all i and j . But such gates contribute to the costs. Though we are not able to locate these gates exactly in the circuit, we can find a certain number, say n , of distinct places, where paths to such gates start. Then, we will conclude, there must be at least $n/2$ such gates (as one gate has only two inputs).

The following lemma is the counterpart to Lemma 3.

LEMMA 9. Let x_i, x_j ($i, j \in S$) be such that a free path $(x_i \Rightarrow t)$ and a free path $(x_j \Rightarrow t)$ meet before any of them splits into another free path to t . Let C be the node where these paths meet; then

- (i) either $\exists$ free path $(G_i \Rightarrow G_j)$ or $\exists$ free path $(G_j \Rightarrow G_i)$;
- (ii) each such path goes through a gate g , $g \notin [G_i, C]^f \cup [G_j, C]^f$.

Proof. Consider Fig. 11. First note that by the hypothesis $\neg \exists$ free path $([G_i, C]^f \Rightarrow (G_j, C)^f)$ or vice versa.

FIG. 10

FIG. 11

Fix all variables except x_i, x_j and q such that $(\mathbf{a}_1) = i, (\mathbf{a}_2) = j$ and such that by the induced P.A. α : $\text{res}(G_k)_\alpha = \text{const.}$ for all $k \notin \{i, j\}$.

(i) Both types of paths cannot be present, because then the graph would contain a cycle. Suppose no path $(G_i \Rightarrow G_j)$ and no path $(G_j \Rightarrow G_i)$ is present or all such paths are not free. Then exclude, as in the proof of Lemma 3, the cases $C = \text{AND-type gate}$ and $C = \oplus\text{-type gate}$.

(ii) Without loss of generality $\exists$ free paths $(G_i \Rightarrow G_j)$. Suppose one such path contains no gate $g \notin [G_i, C]^f \cup [G_j, C]^f$. Denote by D the last node this path has in common with the free path $(G_i \Rightarrow C)$ (the first candidate for D is G_i). Then there is an edge e_j from D to G_j . Fix $q := 0$.

As $(\text{res}(t)_\alpha)_{q:=0}$ depends only on x_i and x_j ,

$$(\text{res}(D)_\alpha)_{q:=0} = x_i^a \text{ or } 0 \text{ or } 1.$$

If $(\text{res}(D)_\alpha)_{q:=0} = 0$ or 1 , then

$$(\text{res}(t)_\alpha)_{q:=0} \text{ is independent of } x_i.$$

Hence $(\text{res}(D)_\alpha)_{q:=0} = x_i^a$. Now fix $x_i := c$ such that

$$(\text{res}(G_j)_\alpha)_{\{q:=0, x_i:=c\}}$$

is independent of x_j . Hence $(\text{res}(t)_\alpha)_{\{q:=0, x:=c\}}$ is independent of x_j . But $(f_\alpha)_{\{q:=0, x_i:=c\}} = x_j \oplus c$.

DEFINITION. For $i, j \in S$: iRj : $\Leftrightarrow$ free path $(G_i \Rightarrow t)$ and free path $(G_j \Rightarrow t)$ meet before any of them splits into another free path to t .

LEMMA 10. R is an equivalence relation.

R partitions S into, say, k groups: $S = S_1 \cup \dots \cup S_k$. The free paths $(G_i \Rightarrow t)$ of the group r meet in a point t_r before splitting into other free paths to t . So the free paths $(G_i \Rightarrow t)$ for $i \in S_r$ form a tree with root t_r . Call this tree the *free tree* of group r .

If $S = \{i\}$, then we consider the node G_i as the free tree as well as $G_i = t_r$.

Let $\mathbf{A} = \{g \mid g \text{ is a node in the circuit and } \neg \exists \text{ free path } (g \Rightarrow t)\}$. Clearly by Lemma 8 and by the definition of $\mathbf{E}$:

$$\mathbf{G} \cap \mathbf{A} = \mathbf{E} \cap \mathbf{A} = \emptyset.$$

Let $T = \{i \in S \mid \exists r \text{ such that } i \in S_r \text{ and } \#S_r \geq 2\}$, $k = \# \{r \mid \#S_r \geq 2\}$.

LEMMA 11. $\#\mathbf{A} \geq (\#T - k)/2$.

Proof. We claim: on any free tree with n leaves ($n \geq 2$), there are at least $n - 1$ nodes D such that a free path $(D \Rightarrow \mathbf{A})$ leaves the free tree in D . As for different trees such sets of nodes D are disjoint, summing over all free trees one finds at least $\#T - k$ paths going into $\mathbf{A}$; hence $\#\mathbf{A} \geq (\#T - k)/2$.

To prove the claim, observe that in the free tree of group r there is a node F such that a free path $(G_i \Rightarrow t)$ and free path $(G_j \Rightarrow t)$ meet in F ; $i, j \in S_r$ but $\neg \exists l \in S_r$ such that a free path $(G_l \Rightarrow t)$ goes through F . Apply to i and j Lemma 9. This implies the existence of one node D —without loss of generality, $D \in (G_i, F)^f$ —a gate $g \in [G_i, F]^f \cup [G_j, F]^f$ and a path $(D \Rightarrow g)$. If there was a free path $(g \Rightarrow t)$, then $= \neg i R j$. Hence $g \in \mathbf{A}$.

Now mark $[G_i, F]$ in the free tree and repeat the above argument. As the unmarked paths are disjoint from the marked ones, all new nodes D are disjoint from the old ones.

Repeating this argument $n - 1$ times proves the claim. $\square$

Proof of Theorem 3. Consider free paths $(t_r \Rightarrow t)$ and suppose there is a free path $(t_r \Rightarrow t)$ and a free path $(t_{r'} \Rightarrow t)$ which both don't split into another free path to t . Then there are two cases:

(i) The free path $(t_r \Rightarrow t)$ meets the free path $(t_{r'} \Rightarrow t)$. Then by definition of R and as none of the paths splits into another free path to t , neither t_r nor $t_{r'}$ is root of a free tree.

(ii) There is a free path $(t_r \Rightarrow t_{r'})$ or a free path $(t_{r'} \Rightarrow t_r)$. Then t_r or $t_{r'}$ is not a root of a free tree.

Hence all but one of the free paths $(t_r \Rightarrow t)$ must split into another free path to t .

This implies the existence of a set B' of nodes, in which free paths $(t_r \Rightarrow t)$ split into other free paths to t . Observe that there are $s - \#T + k$ roots t_r of free trees.

As in the proof of Theorem 2 one concludes that these must be $s - \#T + k - 1$ first such nodes on free paths $(t_r \Rightarrow t)$.

Apply Lemma 2 to the graph which consists of input nodes G , interior nodes E , and the free paths between them. By Lemma 7 it has s input nodes. t is the output node. Hence

$$\#\mathbf{E} \geq 2s - \#T + k - 2.$$

Finally by Lemma 7 and 11:

$$L(f) \geq \#G + \#E + \#A \geq 3s - \frac{\#T - k}{2} - 2.$$

As $\#T \leq s$ the theorem follows. $\square$

5. A trade-off result.

DEFINITION. A *tree network* is a circuit with the property that all interior nodes (i.e., which are not input nodes) have outdegree 1.

Tree networks are isomorphic to Boolean formulas. For a switching function f define

$$T(f) = \min \{ \# \text{ interior nodes in a tree network } N \mid N \text{ computes } f \}.$$

T is called the (minimal) formula size. If one allows only AND-like connectives in circuits and tree networks, the minimal formula size of the parity function $x_1 \oplus \cdots \oplus x_n$ is $n^2 - 1$ [5] and the circuit complexity is $3n - 3$ [18]. We prove a similar result for the case in which $\oplus$ -like connectives are also allowed in formulas.

Notation. Let N be a circuit, X the set of input variables and $S \subset X$. A node C in N is called *S-node* if there are nodes C_1, C_2 in N , $C_1 \neq C_2$ (possibly $C_1, C_2 \in S$) such that there is an edge $(C_1 \Rightarrow C)$, an edge $(C_2 \Rightarrow C)$, a path $(S \Rightarrow C_1)$ if $C_1 \notin S$ and a path $(S \Rightarrow C_2)$ if $C_2 \notin S$.

The following lemma is proved by a method very similar to the proof of Lemma 3.

LEMMA 12. Let $f: K^n \rightarrow K$ be a switching function, X the set of input variables, $S \subset X$ and P.A.'s $\alpha_1, \dots, \alpha_M$ which fix all variables in $X \setminus S$ such that

$$f_{\alpha_i} \neq f_{\alpha_j} \quad \text{for } i \neq j \quad (1 \leq i, j \leq M).$$

Let N be a circuit for f . Then

- (i) $P_s := \# \{ C \mid C \text{ is an } S\text{-node in } N \} \geq (\log(M))/4 - 1$.
- (ii) (Neciporuk) If N is a tree network, then

$$\sum_{j \in S} \text{outdegree } x_j \geq \frac{\log(M)}{4} - 1.$$

Proof. (ii) follows from (i) and the definition of a tree network.

(i) Let N' be the subcircuit of N which consists only of the paths $(S \Rightarrow t)$. In N' the nodes with indegree 2 are exactly the S -nodes of N .

Let α be a P.A. which fixes all variables in $X \setminus S$. Then $\text{res}(n)_\alpha = \text{const.}$ for all nodes not in N' .

Let A be an S node or $A \in S$, let B be an S -node or $B = t$, and suppose $\exists$ path $(A \Rightarrow B)$ which contains no other S -node. Let e be the edge on this path which goes into B . We call such a path $(A \Rightarrow B)$ a *programmable edge* of N' . Why this name?

Consider how $\text{res}(e)_\alpha$ depends on $\text{res}(A)_\alpha$ for different P.A.'s α . No matter how many gates are on this path, one can "program"—by choice of α —this

dependence in at most 4 ways:

$$\text{res}(e)_\alpha = \text{res}(A)_\alpha \text{ or } \neg \text{res}(A)_\alpha \text{ or } 0 \text{ or } 1.$$

(Compare with the proof of Lemma 3.)

N' contains at most $2P_s + 1$ programmable edges. Hence there are at most 4^{2P_s+1} different subfunctions f_α which can be “programmed” by different P.A.’s α .

By the hypothesis: $4^{2P_s+1} \geq M$. This proves the lemma. $\square$

We apply Lemma 11 to the function $f: K^n \rightarrow K$, defined as follows: let $n = \log(s) - \log \log(s) + 2s$ ($s = 2^{2^\tau}$, $\tau \in \mathbb{N}$).

Partition the input variables into

$$\{a_1, \dots, a_{\log(s) - \log \log(s)}\}, \{x_1, \dots, x_s\}, \{y_1, \dots, y_s\}.$$

Denote by $\mathbf{a}$ the string

$$a_1, \dots, a_{\log(s) - \log \log(s)}.$$

Denote by $\mathbf{b}$ the string

$$x_{(\mathbf{a}) \log(s) + 1} \dots x_{((\mathbf{a}) + 1) \log(s)}.$$

Define $f(a_1, \dots, y_s) = y_{(\mathbf{b})}$.

THEOREM 4. (i) $T(f) \geq O(n^2 / \log(n))$. (ii) $L(f) = O(n)$.

Proof. (i) For $0 \leq i \leq s / \log(s) - 1$ define

$$S_i = \{x_{i \cdot \log(s) + 1}, \dots, x_{(i+1) \log(s)}\}.$$

Fix $\mathbf{a}$ such that $(\mathbf{a}) = i$ and fix the x_j , $j \notin S_i$, arbitrarily. Now each of the 2^s different assignments for the variables in $y_1, \dots, y_s$ produces a different subfunction of f which depends only on the variables in S_i .

Applying Lemma 12 (ii) yields

$$\forall i \sum_{x_j \in S_i} \text{outdegree}(x_j) \geq \frac{s}{4} - 1.$$

As $S_i \cap S_j = \emptyset$ for $i \neq j$:

$$\begin{aligned} T(f) &\geq \sum_i \sum_{x_j \in S_i} [\text{outdegree}(x_j)] \\ &\geq \left(\frac{s}{4} - 1\right)(s / \log(s)) = O(n^2 / (\log(n))). \end{aligned}$$

(ii) Realize f in a circuit as follows: for $1 \leq k \leq \log(s)$ realize the functions

$$f_k: K^{s / \log(s) + \log(s) - \log \log(s)} \rightarrow K$$

$$f_k(\mathbf{a}, x_k, x_{\log(s) + k}, x_{2 \log(s) + k}, \dots, x_{s - \log(s) + k}) = x_{(\mathbf{a}) \log(s) + k}.$$

By Theorem 1 each f_k can be realized with cost $3 \cdot s / \log(s)$.

Let $\{c_1, \dots, c_{\log(s)}\}$ be a set of new variables. Denote by $\mathbf{c}$ the string $c_1 \dots c_{\log(s)}$. Construct a circuit for

$$f': K^{s + \log(s)} \rightarrow K: f'(\mathbf{c}, y_1, \dots, y_s) = y_{(\mathbf{c})}.$$

By Theorem 1 f' has cost $\leq 3s$. For $1 \leq k \leq \log(s)$ connect the terminal node of the circuit f_k with the input c_k of f' . This gives a realization of f with cost $6s$ and the theorem is proven. $\square$

Acknowledgments. I want to thank Professor John Hopcroft, Dr. Peter van Emde Boas, and Dany Paul for encouragement, help and discussion. Zvi Galil pointed out to me how to improve the original $2.25n$ -lower bound of Theorem 3 [11] into $2.5n$.

REFERENCES

- [1] S. A. COOK, *The complexity of theorem proving procedures*, Proc. 3rd ACM-STOC, 1971, pp. 151–158.
- [2] M. J. FISCHER, *Lectures on network complexity*, presented at the University of Frankfurt, 1974.
- [3] G. HOTZ, *Schaltkreis-theorie*, de Gruyter, Berlin, 1974.
- [4] R. M. KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatchers, eds., Plenum Press, New York, 1972, pp. 85–104.
- [5] KHRAPCHENKO, *On the complexity of the linear function in the class of Π -circuits*, Mat. Zametki, 9 (1971), no. 1, pp. 35–40. (In Russian.)
- [6] O. B. LUPANOV, *Ueber den Schaltaufwand bei der Realisierung logischer Funktionen*, Probleme der Kybernetik, 3 (1957).
- [7] K. MELHORN AND Z. GALIL, *Monotone switching circuits and Boolean matrix product*, preprint, 1974.
- [8] A. MEYER, 6.853 *Lecture notes*, Dept. of Electrical Engineering, Mass. Inst. of Tech., Cambridge, MA, 1974.
- [9] E. I. NECIPORUK, *A Boolean function*, Soviet Math. Probl., 7 (1966), pp. 999–1000.
- [10] M. S. PATERSON, *Complexity of monotone networks for Boolean matrix product*, Tech. Rep., Univ. of Warwick, England, 1974.
- [11] W. PAUL, *$2.25N$ -lower bound on the combinatorial complexity of Boolean functions*, Tech. Rep. TR 74-222, Cornell Univ, Ithaca, NY, 1974.
- [12] V. PRATT, *The power of negative thinking in multiplying Boolean matrices*, 6th Ann. Symp. on Theory of Computing (SIGACT), 1974.
- [13] J. E. SAVAGE, *The complexity of computing*, draft, 1974.
- [14] J. E. SAVAGE AND E. A. LAMAGNA, *Combinational complexity of some monotone functions*, 15th Ann. Symp. on Switching and Automata Theory (SWAT), 1974.
- [15] J. E. SAVAGE, *Computational work and time on finite machines*, J. Assoc. Comp. Mach., 19 (1974), pp. 660–674.
- [16] C. E. SHANNON, Bell System Tech. J., 28 (1949), no. 1.
- [17] C. P. SCHNORR, *Lower bounds for the product of time and space requirements of Turing machine computations*, Proc. MFCS, High Tatras, Poland, Sept. 3–8, 1973, pp. 153–163.
- [18] ———, *Zwei lineare untere Schranken fuer die Komplexitaet Boolescher Funktionen*, Computing, 13 (1974), pp. 155–171.
- [19] ———, *The combinational complexity of equivalence, GI-Fachtagung Automatentheorie und Formale Sprachen, Kaiserslautern*, Springer Lecture Notes, Springer-Verlag, Berlin, 1974.
- [20] L. J. STOCKMEYER, *The complexity of decision problems in Automata theory and Logic*, Project MAC TR 133, Mass. Inst. of Tech, Cambridge, MA, 1974.
- [21] V. STRASSEN, *Berechnung und Programm 1*, Acta Informatica (1972), pp. 320–335.