2.25 N-LOWER BOUND ON THE COMBINATIONAL COMPLEXITY OF BOOLEAN FUNCTIONS

Wolfgang J. Paul+

TR 74-222

December 1974

Computer Science Department
Cornell University
Ithaca, New York 14853

+This research was supported by DAAD Grant No. 430/402/563/5.

2.25 N-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 $\Omega = \{f | f:\{0,1\}^2 \to \{0,1\}\}$. A new Method for proving linear lower bounds of size $2n$ is presented. Combining it with methods presented in [7] and [9], we establish for a special set of functions $f^n:\{0,1\}^n \to \{0,1\}$: $2.25n \leq L(f) \leq 6n$.

+This research was supported by DAAD grant No. 430/402/563/5.

A 2.25 N-LOWER BOUND ON THE COMBINATIONAL COMPLEXITY OF BOOLEAN FUNCTIONS

Wolfgang J. Paul+

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 [1],[8]. However, for this purpose lower bounds of size greater than O(nlog(n)) would be needed.

The only thing known about functions $f:\{0,1\}^n \to \{0,1\}$ with $L(f) > O(n)$ is that they exist [3] and that a Turing machine with tape complexity $> O(n\log(n))$ can find one [10]. Besides that, the best lower bounds known for the basis of the abstract are of size 2n [9] .

We first list some basic notations and facts. In Theorem 1, with known techniques a 2n-lower bound is derived for a function

+This research was supported by DAAD grant No. 430/402/563/5.

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.

In Theorem 3, extension of the known techniques together with the new method are applied to prove a 2.25 n-lower bound. This proof turns out to be considerably more difficult than proofs of 2n-lower bounds.

I want to thank Professor John Hopcroft, Dr. Peter van Emde Boas, and Dany Paul for encouragement, help and discussion.

2. Basic notations and facts

Let $K = \{0,1\}$.

$\Omega = \{f \mid f:K^2 \to K\}$ 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 input node. A node with outdegree 0 is output node. $X_n = \{x_1,\ldots,x_n\}$ is the set of input nodes of N.

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

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

Assigning values 0 or 1 to the variables of a subset $S \subset X_n$ allows, 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 $A = \{x_i:=c_i \mid c_i \in K,\ i \in J \subseteq [1:n]\}$.

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

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

$$[n,n')_p = [n,n']_p \setminus \{n'\}$$

$$(n,n']_p = [n,n']_p \setminus \{n\}$$

$$(n,n')_p = [n,n']_p \setminus \{n,n'\}$$

For sets of nodes S,S' define

$\exists$ path(S => S') $:\Leftrightarrow \exists\, n \in S,\ n' \in S'$: $\exists$ path(n => n')

<u>Example</u>: In figure 1, $\exists$ path(A => C); $\exists$ path([A,C] => [D,F]);

$$\begin{tikzcd}[row sep=small] A \arrow[d] & D \arrow[d] \\ B \arrow[d] \arrow[ur] & E \arrow[d] \\ C & F \end{tikzcd}$$

Figure 1

Denote by $\wedge$, $\oplus$ $\neg$ logical AND, + mod 2, negation.

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:

1) $(\neg a) \oplus b = a \oplus (\neg b) = \neg(a \oplus b)$

2) in an optimal circuit all labellings of nodes are of the form $(y_1^a \wedge y_2^b)^c$ (AND-type gate)
or $(y_1 \oplus y_2)^d$ ($\oplus$ -type gate) $(a,b,c,d \in K)$

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.

3) an optimal circuit for a single function f has exactly one output node t with res(t) = f.

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

4) if one input of a gate is constant, independent of the assignments of values to the input nodes, this gate can be eliminated.

5) in an optimal circuit no two edges, starting from one node, go into the same node, i.e. figure 2 does not occur in an optimal circuit.

figure 2

6) If two functions f and g are equal then $f_A = g_A$ for all P.A.'s A. ∎

In the various coming figures the following symbolism will be used

x_i i'th input node

 AND-type gate

 $\oplus$ -type gate

 any gate

 path

 splitting path, branching point

 edge

 negation

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 costs reading in a storage? On the other hand, it's proof introduces some techniques, which will be used later.

For $\underline{a} = a_1 \ldots a_r \in K^*$ denote by

$(\underline{a})$ = binary number represented by $\underline{a}$

Theorem 1: For $f: K^{n+\log(n)} \to K$

$$f(a_1,\ldots,a_{\log(n)},x_1,\ldots,x_n) = x_{(\underline{a})}:$$
$$2n-2 \leq L(f) \leq 3n$$

Proof: (we prove only the lower bound).

Define for $1 \leq s \leq n$ the statement E_s:

E_s: For any function $f: K^{n+\log n} \to K$ with the property $[\exists\, S \subseteq [1:n],\ \#S = s$ s.t. for all $\underline{a}$ with $(\underline{a}) \in S$, $\mathrm{length}(\underline{a}) = \log(n)$: $f(\underline{a},x_1,\ldots,x_n) = x_{(\underline{a})}]$ 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:

<u>Case 1</u>: $\exists_i \in S$: outdegree(inputnode x_i) ≥ 2. Setting $x_i :=$ const eliminates thus at least two gates. The rest circuit N' after elimination of the gates, conputes 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}. $\blacksquare$

<u>Case 2</u>: $\exists_i \in S$: outdegree(inputnode x_i) $= 1$ and the edge from x_i goes into an AND-type gate C, i.e. $res(C) = (x_i^a \wedge y^b)^c$ with $a,b,c \in K$ and y some function of the other input variables. Claim $C \neq t$. As $s \geq 2$, $\exists j \in S$ s.t. $j \neq i$. Fix the variables in $\underline{a}$ s.t. $(\underline{a}) = j$. Call this P.A. A. By definition of f: $f_A = res(t)_A = x_j$. But setting $x_i := (\neg a)$ yields $(res(C)_A)_{x_i := (\neg a)} = (\neg c)$ independent of x_j. Hence $C \neq t$. Fix $x_i := (\neg a)$. This eliminates C and it's successor. $\blacksquare$

<u>Case 3</u>: $\exists i \in S$: outdegree(inputnode x_i) $= 1$ and the edge from x_i goes into a $\oplus$ -type gate C. For the P.A. A of Case 2: $res(t)_A$ is independent of x_i. But $res(C)_A$ depends on x_i, hence $C \neq t$.

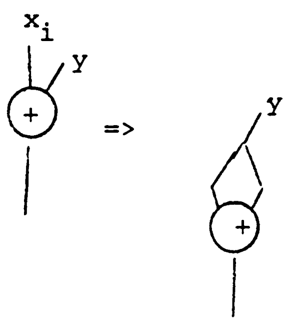


Figure 3

Consider fig. 3. Identify x_i with y. In the new circuit N': $res_{N'}(C') =$ const. This eliminates C and it's successor. $res_{N'}(t)$ 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 theorem. $\blacksquare$

While the argumentation in the proof of Theorem 1 was limited to the top of the circuit, the next proof establishes the existence of branching points, which may be deep in the circuit.

Theorem 2: If for $f:K^n \to K \; \exists \; S \subseteq [1:n]$ s.t.
$[\forall i,j \in S,\ i \neq j \; \exists$ P.A.'s A^1_{ij}, A^2_{ij} which fix all variables except x_i and x_j s.t.

$$f_{A^1_{ij}}(x_i,x_j) = (x_i^a \wedge x_j^b)^c$$

$$f_{A^2_{ij}}(x_i,x_j) = (x_i \oplus x_j)^d \qquad (a,b,c,d \in K)]$$

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

We first prove a Lemma.

Lemma 2: $\forall\, i,j \in S,\ i \neq j$: before a path$(x_i \Rightarrow t)$ meets a path path$(x_j \Rightarrow t)$, one of these paths splits.

Proof: Suppose that it is false for i and j. Let C be the node where the paths meet. Let e_1, e_2 be the input wires of C.

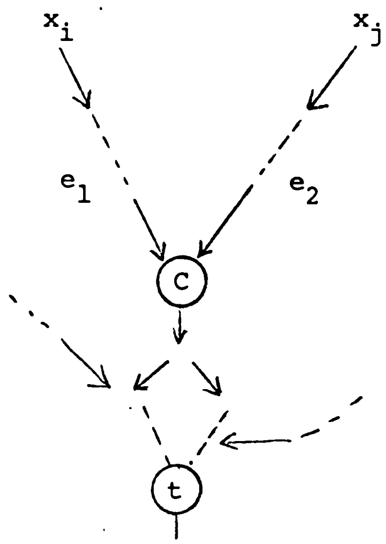


Figure 4

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

$$\mathrm{res}(e_1)_A = x_i^{a_1} \quad \text{or } 0 \text{ or } 1$$

$$\mathrm{res}(e_2)_A = x_j^{a_2} \quad \text{or } 0 \text{ or } 1$$

$$\mathrm{res}(t)_A = \mathrm{res}(C)_A^{a_3} \text{ a } 0 \text{ or } 1.$$

Suppose C is an AND-type gate. Then

$$\mathrm{res}(t)_{A_{ij}^2} = (x_i^{a_4} \wedge x_j^{a_5})^{a_6} \text{ or } 0 \text{ or } 1 \neq f_{A_{ij}^2} .$$

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

<u>Proof of Theorem 2</u>: There are #S-1 input nodes x_i, $i \in S$, such that any path(x_i => t) splits. Suppose that is false. Then there are $i_0, j_0 \in S$ s.t. there is only one path(x_{i_0} => t) and one path(x_{j_0} => t). i_0 and j_0 contradict Lemma 1 . Hence on #S-1 paths(x_i => t) there are <u>first</u> nodes D_i with outdegree(D_i) $\geq$ 2. Again by Lemma 2 these are mutually distinct.

Begin to construct a circuit for f from the input nodes. In the beginning, one has $\geq$ # S "cables hanging down" from the input nodes. The number of down hanging cables is decreased by one gate only by one. By the #S-1 branching points it is increased during the construction by #S-1. In the end it is 1. Hence the circuit contains 2#S-2 gates. ∎

That such branching points must not be directly under the input nodes is shown in Figure 5.

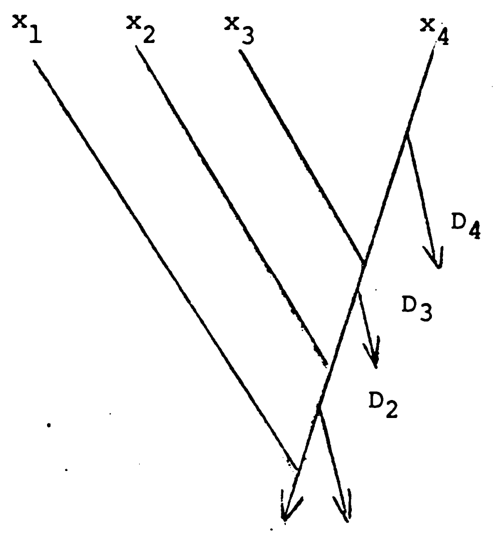


Figure 5

<u>Example</u>: For any $c \in [1{:}n-1]$ and

$$f_c : K^n \to K$$

$$f_c(x_1,\ldots,x_n) = 1 \text{ if } \Sigma x_i = c:$$

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

4. A 2.25 n-lower bound

Notation: $\underline{a_1} = a_1,\dots,a_{\log(n)}$

$\underline{a_2} = a_{\log(n)+1}\dots a_{2\log(n)}$ $\qquad (a_i \in K)$

Theorem 3: For $f: K^{n+2\log(n)+1} \to K$

$$f(a_1,\dots,a_{2\log(n)},q,x_1,\dots,x_n) = \begin{cases} x_{(\underline{a_1})} \wedge x_{(\underline{a_2})} & \text{if } q = 1 \\ x_{(\underline{a_1})} \oplus x_{(\underline{a_2})} & \text{if } q = 0 \end{cases}$$

$$2.25\, n - 2 \le L(f) \le 6n + C \quad (C \in \mathbb{N} \text{ independent of } n)$$

Proof: Note, that both Theorem 1 and Theorem 2 imply 2n-lower bounds for f. As in the proof of Theorem 1, we first make f independent of all inputs x_i, which eliminate 3 gates. When that is no longer possible, we will know quite exactly what the "top" of the circuit looks like. For the remaining s inputs, we prove the existence of $(5s)/2 - 2$ gates without an inductive argument. Define for $1 \le s \le n$ the statement E_s:

E_s: For any function $f: K^{n+2\log(n)+1} \to K$ with the property $[\exists S \subseteq [1:n],\ \#S = s$ s.t. for all $\underline{a_1}$, $\underline{a_2}$ with $(\underline{a_1}),(\underline{a_2}) \in S$:

$$f(\underline{a_1},\underline{a_2},q,x_1,\dots,x_n) = \begin{cases} x_{(\underline{a_1})} \wedge x_{(\underline{a_2})} & \text{if } q = 1 \\ x_{(\underline{a_1})} \oplus x_{(\underline{a_2})} & \text{if } q = 0 \end{cases} \quad]$$

holds: $2.25\, s - 2 \le L(f)$.

E_1 is trivially true.

E_{s-1} implies E_s: Consider an optimal realization of N a function f as defined in E_s.

Case I: $\exists i \in S$ s.t. outdegree (input node x_i) ≥ 3.

Similar to case 1 in the proof of Theorem 1.

Case II: $\exists i \in S$ s.t. outdegree (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.

Almost like case 2 in the proof of Theorem 1.

A small difficulty comes in the case of figure 6.

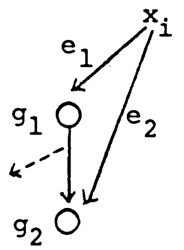


fig. 6

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

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

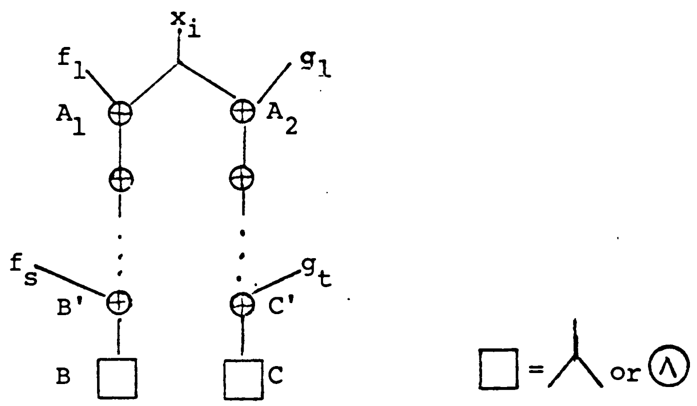


figure 7

By definition of B and C , there is only one path ($A_1 \Rightarrow B$) and only one path ($A_2 \Rightarrow C$). Because of Lemma 1.1 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 3: $[A_1,B'] \cap [A_2,C'] = \phi$

Proof: Suppose not. Then the situation is like that in figure 8.

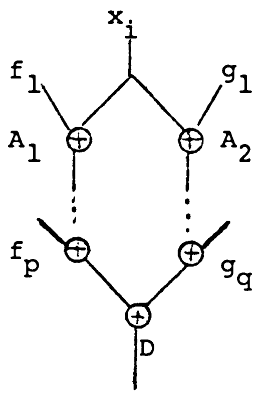


figure 8

Hence $res(D) = f_1 \oplus \ldots \oplus g_q \oplus x_i \oplus x_i$
$= f_1 \oplus \ldots \oplus g_q$

As no other path goes out of $[A_1,D)$ and $[A_2,D)$, $f_1,\ldots,g_q$ are independent of x_i. Hence res(D) is independent of x_i. By the same reasoning, res(t) depends on x_i only via res(D). Hence 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 3. ■

Lemma 4: 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 w.l.o.g. only C exists:

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

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

If only C exists, the existence of such a path implies the existence of B. ■

From now on w.l.o.g. C exists and if B exists, then

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

Lemma 5: $C \neq t$.

Suppose $C = t$. Then C is an AND-type gate and $\exists\, \text{path}(A_2 \Rightarrow C)$. This implies the existence of B. Hence if B does not exist, the Lemma is true. If B exists, proceed as

Since $\#S \geq 2$, $\exists j \neq i$, $j \in S$. Fix $\underline{a_1}$, $\underline{a_2}$ s.t. $(\underline{a_1}) = (\underline{a_2}) = j$, $q = 1$. Fix all other variables except x_i and x_j arbitrarily. Call this P.A. A. Then $f_A = x_j$ independent of x_i.

Case 1: $C = B$. Then C is an AND-type gate and $(f_1)_A, \ldots, (g_t)_A$ depend only on x_j. Hence changing x_i changes $res(B')_A$ and $res(C')_A$ simultaneously. Changing x_j can have two effects: either it changes $res(B')_A$ and $res(C')_A$ simultaneously or it changes, w.l.o.g. only $res(C')_A$, and $res(B')_A$ depends only on x_i. The case, that both $res(C')_A$ and $res(B')_A$ depend only on x_i, is trivial.

In the first case change x_i to $(\neg x_i)$ and x_j to $(\neg x_j)$. Then

$$res(t)_A\ (\neg x_i, \neg x_j) = res(C)_A\ (\neg x_i, \neg x_j) = res(C)_A\ (x_i, x_j)$$

as neither $res(B')_A$ nor $res(C')_A$ have changed.

But $f_A(\neg x_i, \neg x_j) = \neg f_A(x_i, x_j)$, hence $C \neq t$.

In the second case, fix x_i (and hence $res(B')_A$) such that $res(C)_A$ becomes constant independent of x_j.

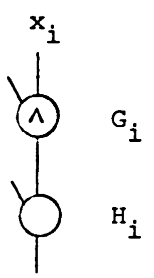
Case 2: $t = C \neq B$. As $\neg\exists$ path $(B \Rightarrow [A_2,C'])$, $(g_1)_A,\dots,(g_s)_A$ are independent of x_i, hence $res(C')_A$ depends on x_i. Let $(res(C))_A = (res(C')^a_A \wedge y^b)^c$ for some function y. Fix $x_j := 0$. Then fix $x_i = c_0$ s.t. $(res(C')_A)(c_0,0) = \neg a$, hence $(res(C)_A)(c_0,0) = \neg c$. Now change $x_j := 1$. Again adjust $x_i := c$ (not necessarily $\neq c_0$), s.t. $(res(C')_A(c_1,1) = \neg a$. Hence $res(t)_A(c_0,0) = res(t)_A(c_1,1)$, i.e. $res(t) \neq f$. ■

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 $res_{N'}(C') = \neg a$. 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 s.t. $res_{N'}(C) = const$. This eliminates C and it's (by Lemma 5 existing) successor. ■

Case IV: $\exists i \in S$: outdegree(inputnode 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 $[A_1,B']$. ■

Case V: $\exists i \in S$: outdegree(inputnode x_i) = 1, the edge from x_i goes into an AND-type gate g and outdegree(g) $\geq$ 2. Fix $x_i := c$ such that $res(g)_{x_i:=c}$ becomes constant. This eliminates g and it's successors. ∎

If none of the Cases I-V are applicable, we cannot eliminate enough gates by decreasing S. But we know: $\forall i \in S$: Inputnode x_i has exactly one outgoing edge. This goes into an AND-type gate. Call this gate G_i. G_i has exactly one outgoing edge which goes into a gate H_i (Fig. 9).



Define $\underline{G} = \{G_i | i \in S\}$

$\underline{H} = \{H_i | i \in S\}$

Figure 9

Lemma 6: $\forall i,j,\ i \neq j$: $G_i \neq G_j$, $H_i \neq H_j$, $H_i \neq G_j$.

Proof: As no path($x_i \Rightarrow t$) splits before H_i, assuming any such equality contradicts Lemma 2. ∎

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

Lemma 7: $\forall i \in S$: $\exists$ 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, $j \neq i$ (Figure 10).

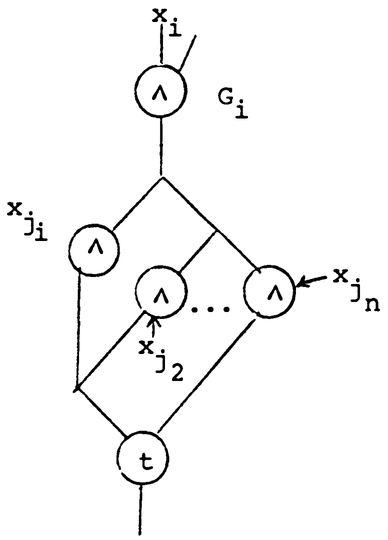


Figure 10

Fix all variables except x_i s.t. $(\underline{a}_1) = (\underline{a}_2) = i$, $q = 1$ and s.t. by the induced P.A. A: $\mathrm{res}(G_j)_A$ = const for all $j \neq i$. Then $f_A = x_i$ but, as there is no free path($G_i \Rightarrow t$): $\mathrm{res}(t)_A$ independent of x_i. ∎

Now it is seemingly easy to prove the lower bound. One has the G_i's, the H_i's, the free paths and the branching points of Lemma 2. But Fig. 11 shows, that we are not yet done.

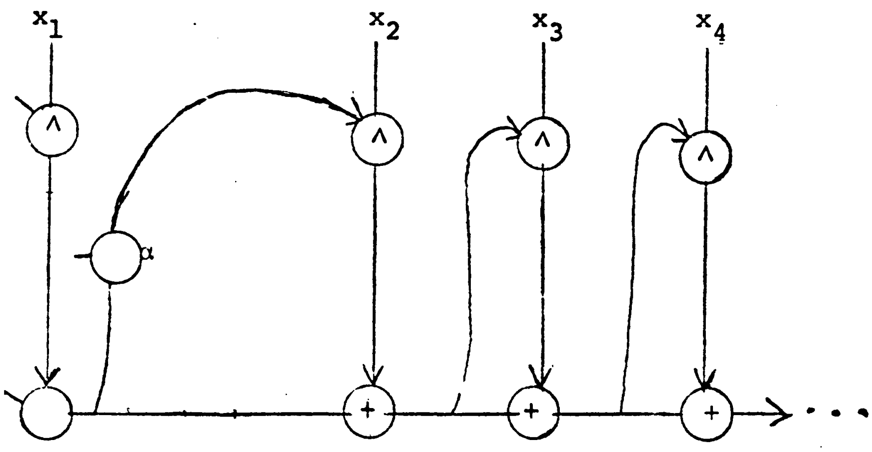


Figure 11

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 $\lceil n/2 \rceil$ such gates (as one gate has only two inputs).

<u>Definition</u>: A free path(n => n') splits into a free path to n" if on that path there is a node C (possibly C = n)) with outdegree (C) $\geq$ 2 and $\exists$ free path(C => n").

The following lemma is the counterpart to Lemma 2.

Lemma 8: Let x_i, x_j $(i,j \in S)$ be s.t., free paths$(x_i \Rightarrow t)$ and free paths$(x_j \Rightarrow t)$ meet before any of them splits into another free path to t. Let C be the node, where these paths meet, then

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

Proof: Consider fig. 12. First note that by the hypothesis $\neg \exists$ free path$([G_i, C)^f \Rightarrow (G_j, C)^f)$ or vice versa.

Fix all variables except x_i, x_j and q s.t. $(\underline{a}_1) = i$, $(\underline{a}_2) = j$ and s.t. by the induced P.A. A: $res(G_k)_A$ = const for all $k \notin \{i,j\}$.

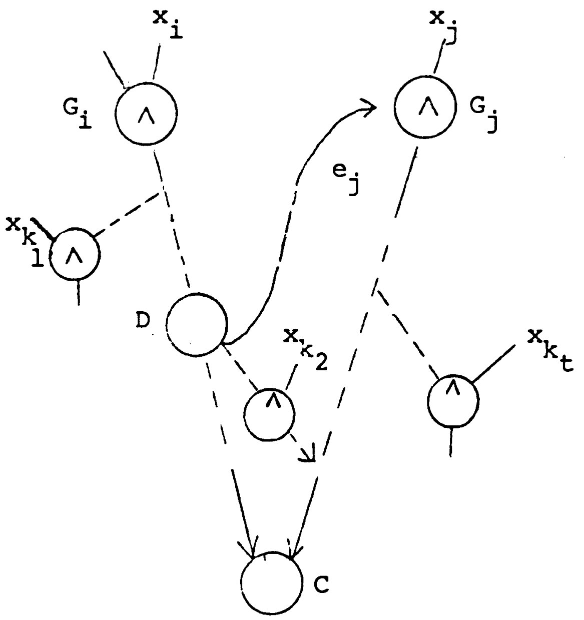


Figure 12

1) 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 2, the cases C = AND-type gate and $C = \oplus$-type gate.

2) W.l.o.g. $\exists$ 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 H_i). Then there is an edge e_j from D to G_j. Fix $q := 0$.

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

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

If $(\mathrm{res}(D)_A)_{q:=0} = 0$ or 1, then $(\mathrm{res}(t)_A)_{q:=0}$ is independent of x_i.

$$\text{Hence } (\mathrm{res}(D)_A)_{q:=0} = x_i^a .$$

Now fix $x_i := c$ such that $(\mathrm{res}(G_j)_A)_{\{q:=0,x_i':=c\}}$ independent of x_j. Hence $(\mathrm{res}(t)_A)_{\{q:=0,x:=c\}}$ independent of x_j. But $(f_A)_{\{q:=0,x_i:=c\}} = x_j \oplus c$. ∎

<u>Definition</u>: for $i, j \in S$: $i \, R \, j$: free path ($G_i \Longrightarrow t$) and free path ($G_j \Longrightarrow t$) meet before any of them splits into another free path to t.

<u>Lemma 9</u>: R is equivalence relation. ∎

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

If $S_r = \{i\}$, then we consider the edge $(G_i \Rightarrow H_i)$ as the free tree of group r and H_i as t_r.

Let $\alpha = \{g \mid g$ is a node in the circuit and $\neg\exists$ free path $(g \Rightarrow t)\}$. Clearly by Lemmas 6 and 7:

$$\underline{G} \cap \alpha = \underline{H} \cap \alpha = \emptyset.$$

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

<u>Lemma 10</u>: $\#\alpha \geq (\#T-k)/2$.

<u>Proof</u>: 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 $(D \Rightarrow \alpha)$ leaves the free tree in D. As for different trees such sets of nodes D are disjoint, there are at least #T-k paths going into α ,hence $\#\alpha \geq (\#T-k)/2$.

To prove the claim, observe that in the free tree of group r there is a node E such that a free path $(G_i \Rightarrow t)$ and a free path $(G_j \Rightarrow t)$ meet in E; $i,j \in S_r$ but $\neg\exists k \in S_r$ s.t. a free path $(G_k \Rightarrow t)$ goes through E. Apply to i and j Lemma 8. This implies the existence of one node D-- w.l.o.g., $D \in (G_i,E)^{f}$, a gate $g \notin (G_i,E)^f \cup (G_j,E)^f$, and a path $(D \Rightarrow g)$. If there was a free path $(g \Rightarrow t)$, then $\neg iRj$. Hence $g \in \alpha$.

Now mark (G_i,E) in the free tree and repeat the above argument for the unmarked part of the tree. As all 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. ■

Now consider free paths $(t_r \Rightarrow t)$ and suppose there are 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 node C s.t. there is a free path($t_r \Rightarrow C$) and a free path($C \Rightarrow t_{r'}$).Then t_r is not root of a free tree.

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

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

As in the proof of Theorem 2, one concludes that there must be $s - \#T+k-1$ first such nodes on free paths$(t_r \Rightarrow t)$. Call this set of nodes B. Partition $B = B_0 \dot{\cup} B_1 \dot{\cup} B_2$ by

$b \in B_2$ if there are at least two free paths$(b \Rightarrow t)$, none of which goes through an H_i (as they are free, they cannot go through a G_i).

$b \in B_1$ if there is exactly one free path$(b \Rightarrow t)$, which goes not through an H_i. (This implies that at least one free path$(b \Rightarrow t)$ must go through an H_i).

$b \in B_0$ if all free paths$(b \Rightarrow t)$ -- there are at least two -- go through H_i's.

This implies again, as in the proof of Theorem 2, the existence of $2\#B_2+\#B_1-1$ gates which are neither in $\underline{G}$ nor in $\underline{H}$ nor in α. Hence

$$L(f) \geq \#\underline{G} + \#\underline{H} + \#\alpha + 2\#B_2 + \#B_1 - 1$$
$$\geq 2s+(\#T-k)/2+2\#B_2 + \#B_1 - 1. \quad (1)$$

As there are only s gates H_i, one input of which is already occupied by the branches from G_i,

$$2\#B_0 + \#B_1 \leq s \quad (2)$$

Finally,

$$\#B_0 + \#B_1 + \#B_2 = s - \#T+k-1 \quad (3)$$
$$\#B_0 \geq 0,\ \#B_1 \geq 0,\ \#B_2 \geq 0 \quad (4)$$

Applying (3) to (1) yields

$$L(f) \geq 2s + \frac{1}{2}\#s + \frac{3}{2}\#B_2 + \frac{1}{2}\#B_1 - \frac{1}{2}\#B_0 - \frac{3}{2}.$$

Minimizing this under the constraints of (2) and (4) yields:

$$\#B_1 = \#B_2 = 0,\ \#B_0 = s/2\ \text{, and } L(f) \geq (2 + 1/4)s - \frac{3}{2}.$$

This proves the theorem. ∎

References

[1] Fischer, M.J., "Lectures on Network Complexity", presented at the University of Frankfort, 1974.

[2] Hotz, G., Schaltkreistheorie, de Gruyter Lehrbuch, 1974.

[3] Lupanov, O.B. Ueber den Schaltanfwand bei der Realisierung logischer Funktionen. Probleme der Kybernetik, Band 3, 1957.

[4] Paterson, M.S., Complexity of monotone networks for boolean matrix product. Technical report, University of Warwick, 1974.

[5] Pratt, V., The power of negative thinking in multiplying boolean matrices. 6th Annual Symposium on Theory of Computing (SIGACT), 1974.

[6] Savage, J.E. and E.A. Lamagna, "Combinational coplexity of some monotone functions", 15th Annual Symposium on Switching and Automata Theory (SWAT), 1974.

[7] Savage, J.E., "The Complexity of Computing", Draft, 1974.

[8] Schnorr, C.P., "Lower Bounds for the Product of Time and Space Requirements of Turing Machine Computations", in Mathematical Foundations of Computer Science. Proceedings of symposium and summer school High Tatras, September 3-8, 1973. Mathematical Institute of the Slovak Academy of Sciences, Computing Research Centre. United Nations D.P. Bratislava.

[9] Schnorr, C.P., "Zwei lineare untere Schranken fuer die Komplexitaet Boolescher Funktionen". Preprint Fachbereich Mathematik, Universitaet Frankfurt/M, 1973.

[10] Stockmeyer, L.J., The complexity of decision problems in automata theory and logic. MAC TR 133, 1974.

[11] Strassen, V., "Berechnung und Programm 1. Acta Informatica 320-335 (1972)