Theoretical Computer Science 28 (1984) 337–345
North-Holland

NOTE

A BOOLEAN FUNCTION REQUIRING 3*n* NETWORK SIZE

Norbert BLUM
Fachbereich 10, Universität des Saarlandes, D-6600 Saarbrücken, Fed. Rep. Germany

Communicated by M.S. Paterson
Received September 1982
Revised July 1983

Abstract. Paul (1977) has proved a $2.5n$-lower bound for the network complexity of an explicit Boolean function. We modify the definition of Paul's function slightly and prove a $3n$-lower bound for the network complexity of that function.

1. Introduction

One of the most difficult problems in complexity theory is proving a nonlinear lower bound for the network complexity of an explicit Boolean function. Although it is well known by a counting argument that relative to the full basis most Boolean functions need exponentially many operations, only linear lower bounds with small constant factor are known for explicit Boolean functions. Schnorr [3] first proved a $2n$-lower bound for an n-ary Boolean function. Next, Paul [2] proved a $2.5n$-lower bound for another n-ary Boolean function. Stockmeyer [5] proved that the lower bound of Paul holds for a larger class of functions. In [4] Schnorr gives a 'proof' for a $3n$-lower bound for the function defined by Paul, but Wegener [6] pointed out a gap in the proof of a lemma. In [1] we use a weaker version of that lemma and prove a $2.75n$-lower bound. Here we modify the definition of Paul's function slightly and prove a $3n$-lower bound.

2. Preliminaries

Let $K=\{0,1\}$ and $F_n=\{f \mid f: K^n \to K\}$. F_2 is the set of *basic operations*. $x_i: K^n \to K$ denotes the ith *variable*. Let $V_n=\{x_i \mid 1 \leq i \leq n\}$.

Definition. A *network* β is a directed, acyclic graph satisfying the following conditions:

(1) Each node has indegree 0 or 2.

(2) The nodes v with indegree 0 are the *input nodes* of β and are labelled with a variable $\mathrm{op}(v) \in V_n$.

(3) Each node u with indegree 2 is called a *gate* and is labelled with an $\mathrm{op}(u) \in F_2$. The edges entering u are associated in a fixed ordered way with the arguments of $\mathrm{op}(u) \in F_2$.

With each node v we associate a function $\mathrm{res}_\beta(v) : K^n \to K$ as follows:

$$\mathrm{res}_\beta(v) = \begin{cases} \mathrm{op}(v) & \text{if } v \text{ is an input node,} \\ \mathrm{res}_\beta(u)\ \mathrm{op}(v)\ \mathrm{res}_\beta(w) & \text{otherwise,} \\ & \text{where } u, w \text{ are the predecessors} \\ & \text{of } v \text{ in that order.} \end{cases}$$

The network β *computes* all functions $f \in F_n$ such that there exists a node $v \in \beta$ with $\mathrm{res}_\beta(v) = f$. $\mathrm{Res}_\beta(v)$ *depends on* input variable x_i if and only if there exists an $(a_1, \ldots, a_i, \ldots, a_n)$ such that

$$\mathrm{res}_\beta(v)(a_1, \ldots a_i, \ldots, a_n) \neq \mathrm{res}_\beta(v)(a_1, \ldots, \neg a_i, \ldots, a_n).$$

$C(f)$ denotes the network complexity of the function f, i.e., $C(f)$ is the minimal number of gates which is necessary for computing f.

For $f \in F_n$ and $a \in K$ let

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

We say that $f \in F_2$ is $\wedge$-type, if

$$\exists a, b, c \in K : f(x, y) = (x^a \wedge y^b)^c;$$

$f \in F_2$ is $\oplus$-type, if

$$\exists a \in K : f(x, y) = (x \oplus y)^a.$$

A node $v \in \beta$ such that $\mathrm{op}(v)$ is $\wedge$-type ($\oplus$-type) is called an $\wedge$-type gate ($\oplus$-type gate).

The functions $f \in F_2$ can be classified in the following way. There are

(i) 2 constant functions,

(ii) 4 functions depending on one variable,

(iii) 10 functions depending on two variables, 8 of these functions are $\wedge$-type and 2 are $\oplus$-type.

For a node v in β let

$$\mathrm{suc}(v) = \{u \mid v \to u \text{ is an edge in } \beta\}$$

and

$$\mathrm{pred}(v) = \{u \mid u \to v \text{ is an edge in } \beta\}$$

be the set of direct successors and direct predecessors of v.

The functions, associated with the nodes in pred(v), are called input functions of v. Throughout this paper, we use the following fact.

Fact. *Let β be a network computing $f \in F_n$. Let $v \in \beta$ be an $\wedge$-type gate or a $\oplus$-type gate. If one input function of v is constant, then we can eliminate gate v and the reduced network still computes f.*

Let $U \subset V_n$ and $\alpha: U \to K$ be a mapping. Frequently we consider the restriction f_α of $f \in F_n$ under the assignment α. More precisely, f_α is defined by

$$f_\alpha(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \quad \text{with } y_i = \begin{cases} \alpha(x_i) & \text{if } x_i \in U, \\ x_i & \text{if } x_i \notin U. \end{cases}$$

In a natural way an assignment α associates with a network β a subnetwork β_α which is derived by fixing input variables according to α and eliminating the unnecessary gates.

In the sequel we write res(v) for res$_\beta(v)$ if β is kept fixed. $\# S$ denotes the cardinality of set S.

For proving the lower bound, we consider paths in a network. $(v \Rightarrow u)$ denotes a path from node v to node u. $(v \Rightarrow u[$ denotes a path from node v to a node in pred(u).

3. The lower bound

For $a = a_1 \cdots a_r \in K^*$, let (a) denote the binary number represented by $a + 1$. Let

$$a_1 = a_1 \cdots a_{\log n}, \qquad a_2 = a_{\log n+1} \cdots a_{2\log n}$$

and

$$a_3 = a_{2\log n+1} \cdots a_{3\log n} \quad (a_i \in K).$$

Then we define

$$f: K^{n+3\log n+1} \to K,$$

$$f(a_1, \ldots, a_{3\log n}, r, x_1, \ldots x_n) := x^r_{(a_1)} \wedge (x_{(a_2)} \oplus x_{(a_3)}).$$

Remark. If $r = 1$ and $x_{(a_1)} = 1$, then $f(a_1, a_2, a_3, r, x_1, \ldots, x_n) = x_{(a_2)} \oplus x_{(a_3)}$.
If $r = 1$ and $x_{(a_3)} =$ [illegible], then $f(a_1, a_2, a_3, r, x_1, \ldots, x_n) = x_{(a_1)} \wedge x_{(a_2)}$.

Paul has defined the following function:

$$h: K^{n+2\log n+1} \to K,$$

$$h(a_1, \ldots, a_{2\log n}, q, x_1, \ldots, x_n) = q(x_{(a_1)} \wedge x_{(a_2)}) \vee \neg q(x_{(a_1)} \oplus x_{(a_2)}).$$

The function f is similar to the function h.

For h, Paul has proved a $2.5n$-lower bound. First he makes h independent of some inputs x_i, which allows him to eliminate 3 gates each. After this, he knows quite exactly, how the 'top' of the network looks. For the remaining s inputs, he proves without an inductive argument the existence of $\frac{5}{2}s-2$ gates.

Theorem 3.1. *For the f defined above, there holds the following*:

$$C(f) \geq 3n-3.$$

Proof. First we make f independent of some inputs x_i which allows the elimination of 3 gates each. We use for this the entire proof of Paul and only give a sketch of this part. For a more detailed analysis, see [2].

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

$$E_s: C(f) \geq 3s-3 \quad \text{for any function } f: K^{n+3\log n+1} \to K$$

with the property:

$$[\exists S \subseteq \{1, \ldots, n\} \quad \# S = s \text{ such that for } a_1, a_2, a_3 \text{ with}$$

$$(a_1), (a_2), (a_3) \in S: f(a_1, a_2, a_3, r, x_1, \ldots, x_n) = x^r_{(a_1)} \wedge (x_{(a_2)} \oplus x_{(a_3)})].$$

E_1 is trivially true. Let E_{s-1} be true. Now we prove that E_{s-1} implies E_s. Let β be any minimal size network for f. Without loss of generality we assume that for each $i \in S$ there is a unique node $v \in \beta$ with $\mathrm{op}(v) = x_i$.

Case 1. $\exists i \in S$ such that $\# \mathrm{suc}(x_i) \geq 3$.

By fixing x_i at 0 we can eliminate at least 3 gates of β. The reduced network computes the restriction $f_{x_i=0}$ of f. From the induction hypothesis it follows that $C(f) \geq 3s-3$.

Case 2. $\exists i \in S$ such that $\# \mathrm{suc}(x_i) = 2$ and $\exists v \in \mathrm{suc}(x_i)$ such that v is an $\wedge$-type gate.

Choose $c \in K$ such that $\mathrm{res}(v)_{x_i=c}$ is constant. Then, by fixing x_i at c, we can eliminate all nodes in $\mathrm{suc}(x_i)$ and all nodes in $\mathrm{suc}(v)$. Since β is of minimal size, there are at least three such different nodes. The reduced network computes the restriction $f_{x_i=c}$ of f. From the induction hypothesis it follows that $C(f) \geq 3s-3$.

Case 3. $\exists i \in S$ such that, $\forall v \in \mathrm{suc}(x_i)$, v is a $\oplus$-type gate. Then there exist nodes $u_1, \ldots, u_r$ in β with

(1) $u_1 \in \mathrm{suc}(x_i)$,

(2) u_j is a $\oplus$-type gate $\forall j \in \{1, \ldots, r\}$,

(3) $u_{j+1} \in \mathrm{suc}(u_j)$ and $\# \mathrm{suc}(u_j) = 1$ for $1 \leq j \leq r-1$,

(4) $\# \mathrm{suc}(u_r) > 1$ or for $w \in \mathrm{suc}(u_r)$ holds: w is an $\wedge$-type gate.

Let x_i, g_1 be the input functions of u_1 and $\mathrm{res}_\beta(u_j), g_{j+1}$ be the input functions of u_{j+1}, $1 \leq j < r$. Paul [2, Case III] proves that $u_1, \ldots, u_r$ can be chosen such that there is no path from x_i to the nodes computing $g_1, \ldots, g_r$. Then $\mathrm{res}(u_r) = x_i \oplus g$ for the function $g = g_1 \oplus \cdots \oplus g_r$ which does not depend on x_i.

Hence $\mathrm{res}(u_r)_{x_i:=g}$ and $\mathrm{res}(u_r)_{x_i:=\neg g}$ are constant. Therefore, for each of the substitutions $x_i := g$ and $x_i := \neg g$, we can eliminate u_r and all nodes in $\mathrm{suc}(u_r)$. g can be computed with $r-1$ extra gates. We distinguish two cases:

(i) $\#\mathrm{suc}(u_r) \geq 2$. Then we eliminate at least $r+2$ gates by fixing x_i at g or at $\neg g$.

(ii) $\#\mathrm{suc}(u_r) = 1$. Then $w \in \mathrm{suc}(u_r)$ is an $\wedge$-type gate. Choose $\tilde{g} \in \{g, \neg g\}$ such that $\mathrm{res}(w)_{x_i:=\tilde{g}}$ is constant. Then by fixing x_i at $\tilde{g}$, we can eliminate at least $r+2$ gates, namely u_r, w and all nodes in $\mathrm{suc}(w)$.

The induction hypothesis now implies $C(f) \geq 3s-3$.

If none of Cases 1–3 applies, then $\forall i \in S$ the following holds:

(i) $\#\mathrm{suc}(x_i) = 1$. We denote the node in $\mathrm{suc}(x_i)$ by G_i.

(ii) G_i is an $\wedge$-type gate.

Let $G = \{G_i \mid i \in S\}$. Paul proves the following lemma.

Lemma 3.2. *$\forall i, j \in S$ with $i \neq j$, $G_i \neq G_j$.*

Proof. Suppose $\exists i, j \in S$, $i \neq j$, with $G_i = G_j$. Then there exists a $c \in K$ such that $\mathrm{res}(G_j)_{x_i:=c}$ is constant. Hence $f_{x_i:=c}$ does not depend on x_j. But $f(a_1, a_2, a_3, r, x_1, \ldots, x_n) = c \oplus x_j$ for $(a_1) \neq i, j$, $r = 1$, $x_{(a_1)} = 1$, $(a_2) = i$, $x_i = c$ and $(a_3) = j$ depends on x_j, a contradiction.

Case 4. $\exists i \in S$ such that $\#\mathrm{suc}(G_i) \geq 2$.

Consider $c \in K$ with $\mathrm{res}(G_i)_{x_i:=c}$ is constant. Then fixing x_i at c eliminates G_i and all nodes in $\mathrm{suc}(G_i)$. There are at least three different such gates. Hence from the induction hypothesis it follows that $C(f) \geq 3s-3$.

It remains to consider the following case:

Case 5. $\forall i \in S$ we have that $\#\mathrm{suc}(G_i) = 1$.

We denote the unique direct successor of G_i by Q_i. Before analysing Case 5, we give some definitions.

A path $(v \Rightarrow w)$ in β is called *free*, if no node $\neq v, w$ is in G.

A node w in β is called a *split*, if the outdegree of w is ≥ 2.

Let t be the node of β with $\mathrm{res}(t) = f$. A split w in β is called a *free split*, if $\exists u_1, u_2 \in \beta$, $u_1 \neq u_2$, such that

(a) $u_1, u_2 \in \mathrm{suc}(w)$,

(b) $\exists$ free paths $(u_1 \Rightarrow t)$ and $(u_2 \Rightarrow t)$ in β.

A node w is called the *collector* of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$, $i \neq j$, if w is the first node which lies on both paths.

The next four lemmas are due to Paul, except the observation that C is a $\oplus$-type gate in Lemma 3.4, which is due to Schnorr.

Lemma 3.3. *$\forall i \in S$, $\exists$ free path $(G_i \Rightarrow t)$.*

Proof. Suppose that no free path $(G_i \Rightarrow t)$ exists. Then each path $(G_i \Rightarrow t)$ passes some G_j with $j \neq i$. Construct the assignment α by fixing all variables except x_i, such that

(i) $\mathrm{res}(G_\nu)_\alpha$ is constant $\forall \nu \in S\backslash\{i\}$,

(ii) $f_\alpha = x_i^a$, $a \in K$.

Since each path $(G_i \Rightarrow t)$ goes through some G_ν with $\nu \neq i$, $\mathrm{res}(t)_\alpha$ does not depen on x_i. But this is a contradiction to $f_\alpha = x_i^a$ and $\mathrm{res}(t)_\alpha = f_\alpha$.

Note that from Lemma 3.3 it follows that $G \cap Q = \emptyset$.

Lemma 3.4. *Let $i, j \in S$, $i \neq j$. Let C be the collector of a free path $(G_i \Rightarrow t)$ and* *free path $(G_j \Rightarrow t)$.*

Let

$$\nexists \textit{ free split } \neq C \textit{ on path } (G_i \Rightarrow C)$$

and

$$\nexists \textit{ free split } \neq C \textit{ on path } (G_j \Rightarrow C).$$

Then the following holds:

(i) *C is a $\oplus$-type gate, and*

(ii) *$\exists$ free path $(G_i \Rightarrow G_j)$, or $\exists$ free path $(G_j \Rightarrow G_i)$.*

Proof. Suppose, the assertion does not hold. We distinguish two cases.

(a) C is an $\oplus$-type gate.

Construct assignment α by fixing all variables except x_i, x_j such that

(i) $\mathrm{res}(G_\nu)_\alpha$ is constant $\forall \nu \in S\backslash\{i, j\}$,

(ii) $f_\alpha = x_i \wedge x_j^a$, $a \in K$.

By assumption, all paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$ go through C or a G_ν, $\nu \in S\backslash\{i, j\}$. Thus $\mathrm{res}(C)_\alpha = (x_i \oplus x_j)^b$, $b \in K$, or $\mathrm{res}(C)_\alpha$ depends on at most one variable, and so the same holds for $\mathrm{res}(t)_\alpha$. Hence $f_\alpha \neq \mathrm{res}(t)_\alpha$, a contradiction.

(b) C is an $\wedge$-type gate.

Construct assignment α by fixing all variables except x_i, x_j such that

(i) $\mathrm{res}(G_\nu)_\alpha$ is constant $\forall \nu \in S\backslash\{i, j\}$,

(ii) $f_\alpha = x_i \oplus x_j$.

(Here, we need the control variable r for forcing that such an assignment α exists.)

If $\mathrm{res}(G_j)_\alpha$ depends on x_i, choose $c \in K$ such that $\mathrm{res}(G_j)_{\alpha, x_i := c}$ is constant. Hence $\mathrm{res}(t)_{\alpha, x_i := c}$ does not depend on x_j, but $f_{\alpha, x_i := c} = c \oplus x_j$ depends on x_j, a contradiction. The case where $\mathrm{res}(G_i)_\alpha$ depends on x_j is symmetric.

Now, by assumption, all paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$ go through C or a G_ν, $\nu \in S\backslash\{i, j\}$. Since $\mathrm{res}(C)_\alpha = (x_i^a \wedge x_j^b)^c$, $a, b, c \in K$ or $\mathrm{res}(C)_\alpha$ depends on at most one variable; for $\mathrm{res}(t)_\alpha$ the same holds.

Hence $f_\alpha \neq \mathrm{res}(t)_\alpha$, a contradiction.

From Lemma 3.4 we immediately have the following.

Lemma 3.5. $\forall i, j \in S$ *if $i \neq j$ holds, then $Q_i \neq Q_j$.*

Now we have isolated $2s$ gates, namely the gates G_i, Q_i for $i \in S$. Let $Q = \{Q_i \mid i \in S\}$.

Lemma 3.6. *There are at least $s-1$ mutually distinct splits in β.*

Proof. Let $S' = S$. Choose $i, j \in S'$ with free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$ such that for every $l \in S' \setminus \{i, j\}$ no free path $(G_l \Rightarrow t)$ goes through the collector C of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$.

From Lemma 3.4 follows that one of the paths $(G_i \Rightarrow C[$ and $(G_j \Rightarrow C[$ respectively, splits into a free path to the output node t or splits into a free path to the node G_j and G_i, respectively. Without loss of generality let the path $(G_i \Rightarrow C[$ split. Set $S' = S' - \{i\}$. Note, that by construction no path $(G_l \Rightarrow t)$, $l \in S'$, has a common node with the path $(G_i \Rightarrow C[$.

Repeating this argument $s-1$ times proves the lemma.

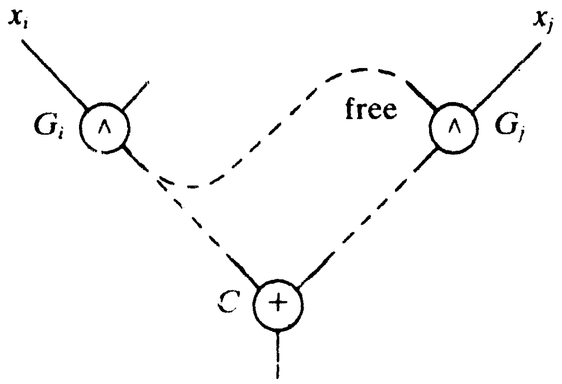
The rest of the proof is new.

Since we have at least $s-1$ splits, we have to connect at least $2(s-1)$ edges with the output node t. For edges, which correspond to a free path, no node in G can help to connect these with the output node t on the free paths by the definition of a free path. Next we prove, that all but one of the $s-1$ splits from Lemma 3.6 have to be free.

Assume, less than $s-1$ splits, isolated in the proof of Lemma 3.6, are free. Then by Lemma 3.4 there exist i, j, $i \neq j$, with the following property:

Let C be the collector of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$. Then there is no free split on the path $(G_i \Rightarrow C[$ and no free split on the path $(G_j \Rightarrow C[$. There is a free path $(G_i \Rightarrow G_j)$ and C is a $\oplus$-type gate.

Then we have the following situation:



emma 3.7. *$\forall \nu \in S \setminus \{j\}$, $\exists$ free path $(G_\nu \Rightarrow G_i)$ or $\exists$ free path $(G_\nu \Rightarrow G_j)$.*

roof. For i we know by assumption that $\exists$ free path $(G_i \Rightarrow G_j)$. Assume $\exists l \in S \setminus \{i, j\}$ ith

$$\nexists \text{free path } (G_l \Rightarrow G_i) \quad \text{and} \quad \nexists \text{free path } (G_l \Rightarrow G_j).$$

Now we construct assignment α by fixing all variables except x_i, x_j, x_l such that

(a) $\text{res}(G_\nu)_\alpha$ is constant $\forall \nu \in S\backslash\{i, j, l\}$,

(b) $f_\alpha = x_j \wedge (x_i \oplus x_l)$.

We distinguish two cases:

Case 1. $\text{res}(G_j)_\alpha$ does not depend on x_i. Now fix x_l at $a \in K$ such that $\text{res}(G_l)_{\alpha, x_l := a}$ is constant. Then $f_{\alpha, x_l := a} = x_j \wedge x_i^b$, $b \in K$.

Now, as in the proof of Lemma 3.4, we show that Case 1 cannot happen.

Case 2. $\text{res}(G_j)_\alpha$ depends on x_i.

Hence $\text{res}(G_j)_\alpha = (x_i^c \wedge x_j^d)^e$, $c, d, e \in K$. Fix x_i at $\neg c$. Then $\text{res}(G_j)_{\alpha, x_i := \neg c}$ is constant and hence $\text{res}(t)_{\alpha, x_i := \neg c}$ does not depend on x_j. But

$$f_{\alpha, x_i := \neg c} = x_j \wedge (\neg c \oplus x_l) = x_j \wedge x_l^c$$

depends on x_j, a contradiction.

Now we prove that all other splits, which are isolated in the proof of Lemma 3.6, have to be free.

Lemma 3.8. *$\forall l, \nu \in S\backslash\{j\}$, $\nu \neq l$, if D is the collector of a free path $(G_l \Rightarrow t)$ and a free path $(G_\nu \Rightarrow t)$, then*

$$\exists \text{ free split} \neq D \text{ on path } (G_l \Rightarrow D)$$

or

$$\exists \text{ free split} \neq D \text{ on path } (G_\nu \Rightarrow D).$$

Proof. Assume that $\nexists$ free split on path $(G_l \Rightarrow D)$ and $\nexists$ free split on path $(G_\nu \Rightarrow D)$. Then applying Lemma 3.7 to l, ν there exists a path $(G_j \Rightarrow G_l)$ or there exists a path $(G_j \Rightarrow G_\nu)$. But by construction, there exist paths $(G_l \Rightarrow G_j)$ and $(G_\nu \Rightarrow G_j)$ and, hence, we have a cycle in the network. But this cannot happen by the definition of a network.

From Lemma 3.8 we can directly derive the following.

Lemma 3.9. *There are at least $s-2$ mutually distinct free splits in β.*

By Lemma 3.9 and Lemma 3.3 we have to connect at least $2(s-2)+2$ edges on free paths to the output node t. Since the paths are free, the nodes in G cannot help to connect these edges to the output node. For the nodes in Q only one input wire is free for connecting these edges. There are s nodes in Q. Hence at least $s-2$ edges have to connect to the output node using new nodes. One new node (except the output node) can decrease the number of edges only by one. The output node can decrease the number of edges only by two. Hence we need at least $s-3$ new gates.

Hence

$$C(f) \geq \# G + \# Q + s - 3 = 3s - 3.$$

This finishes the proof of the theorem.

Acknowledgment

I would like to thank Kurt Mehlhorn and Mike Paterson for valuable comments. Mike Paterson pointed out that the proof also works for the function f which is much simpler than a previously used function.

References

[1] N. Blum, A 2.75n-lower bound on the network complexity of boolean functions, Tech. Rept. A 81/05, Universität des Saarlandes, 1981.
[2] W.J. Paul, A 2.5n-lower bound on the combinational complexity of boolean functions, *SIAM J. Comput.* **6** (1977) 427–443.
[3] C.P. Schnorr, Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen, *Computing* **13** (1974) 155–171.
[4] C.P. Schnorr, A 3n-lower bound on the network complexity of boolean functions, *Theoret. Comput. Sci.* **10** (1980) 83–92.
[5] L.J. Stockmeyer, On the combinational complexity of certain symmetric Boolean functions, *Math. Systems Theory* **10** (1977) 323–336.
[6] I. Wegener, Private communication, 1981.