Astérisque

MICHAEL S. PATERSON

An introduction to boolean function complexity

Astérisque, tome 38-39 (1976), p. 183-201

<http://www.numdam.org/item?id=AST_1976__38-39__183_0>

Société Mathématique de France
Astérisque 38-39 (1976) p.183-201

AN INTRODUCTION TO BOOLEAN FUNCTION COMPLEXITY

by

Michael S. Paterson
(University of Warwick, Coventry, U. K.)

Abstract.

The "complexity" of a finite Boolean function may be defined with respect to its computation by networks of logical elements in a variety of ways. The three complexities of "circuit size", "formula size" and "depth" are considered, and some of the principal results concerning their relationships and estimations are presented, with outlined proofs for some of the simpler theorems. This survey is ruthlessly restricted to networks in which all two-argument logical functions may be used. A rich corpus of theory related to logical networks under a variety of restrictions may be found in the literature, but is apt to be confusing in a first introduction.

Keywords and phrases: Boolean functions, complexity, logical networks, finite functions, formula size, depth.

CR categories: 5.21, 5.25

This paper was prepared while the author was a visitor at Stanford University Computer Science Department, and was supported by National Science Foundation grant MCS72-03752 A03, by Office of Naval Research grant N00014-76-C-0330 and by the IBM Corporation.

1. Introduction.

My purpose in composing this brief account is to introduce the general mathematical reader to some of the results and problems concerned with the complexity analysis of Boolean functions. In the interests of conciseness and coherence of presentation I shall attempt to cover just a few restricted areas which I have found to be of particular theoretical interest.

The study of Boolean function complexity draws its importance from several branches of computer science. The original and most obvious motivation is that many of the tasks for which digital electronic equipment must be designed can be usefully represented as the computation of Boolean functions. As examples, I have in mind sorting networks, unary-to-binary converters, multiplication units and address decoders. A second catchment area lies in the recently active field of algebraic algorithmic complexity. Attractively structured problems such as matrix multiplication, polynomial evaluation and convolution product have their simplest incarnation over the two-element Boolean domain. The aim is to reach a complete understanding of the complexity of such basic algorithms. Finally I should mention "machine-based" complexity where we are concerned with time or space bounds on the behaviour of Turing machines, random-access machines or other abstractions of digital computers. For example, a Turing machine accepting or rejecting an input string may be so simulated by a Boolean network computing a function that lower bounds on the complexity of such a function yield corresponding bounds on the running time of the Turing machine [20,23]. A proof that $P \neq NP$, see [8], is in principle feasible by such an approach.

2. Definitions.

Let B_n be the set of n-argument Boolean functions $\{f: \{0,1\}^n \to \{0,1\}\}$. (The correspondence with the familiar Boolean domain {false, true} is that 1 represents true.) We note that $|B_n| = 2^{2^n}$, so, for example, $|B_2| = 16$. To introduce our notations and terminology for these 16 basic functions we list them in Table 1. To obviate explicit function tables, definitions in terms of GF(2), the two-element field, are provided.

Symbol for f	Name for f	f(x,y)
0	constants	0
1		1
π_1	projections	x
π_2		y
$\bar{\pi}_1$	--	1+x
$\bar{\pi}_2$	--	1+y
$\wedge$, AND	conjunction	x.y
$\bar{\wedge}$, NAND	nand	1+x.y
$\vee$, OR	disjunction	x+y+x.y
$\bar{\vee}$, NOR	nor	(1+x).(1+y)
$\rightarrow$	implication	1+x+x.y
$\leftarrow$	--	1+y+x.y
$\bar{\rightarrow}$	--	x.(1+y)
$\bar{\leftarrow}$	--	y.(1+x)
$\neq$, $\oplus$	not-equivalence	x+y
$\equiv$	equivalence, bi-implication	1+x+y

The 16 functions of B_2 with GF(2) equivalents.

Table 1.

Functions in B_n are to be computed by <u>acyclic circuits over the basis</u> B_2 . These may be represented as finite directed acyclic graphs with n <u>input nodes</u> and one <u>output node</u>, each input node corresponding with one of the arguments and each other node being associated with some element of B_2 . The indegree of the input nodes is zero and each other node has an ordered pair of incoming arcs. An association of binary

values to the inputs naturally induces binary values on all other nodes (by applying the appropriate basis function at each to the values of its predecessors) and hence the circuit defines a function of B_n computed at its output node. A small example is given in Figure 2.

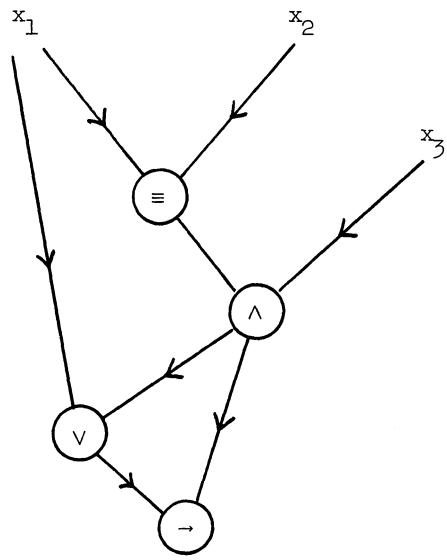

Figure 2

An alternative formulation which is in many respects equivalent is as a straight-line program, a fixed sequence of computation steps at each step of which a basic function is applied to two arguments which may be either results of previous steps or input argument values. There are in general many such sequences represented by a single acyclic circuit. One that yields Figure 2 is given below.

$$
\begin{aligned}
v_1 &:= x_1 \equiv x_2 \\
v_2 &:= v_1 \wedge x_3 \\
v_3 &:= x_1 \vee v_2 \\
\text{output} &:= v_3 \rightarrow v_2
\end{aligned}
$$

Various parameters of circuits may be used as a basis for complexity measures. The most immediate is the circuit size, c, which counts the number of internal nodes or logical gates, and corresponds also to the number of steps in a program. If each gate in a circuit requires the same fixed execution time then the total time in a parallel computation by the circuit will be limited by the depth, d, of the circuit, the maximum number of gates on a path from an input to the output node. In the example, $c = 4$ and $d = 3$.

A mathematician might prefer to represent a Boolean function as a well-formed linear expression over the input variables with function symbols corresponding to elements of B_2. This is equivalent to an acyclic circuit where the logical gates have fanout (outdegree) at most one. Note that the input nodes may have arbitrary fanout, but it is convenient in diagrams to replicate inputs so that the circuit can be drawn as a tree in closer correspondence with the structure of the linear formula. The size of a formula is just the circuit size, the number of internal nodes. A formula of size 6 which "computes" the same function as in Figure 2 is represented in Figure 3.

$$[x_1 \vee ((x_1 \equiv x_2) \wedge x_3)] \rightarrow [(x_1 \equiv x_2) \wedge x_3]$$

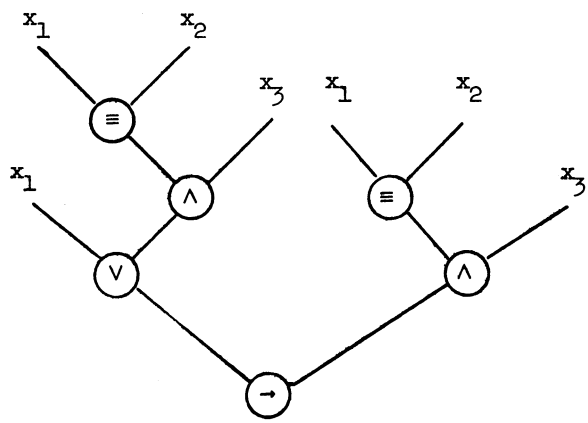

Figure 3

Each of the three measures we have described induces a corresponding complexity measure over B_n in a natural way. For any f in B_n

$$\text{circuit size} = c(f) = \min\{c(\alpha) \mid \alpha \text{ is a circuit for } f\}$$

$$\text{formula size} = \ell(f) = \min\{c(\alpha) \mid \alpha \text{ is a } \underline{\text{formula}} \text{ for } f\}$$

$$\text{depth} = d(f) = \min\{d(\alpha) \mid \alpha \text{ is a circuit for } f\}$$

Since in the example considered, the function $g(x_1,x_2,x_3)$ can be shown to have an equivalent representation as $x_1 \wedge x_2 \wedge x_3$, we have $c(g) = \ell(g) = d(g) = 2$.

In this paper we shall consider only measures defined with respect to the full basis B_2. There is however a considerable literature concerned with other bases containing sometimes functions of more than two arguments or maybe consisting of a particular subset of B_2 appropriate to some technology or application. A useful survey and bibliography for these results can be found in [22].

3. Relationships Among Complexity Measures.

Fortunately the three different measures we have defined are not entirely independent. In this section we summarize the known inter-relationships. Two of these are immediate.

Lemma 1. For all f in B_n,

$$c(f) \leq \ell(f) < 2^{d(f)} .$$

Proof. For the first inequality it is enough to recall that a formula is a restricted form of circuit. The second follows from the observation that for any acyclic circuit an equivalent formula with the same depth can be constructed by repeatedly duplicating nodes of the circuit until the unit fanout restriction is satisfied. Furthermore any binary tree with depth d has at most 2^d-1 internal nodes. □

These inequalities are the best possible of their type since for any n consider the function $CONJ^{(n)}$ in B_n defined by

$$CONJ^{(n)}(x_1,\ldots,x_n) = \bigwedge_{i=1}^{2^p} x_i$$

where $p = \lfloor \log n \rfloor$. (All logarithms in this paper are to base 2 . The notation $\lfloor x \rfloor$ denotes the greatest integer not more than x .) It is evident that for $CONJ^{(n)}$

$$c = \ell = 2^d - 1 = 2^p - 1 \quad .$$

For inequalities in the reverse directions we have no such complete results. For ℓ and d , a technique of Spira [27] shows

$$d \lesssim \alpha \log \ell \quad \text{where} \quad \alpha = 2/\log(3/2) \doteq 3.42 \quad .$$

(We use $f \lesssim g$ to mean $\limsup f/g \leq 1$.) Spira misstates his coefficient as $2 \log 3$. A small refinement improves this coefficient to about 2.465 . Thus d and $\log \ell$ are asymptotically within a constant multiple of each other. A recent result of Paterson and Valiant [16] relates c and d by

$$c \gtrsim \frac{1}{4} d \log d \quad .$$

For each of the above results a construction is given for a circuit of relatively small depth equivalent to a given formula or circuit.

4. Global Bounds.

Although determining the complexity of particular functions seems usually rather difficult, there are surprisingly precise results on the asymptotic complexities of "most" functions. If we write $c(S)$ for $\max\{c(f) \mid f \in S\}$ and similarly for d and ℓ , then uniform constructions for all n yield

$$c(B_n) \lesssim 2^n/n \quad , \quad \ell(B_n) \lesssim 2^n/\log n \quad , \quad d(B_n) \leq n+1 \quad .$$

The first two constructions are due to Lupanov [11,12]. The third result, by McColl and Paterson [13] improves only a little on a simple construction

by Spira [28]. A remarkable feature of the results for c and ℓ is that they are matched asymptotically by lower bounds for "almost all" functions. Counting arguments due to Shannon [26] and Riordan and Shannon [21] respectively can be used to show that for all n, there is a subset $B_n^* \subset B_n$ with $|B_n^*| \sim |B_n|$ such that for all f in B_n^*

$$c(f) \gtrsim 2^n/n \quad , \quad \ell(f) \gtrsim 2^n/\log n \quad .$$

Using Lemma 1, we have also

$$d(f) \geq n - \log\log n + O(1) \quad .$$

To illustrate the form of such counting arguments we outline a proof of the first inequality. It is sufficient to prove the following result.

Lemma 2. For any $\epsilon > 0$, the number of functions of B_n such that $c(f) \leq (1-\epsilon)2^n/n$ is $o(2^{2^n})$.

Proof. We first estimate the number of circuits with n inputs and m gates where the gates are labelled with integers $1,\ldots,m$. A circuit is specified when for each gate the associated function and the origins of its two arguments are given. An upper bound is therefore

$$(n+m)^{2m} \cdot 16^m \quad .$$

It avails us little to reduce the constant 16. However we are interested only in minimal size circuits for some function and this consideration simplifies our task. Firstly there will be just one gate with fanout 0, the output gate. Secondly, no two gates will compute the same function of the inputs, for if otherwise then one of them could be eliminated reducing the circuit size. Each minimal circuit appears exactly $m!$ times in this enumeration since two different labellings must indeed give different labelled circuits. Further each circuit is minimal for precisely one function. The number of distinct functions computed by circuits of size at most M is therefore no more than

$$\sum_{m=0}^{M} \frac{(n+m)^{2m}}{m!} \cdot 16^m \quad .$$

With $M \leq (1-\epsilon) \cdot 2^n/n$, this quantity is bounded above asymptotically by

$$2^{(1-\epsilon)2^n}$$

which accounts for a vanishingly small fraction of B_n . □

Pippenger surveys and generalizes some of the classical results of this section in [19].

5. Lower Bounds for Particular Functions.

One of the most frustrating yet tantalizing aspects of Boolean function complexity is revealed in this section. As we have seen above, nearly all functions have circuit complexity which grows exponentially with the number of arguments. It would be satisfying to be able to present here a simple, explicitly given, function with exponential complexity. The only functions with such complexity known to date involve some kind of diagonalization in their definitions or incorporate the totality of Boolean functions over a slightly smaller set of arguments. Ehrenfeucht [2], and Stockmeyer and Meyer [30] give examples of such functions.

If we restrict ourselves to "natural" functions which avoid all taint of diagonalization the present predicament is extreme. The only lower bounds known for such functions are linear in the number of arguments. In particular, lower bounds asymptotic to $2n$ have been proved for some broad families of functions in B_n by Schnorr [24]. More recently Paul has shown bounds asymptotic to $2\frac{1}{2}n$ [17] and his result has been generalized to a wider class by Stockmeyer [29]. The latter proves this lower bound for the very simple congruence functions $C_m^{(n)}$ for all $m > 2$, defined by

$$C_m^{(n)}(x_1,\ldots,x_n) = 1 \quad \text{if } \sum x_i \equiv 0 \pmod m$$

$$= 0 \quad \text{otherwise,}$$

and matches this for C_4 with the same asymptotic upper bound! The methods of Paul and Stockmeyer are too complicated to follow here, but the flavour of a $2n$ lower bound proof can be given in the simple example

of T_2 . The threshhold functions $T_m^{(n)}$ are defined by

$$T_m^{(n)}(x_1,\ldots,x_n) = 1 \quad \text{if} \quad \Sigma x_i \geq m$$

$$= 0 \quad \text{otherwise.}$$

Theorem 1.

$$c(T_2^{(n)}) \geq 2n-3 \quad \text{for} \quad n \geq 2 .$$

Proof. For $n = 2$, the result is obvious. Suppose $n > 2$ and the result is true for all smaller values. Consider a minimal circuit for $T_2^{(n)}$ with m gates, and select a gate g at maximal distance from the output node. The arguments to this gate must be (distinct) input nodes, x_i , x_j say. If x_i and x_j both have fanout 1 , the dependence of the output on x_i , x_j is only through the value of g . This is absurd since for certain values of the other arguments the circuit must distinguish among three possible values for $x_i + x_j$, namely $0, 1, 2$. We may therefore suppose that x_i has fanout at least two and is connected to two distinct gates g and h . If we fix the value of input x_i to 0 then g and h compute functions of only one argument. These are trivial enough to be absorbed into the functions at the succeeding nodes, eliminating g and h . The resulting circuit has $m-2$ gates and certainly computes $T_2^{(n-1)}$ from the remaining arguments. By induction we have

$$m-2 \geq 2(n-1)-3 \quad , \quad \text{i.e.,} \quad m \geq 2n-3 \quad . \quad \square$$

Other (lower) linear bounds are given by Harper, Hsieh and Savage [4]. These and related results are surveyed by Savage in [22].

Any non-linear lower bound for the circuit size of an explicitly given function would constitute an important advance from our present expertise. To prove $P \neq NP$ by this route would require a non-polynomial lower bound for some suitable function. Some slight progress has been made for the formula size measure and this will be outlined in the next sections. There are as yet no non-trivial lower bounds on depth other than those derived directly for corresponding formula size results using the relation $d > \log \ell$.

6. Lower Bounds on Formula Size.

An important theorem here is due to Neciporuk [15]. Suppose that the arguments to a function f in B_n are partitioned into blocks $R_1, \ldots, R_p$. If for some i the arguments in all the blocks R_j, $j \neq i$, are fixed to 0 or 1 in some way, the result is a restriction of f, a function f' depending only on the variables in R_i. Let m_i be the number of different such restrictions f' for all possible fixations of the other variables. Now the theorem can be stated simply as follows.

Theorem 2 (Neciporuk). There exists $a > 0$ such that for all f,

$$\ell(f) \geq a \cdot \sum_{i=1}^{p} \log m_i$$

where the m_i's are as defined above. □

To explore the maximum possible lower bounds derivable from this theorem we note that if R_i contains r variables then there are on the one hand at most 2^{2^r} possible functions on R_i, and on the other at most 2^{n-r} fixations of the remaining variables. Hence

$$m_i \leq \min\{2^{2^r}, 2^{n-r}\}$$

and the optimum bound, which requires r to be about $\log n$, is of order $n^2/\log n$.

The variety of applications of Neciporuk's theorem is illustrated by the following examples. The full bound of $a \cdot n^2/\log n$ for some $a > 0$ is provable for Neciporuk's original functions [15] and for functions defined by Paul [17]. In both cases the examples involve some notion of "indirect addressing", for instance Paul uses functions of the form

$$f(\underset{\sim}{x}, \underset{\sim}{y}_1, \ldots, \underset{\sim}{y}_k, \underset{\sim}{z}) = z_{\underset{\sim}{y}_{\underset{\sim}{x}}}$$

where $\underset{\sim}{x}$ and the $\underset{\sim}{y}_i$'s are binary vectors of length s, and where $\underset{\sim}{z}$ is a binary vector of length $k = 2^s$. To compute the value of the function, the vector $\underset{\sim}{x}$ is regarded as a binary index to select the

vector $y_{\underset{\sim}{x}}$ which is used similarly to select one binary digit of $\underset{\sim}{z}$. Neciporuk's example can be slightly modified and both upper and lower bounds of order $n^2/\log n$ proved for the result.

More algebraic in nature are the examples of determinant over GF(2) by Kloss [9] and the "stable marriage problem" (exact matching) by Harper and Savage [5]. The lower bounds proved in these cases are only $a \cdot n^{3/2}$. Finally we have unpublished results from two entirely different areas. There is a context-free language over a binary alphabet so that the n-ary function defined by the strings in the language of length n has formula size of order at least $n^2/\log n$. The topological predicate of connectedness on a square binary array yields a function with formula size at least $a \cdot n \cdot \log n$.

We close this section by mentioning two similar theorems giving non-linear lower bounds on formula size. The first is due to Hodes and Specker [7] and has been applied by Hodes to geometric predicates such as convexity and connectedness [6]. The second is a result of Fischer, Meyer and Paterson, a weaker version of which appears in [3]. Both theorems can be roughly expressed as follows.

"Theorem" (X). For all f in B_n either $\ell(f)$ is X-large or there is an X-restriction of f to m variables which is X-linear. □

When X = Hodes - Specker, an X-restriction is made by setting the remaining variables to 0 and an X-linear function is of the form

$$g(x_1, \ldots, x_m) = b_0 \oplus (b_1 \wedge \bigwedge_i \bar{x}_i) \oplus (b_2 \wedge \bigoplus_i x_i)$$

where b_0, b_1, b_2 are Boolean constants and overline denotes negation.

When X = Fischer - Meyer - Paterson, an X-restriction is made by setting equal numbers of variables to 0 and 1, and an X-linear function has the form

$$g(x_1, \ldots, x_m) = b_0 \oplus \bigoplus_i (b_i \wedge x_i)$$

for some constants $b_0, \ldots, b_m$.

In both cases X-large is defined in terms of n and m (the number of variables of the restriction). The largest bounds provable with the first theorem are less than $n \log^* n$ where

$$\log^* n = \text{least } m \text{ such that } \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\}m \geq n \quad .$$

The merit of the second theorem lies in its capacity to prove bounds up to $n \log n / \log \log n$. We shall return to these theorems in the next section where their specializations to symmetric functions are more succinctly expressible.

7. Symmetric Functions.

A symmetric function is one which is invariant under permutations of its arguments, or equivalently, a function $f(x_1,\ldots,x_n)$ is symmetric if and only if there is a function g such that

$$f(x_1,\ldots,x_n) = g\left(\sum_i x_i\right) \quad .$$

There are precisely 2^{n+1} symmetric functions in B_n since $\sum x_i$ can take $n+1$ different values. We denote the set of symmetric functions in B_n by S_n.

A much lower range of complexities is involved here.

Theorem 3. $c(S_n)$ is linear in n, $\ell(S_n)$ is bounded by a polynomial in n, and $d(S_n)$ is $O(\log n)$.

Proof. Each bound results from a two-stage construction. In the first stage a circuit is designed to compute the binary representation of the sum $\sum x_i$. This set of $\lceil \log(n+1) \rceil = p$ functions can be computed either with a circuit of size $O(n)$ or in depth $O(\log n)$, by a recursive splitting process in which representations for the two halves of the argument set are computed and then added together. The addition of two p-digit binary numbers can be performed by a circuit of size $O(p)$ in a straightforward way. For the depth bound a signed-digit representation [1]

can be used so that an addition requires depth only $O(\log p)$. A binary representation is not used until the final result.

The second stage has only to compute the required function from the p results of the first stage. The results given in Section 4 show that this stage requires either only about $2^p/p = O(n/\log n)$ gates or only depth $p+1$. The upper bound on formula size follows from that for depth. □

Detailed constructions for the first stage are provided by Muller and Preparata [14]. A polynomial upper bound for $\ell(S_n)$ is proved by Krapchenko [10]. The best bound on formula size published to date is $O(n^{3.56\ldots})$ and due to Pippenger [18].

The results of Schnorr [24,25] show that for each $n > 2$ all except eight functions have size complexity at least $2n-3$. The eight remaining functions have complexity $n-1$ or 1 . Stockmeyer shows in [29] that at least half of S_n has complexity about $2\frac{1}{2}n$. He also states that $c(S_n) \leq 6n$.

Directing our attention again to formula size we find that Neciporuk's theorem is relatively impotent for symmetric functions since for a block of size r the number of restrictions is limited to $\min\{2^{r+1}, n-r+1\}$ so that only linear lower bounds are derivable.

When the theorem of Hodes and Specker is restricted to symmetric functions it can be restated more dramatically.

Theorem 4. For some (slowly growing) function $t(n)$ with $t \to \infty$ as $n \to \infty$, for all f in S_n

$$\text{either } \ell(f) > n \cdot t(n) \text{ or } \ell(f) < 2n \ .$$

Proof. The only symmetric functions which escape the conditions sufficient for a non-linear lower bound are functions of the form

$$(b_0 \wedge \bigwedge x_i) \oplus (b_1 \wedge \bigvee x_i) \oplus (b_2 \wedge \bigoplus x_i) \oplus b_3 \ .$$

For all n , each of these 16 functions has a formula of size at most $2n-1$. □

The only limitation therefore to the power of the Hodes - Specker theorem for symmetric functions is the lowness of the bound. A rather better bound is attainable for many symmetric functions using the Fischer - Meyer - Paterson result. The corresponding simplification to S_n is as follows.

Theorem 5. For some $a > 0$, all sufficiently large n and for all f in S_n we have, for all k

$$\text{either} \quad \ell(f) > a \cdot n \cdot \log k / \log\log k$$

$$\text{or} \quad f \text{ is a function only of } \bigoplus_i x_i \text{ in the range}$$

$$k \leq \sum_i x_i \leq n-k \; . \quad \square$$

Whereas in the previous theorem a function escaped the lower bound only if it was constant or alternating except possibly in the "end zones" of size one of the sum function, in this second result the "end zones" are of size k. To produce a bound of order $n \cdot \log n / \log\log n$ we need to ensure that $k > n^{\epsilon}$ for some $\epsilon > 0$.

For the families of threshhold functions $T_k^{(n)}$ and congruence functions $C_k^{(n)}$ defined in Section 5 we may establish the following results as corollaries of Theorems 5 and 6.

(i) For all $k \geq 2$, $\ell(T_k^{(n)})/n \to \infty$ as $n \to \infty$.

(ii) For all $\epsilon > 0$, there is a constant $a > 0$ such that

$$\ell(T_k^{(n)}) > a \cdot n \cdot \log n / \log\log n \quad \text{for} \quad n^{\epsilon} < k < n-n^{\epsilon} \; ,$$

and

$$\ell(C_k^{(n)}) > a \cdot n \cdot \log n / \log\log n \quad \text{for} \quad 2 < k < n-n^{\epsilon} \; .$$

The functions $C_4^{(n)}$ are of special interest again since one can construct formulae of order $n \cdot \log n$ for these, to approach the proven lower bound rather closely.

8. Conclusion and Open Problems.

A multitude of problems of practical and theoretical interest can be expressed in terms of the complexity of Boolean functions. In recent years a substantial body of new results in this area has been attained. There remain however embarrassingly large gaps in our knowledge and proof techniques. This can best be appreciated in considering the following set of simply stateable open problems.

1. Prove a non-linear lower bound on the circuit size of some explicitly given Boolean functions.
2. Prove a quadratic lower bound on the formula size of explicit functions.
3. Improve the general inequality $c \gtrsim a \cdot d \cdot \log d$.
4. Prove an asymptotic depth bound not derivable from a corresponding bound on formula size.
5. Show $\ell(S_n) > a \cdot n \cdot \log n$ for some $a > 0$.

References

[1] A. Avizienis. "Signed-digit number representation for fast parallel arithmetic," IRE Trans EC-10, 9, 389-400.

[2] A. Ehrenfeucht. "Practical decidability," Report CU-CS-008-72 University of Colorado (1972).

[3] M. J. Fischer, A. R. Meyer and M. S. Paterson. "Lower bounds on the size of Boolean formulas: preliminary report," Proc. 7th Ann. ACM Symp. on Th. of Computing (1975), 45-49.

[4] L. H. Harper, W. N. Hsieh and J. E. Savage. "A class of Boolean functions with linear combinational complexity," Theoretical Computer Science, 1, 2, 161-183.

[5] L. H. Harper and J. E. Savage. "On the complexity of the marriage problem," Advances in Mathematics 9, 3 (1972), 299-312.

[6] L. Hodes. "The logical complexity of geometric properties in the plane," J. ACM 17, 2 (1970), 339-347.

[7] L. Hodes and E. Specker. "Lengths of formulas and elimination of quantifiers I," in Contributions to Mathematical Logic, K. Schutte, ed., North Holland Publ. Co., (1968), 175-188.

[8] R. M. Karp. "Reducibility among combinatorial problems," in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York (1972), 85-103.

[9] B. M. Kloss. "Estimates of the complexity of solutions of systems of linear equations," Eng. trans. in Soviet Math Dokl. 7, 6 (1966), 1537-1540; orig. Dokl. Akad. Nauk SSSR, 171, 4, 781-783.

[10] V. M. Krapchenko. "The complexity of the realization of symmetrical functions by formulae," English translation in Math. Notes of the Academy of Sciences of the USSR (1972), 70-76; orig. in Matem. Zametki 11, 1, 109-120.

[11] O. B. Lupanov. "Ob odnom metode sinteza skhem," Izv. VUZ (Radiofizika) 1, 1 (1958), 120-140.

[12] O. B. Lupanov. "Complexity of formula realization of functions of logical algebra," Prob. Cyb. 3, (1962), 782-811; orig. Prob. Kibernetiki 3, 61-80.

[13] W. F. McColl and M. S. Paterson. "The depth of all Boolean functions," Th. of Computation Report 7, Univ. of Warwick (1975). To appear in SIAM J. on Computing.

[14] D. E. Muller and F. P. Preparata. "Bounds to complexities of networks for sorting and switching," J. ACM 22, 2 (1975), 195-201.

[15] E. I. Neciporuk. "A Boolean function," Soviet Math. Dokl. 7, 4 (1966), 999-1000, orig. Dokl. Akad. Nauk SSSR 169, 4, 765-766.

[16] M. S. Paterson and L. G. Valiant. "Circuit size is nonlinear in depth," Th. of Computation Report 8, Univ. of Warwick (1975). To appear in Theoretical Computer Science.

[17] W. Paul. "A 2.5 N lower bound for the combinational complexity of Boolean functions," Proc. 7th Ann. ACM Symp. on Th. of Comp. Albuquerque (1975), 27-36.

[18] N. Pippenger. "Short formulae for symmetric functions," IBM Research Report RC-5143, (1974), 15 pp.

[19] N. Pippenger. "Information theory and the complexity of Boolean functions," submitted to Math. Systems Theory, (1976).

[20] N. Pippenger and M. J. Fischer. "Relations among complexity measures," in preparation.

[21] J. Riordan and C. E. Shannon. "The number of two-terminal series-parallel networks," J. Math. and Physics 21 (1942), 83-93.

[22] J. E. Savage. The Complexity of Computing (manuscript).

[23] J. E. Savage. "Computational work and time on finite machines", J. ACM 19, 4 (1972), 660-674.

[24] C. P. Schnorr. "Zwei lineare untere Schranken fuer die Komplexitaet Boolescher Funktionen," Computing 13 (1974), 155-171.

[25] C. P. Schnorr. "The combinational complexity of equivalence," to appear in Theoretical Computer Science 2.

[26] C. E. Shannon. "The synthesis of two-terminal switching circuits," Bell System Technical Journal 28 (1949), 59-98.

[27] P. M. Spira. "On time-hardware complexity tradeoffs for Boolean functions," Proc. 4th Hawaii Int. Symp. on System Sciences (1971), 525-527.

[28] P. M. Spira. "On the time necessary to compute switching functions," IEEE Trans. Computers C-20 (1971), 104-105.

[29] L. J. Stockmeyer. "On the combinational complexity of certain symmetric Boolean functions," IBM Research RC 5829 Math. (1976).

[30] L. J. Stockmeyer and A. R. Meyer. "Inherent computational complexity of decision problems in logic and automata Theory." To appear in the series: Lecture Notes in Computer Science, Springer (1977). An earlier version of this appears as: "The complexity of decision problems in automata theory and logic," by L. J. Stockmeyer, MAC TR 133, M.I.T. (1974).

Michael S. PATERSON
School of Computer Science
University of Warwick
COVENTRY - Warwickshire CV4 7AL
Grande-Bretagne