SOME STUDIES ON CIRCUIT COMPLEXITY

A dissertation submitted in partial fulfilment of the requirements for the
M. Tech. (Computer Science) degree of the Indian Statistical Institute

By
Pranab Chakraborty

under the supervision of

Dr. Rana Barua
Stat-Math Unit
INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Calcutta-700035

July 24, 1996

Certificate of Approval

This is to certify that the thesis titled **SOME STUDIES ON CIRCUIT COMPLEXITY** submitted by **Pranab Chakraborty**, towards partial fulfilment of the requirements for the degree of M. Tech. in Computer Science at the Indian Statistical Institute, Calcutta, embodies the work done under my supervision.

July 24, 1996

Dr. Rana Barua
Stat-Math Unit
Indian Statistical Institute
Calcutta-700 035

Acknowledgements

First, I express my sincere gratitude to my supervisor Dr. Rana Barua for his encouragement and guidance throughout my dissertation work. I feel fortunate to have attended the course on Abstract Algebra, offered by him in our first semester, which will remain as an unforgettable experience for me.

I am deeply indebted to Dr. Jaikumar Radhakrishnan for introducing me to the wonderful subject of circuit complexity. I have also learnt much, while working under him, during my summer training at TIFR, Bombay.

I would like to thank the following people who patiently attended my seminars on communication complexity and helped to clear my concepts on the subject through valuable discussions : Dr. Bimal Roy, Mr. Palash Sarkar, Mr. Subhash C. Nandy, Mr. Pabitra PalChaudhuri, Mr. Jeet Chaudhuri, Mr. Prakash Narayanan, Mr. Subhashis Majumder, Mr. Subhamoy Maitra and Mr. Paramartha Dutta.

Special thanks goes to Mr. Kaushik Dutta, for allowing me to use his personal computer at unearthly hours. Without that facility it wouldn't have been possible for me to finish this document in time. I am also thankful to Mr. Samik Sengupta, Mr. Pinakpani Pal, Mr. Rana Aich, Mr. Bhaktipada Kundu, Mr. Biswadeep Biswas and Mr. Soumen Ghosh who have helped me in many ways.

Finally, I must thank all my classmates for making my two year experience at ISI a memorable one.

Contents

Chapter 1

Introduction

The **Circuit Complexity** of Boolean functions is a topic of long-standing interest in Computer Science. The subject originated form the requirement of minimising hardware circuit costs for computing Boolean functions. The pioneering contribution in this direction was made by Shannon in a paper[47] in which he proposed the size of smallest circuit computing a function to be the measure of its complexity. He proved an upper bound on the complexity of all n-input functions and used a counting argument to show that for most of the Boolean functions this bound is not too far off. The problem of finding lower bounds on the complexity of circuits computing Boolean functions has motivated considerable research over the past four decades.

The other motivation to study this subject came from the need of developing suitable mathematical models to study the theory of **Computational Complexity**. The classification of problems from the point of view of computation can be divided into two major subdisciplines. One of these is the theory of **Algorithms** which gives the upper bounds on the amount of computational resources (time, space etc.) needed to solve particular problems. The other one consists of techniques for analysing a problem in some computational model irrespective of any algorithm and tries to get some lower bound on the amount of resources required to solve it. The most common model of computation is Turing Machine model. Hartmanis and Stearns[20] first formalised the measure of complexity of functions as time on Turing machine and Edmonds[18] felt the need of avoiding brute force search method and foresaw the issue of polynomial-vs-exponential complexity. Cook, in his 1971 paper[15], precisely formulated the $\mathcal{P}$ vs. $\mathcal{NP}$ conjecture. Savage[46] first showed the connection between circuit complexity and Turing machine time. Over the years, many researchers have felt that the combinatorial and static nature of Boolean circuits renders them more suitable for proving lower bounds on complexity of problems and allows for natural variations and restrictions. Thus, gradually it became apparent that devising new frameworks for studying complexity in circuit model and applying them to explicit problems may lead towards solving the

interesting unsolved questions of Complexity Theory.

Outline of this document

This document consists of two parts. In the first part, described in Chapter 2 and Chapter 3, a brief survey on circuit complexity is presented. Details of major portion of this review are contained in the survey work by Boppana and Sipser[13], in the book by Wegener[55], in the Ph.D thesis of Mauricio Karchmer[23] and some recent papers on the *fusion method*[24],[9]. We have tried to touch upon the different techniques available to prove circuit lower bounds and have attempted to list down those results that are considered as breakthroughs achieved in this subject. In Chapter 4, we describe some of our results. This is followed by Conclusion and then a list of references.

Chapter 2

Introduction to Circuit complexity

2.1 The circuit model

A *Boolean circuit* is a directed acyclic graph. Each node of indegree 0, called *input node*, is labeled with a variable or a constant (0 or 1). Each internal node(n_i), called *gate*, has indegree $\delta_i (1 \leq \delta_i \leq k)$ where k is fixed and is labeled with a Boolean function on at most δ_i-inputs. Unless othewise specified, these Boolean functions are restricted to AND($\wedge$), OR($\vee$) and NOT($\neg$). One of the nodes with outdegree 0 is designated the *output node.* Indegree and outdegree of a node are referred as *fanin* and *fanout* respectively. For a Boolean circuit C, $size(s(C))$ denotes the number of gates and the $depth(d(C))$ denotes the maximum distance(in terms of number of gates) from an input to output. A ***Boolean formula*** is a special kind of circuit whose underlying graph is a tree. In a *monotone* circuit no NOT gates are allowed.

A Boolean circuit computes a Boolean function in a natural way. For a function $f : \{0,1\}^n \rightarrow \{0,1\}$, the *circuit complexity* of f is the size of the smallest circuit computing f and is represented as $C(f)$. For $g : \{0,1\}^* \rightarrow \{0,1\}$ and $h : \mathcal{N} \rightarrow \mathcal{N}$, we say that g has *circuit complexity* h if, for all n, $C(g_n) = h(n)$, where g_n is g restricted to $\{0,1\}^n$. The *circuit complexity of a language* is the circuit complexity of its characteristic function. *Depth complexity*($D(f)$) is defined in an analogous fashion.

The class of all n-variable Boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$ is denoted by B_n. There exist 2^{2^n} functions in B_n. A Boolean function is called *monotone* if for $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \{0,1\}^n$, $x \leq y$ ($i.e., x_i \leq y_i \quad \forall i \in \{1, \ldots, n\}$) implies $f(x) \leq f(y)$. For a monotone function f, a *minterm*(*maxterm*) is a minimal set of variables which if we set to 1(0), the function f is set to 1(0). Let $min(f)(max(f))$ be the set of *minterms* and *maxterms* of f respectively. The class of all n-variable monotone functions is denoted by M_n. For a monotone circuit C and a monotone function f,

$s_m(C)$, $d_m(C)$, $s_m(f)$ and $d_m(f)$ are defined in the obvious way. The following facts are immediate.

Fact 2.1 *For a monotone function f, for every $p \in min(f)$ and every $q \in max(f)$, $p \cap q \neq \emptyset$.*

Fact 2.2 *The only functions computable by monotone circuits are monotone functions.*

In the definition of Boolean circuit we have restricted the basic gate operations to $\{\wedge, \vee, \neg\}$, the *DeMorgan basis*. In general we can choose any finite set(Ω) of Boolean functions as *basis*. A basis Ω is called *complete* if any Boolean function can be computed by some Boolean circuit that uses gates labeled by functions solely chosen from Ω. Examples of complete bases are B_2, *DeMorgan* basis, $\{\wedge, \neg\}$, $\{\vee, \neg\}$, $\{1, \oplus, \wedge\}$, $U_2 = B_2 \setminus \{\oplus, \equiv\}$ etc. The monotone basis $\{\wedge, \vee\}$ is incomplete. The following theorem asserts that size and depth complexities of a Boolean function can increase by at most a constant factor if we switch from one complete basis to another.

Theorem 2.1 *Let Ω and Ω' be complete bases, $c = max\{C_\Omega(g) |\ g \in \Omega'\}$ and $d = max\{D_\Omega(g) |\ g \in \Omega'\}$. Then $C_\Omega(f) \leq c\ C_{\Omega'}(f)$ and $D_\Omega(f) \leq d\ D_{\Omega'}(f)$ for all $f \in B_n$.*

Proof. (*From* [55]) We replace gates for $g \in \Omega'$ by optimal (with respect to size or depth) Ω-circuits for g. Starting with an Ω'-circuit computing f we obtain an Ω-circuit with the required properties.

■

We denote by $C(f)$ and $d(f)$, the size and depth complexities of f with respect to the *DeMorgan* basis. In those places where we may use some other basis(Ω) we would explicitly refer the complexity measures as $C_\Omega(f)$ and $d_\Omega(f)$. In the circuit model described above, fanin of each gate is restricted to k. The size and depth complexities of a circuit can increase by at most a constant factor if we switch from fanin k model to fanin 2 model. All the notions of circuit complexity can naturally be extended to *unbounded fanin* situation. Unbounded fanin circuits were first studied in[14].
We now define some complexity classes according to [23].

Definition 2.1 *$\mathcal{NC}^k$ is the set of all families $\mathcal{F}$ of functions for which there exists a family $\mathcal{C}$ of circuits computing $\mathcal{F}$ such that, for all n, $d(C_n) = O(\log^k(n))$ and $s(C_n) = n^{O(1)}$. $\mathcal{NC} = \bigcup_k \mathcal{NC}^k$.*

Definition 2.2 *$\mathcal{AC}^k$ is the set of all families $\mathcal{F}$ of functions for which there exist a family C of unbounded fanin circuits computing $\mathcal{F}$ such that, for all n, $d(C_n) = O(\log^k(n))$ and $s(C_n) = n^{O(1)}$. $\mathcal{AC} = \bigcup_k \mathcal{AC}^k$.*

Definition 2.3 *P(non-uniform) is the set of all families $\mathcal{F}$ of functions which have polynomial circuit complexity.*

We can easily verify the following facts about the complexity classes defined above.

Fact 2.3 *For every k, $\mathcal{AC}^k \subseteq \mathcal{NC}^{k+1} \subseteq \mathcal{AC}^{k+1}$.*

Fact 2.4 *$\mathcal{NC} = \mathcal{AC} \subseteq$ P(non-uniform)*

We can define the monotone versions of the above classes in a similar fashion and the corresponding facts hold there.

2.2 Boolean circuit and Turing machine

The connection between circuit complexity and Turing machine complexity was first shown by Savage[46]. Pippenger and Fischer[37] proved the following theorem.

Theorem 2.2 *[37] If a language $\mathcal{L}$ is accepted by a deterministic Turing machine in $O(T(n))$ steps, then $\mathcal{L}$ has circuit complexity $O(T(n)\log(T(n)))$.*

To demonstrate the main idea of simulating Turing machine computation in circuits, we describe a weaker version of the above theorem as follows.

Theorem 2.3 *If a language $\mathcal{L}$ is accepted by a deterministic Turing machine in $O(T(n))$ steps, then $\mathcal{L}$ has circuit complexity $O(T^4(n))$.*

Proof. (*From* [13]) Let a k-tape, k-head Turing machine $\mathcal{M}$ accept $\mathcal{L}$ in $O(T(n))$ steps. Converting $\mathcal{M}$ to a single tape, single head Turing machine the number of steps increases to $O(T^2(n))$. Now the computation on some input can be viewed as a table where row i is the tape at i-th step. At any point in time we consider the cell currently scanned by the head to contain a symbol representing both the actual symbol and the state of the machine. Let $cell(i,j)$ be the contents of the ith cell at step j. It is easy to see that $cell(i,j)$ only depends on its predecessors $cell(i-1,j-1)$, $cell(i,j-1)$ and $cell(i+1,j-1)$. Thus the possible values of a cell may be encoded in binary and build a small circuit (independent of the length of the input) for each cell which computes its value from its predecessors. Assuming the machine indicates its acceptance in the first tape cell upon halting, we designate the appropriate gate from $cell(1,l)$ to be the output where l is the

index of the last row. Since the total number of cells is $O((T^2(n))^2) = O(T^4(n))$, the simulating circuit has size $O(T^4(n))$. ∎

Thus every language in $\mathcal{P}$ has polynomial circuit complexity. The converse of this is not true. The following example illustrates this.

Example 2.1 *It is known that there exist subsets of the set of natural numbers($\mathcal{N}$) which are recursively enumerable(r.e) but not recursive. Consider one such subset K. Since K is not recursive there can not exist any algorithm that could be used to test whether a given number $x \in K$. So the subset of $\{1\}^*$ which corresponds to K can not be in $\mathcal{P}$. On the other hand every subset of $\{1\}^*$ has linear size circuits. Thus the converse of the theorem does not hold.*

In the above counter-example, we showed that a family of small circuits exist but it isnot clear how these circuits could be obtained. On the other hand, the circuits promised by Theorem 2.3 are quite regular in nature, that is, there exists a polynomial time program which when presented the input n in unary, produces the description of the nth circuit in the family. If this uniformity restriction is imposed on the circuits then it can be shown that the converse holds.
A family of circuits $\{C_1, C_2, \ldots\}$ is said to be $\mathcal{P}$-uniform if there exists a polynomial time program to generate the description of C_n given n in unary.

Theorem 2.4 *If $\mathcal{L}$ has $\mathcal{P}$-uniform circuits then $\mathcal{L} \in \mathcal{P}$.*

Proof. (*From* [38]) Let the $\mathcal{P}$-uniform circuits recognizing $\mathcal{L}$ be $\{C_1, C_2, \ldots\}$. Suppose a subroutine A generates the description of C_i in polynomial time given input $1^{\{i\}}$. Then the following polynomial time program recognizes $\mathcal{L}$.

1. Input x (let $|x| = n$).
2. $C \leftarrow \mathrm{A}(1^{\{n\}})$.
3. Output $C(x)$.

∎

Alternatively, we can view the converse side through a *nonuniform* extension of $\mathcal{P}$.

Definition 2.4 *We say that a language $\mathcal{L}$ is in $\mathcal{P}/poly$ if there exists a sequence $\{a_1, a_2, \ldots\} \subseteq \{0,1\}^*$, called an* advice sequence, *a polynomial $p(n)$, and a language $\mathcal{L}' \in \mathcal{P}$, such that*

- $\forall n \in \mathcal{N},\ |a_n| \leq p(n)$
- $\forall x \in \{0,1\}^*,\ x \in \mathcal{L} \iff < x, a_{|x|} > \in \mathcal{L}'$.

Theorem 2.5 *$\mathcal{L}$ is in $\mathcal{P}/poly$ iff $\mathcal{L}$ has polynomial size circuits.*

Proof. (*From* [38])
($\Rightarrow$) : Suppose $\mathcal{L} \in \mathcal{P}/poly$. Then there exists a sequence of advice $\{a_1, a_2, \ldots\}$ and a language $\mathcal{L}' \in \mathcal{P}$ satisfying Definition 2.4. Thus $\mathcal{L}'$ has polynomial size circuits, say, $\{C_1, C_2, \ldots\}$. The circuits $\{C_1', C_2', \ldots\}$ for the language $\mathcal{L}$ are obtained from $C_{i+|a_i|}$ by presetting the advice a_i.
($\Leftarrow$) : We encode the polynomial size circuits for $\mathcal{L}$ as advice. Since a polynomial size circuit can be evaluated on any input efficiently, this constitutes a valid advice sequence. ∎

Thus one primary motivation towards studying circuit complexity arises from the fact that if one could prove a super-polynomial lower bound on circuit size for computing an explicit problem in $\mathcal{NP}$ it would mean $\mathcal{P} \neq \mathcal{NP}$. The circuit model appears to researchers to be more static and easy to reason about than programs. However so far, no function in $\mathcal{NP}$ has been shown to require even super-linear circuit size. On the other hand, the truth may be that all $\mathcal{NP}$ problems have polynomial circuit complexity, even though $\mathcal{P} \neq \mathcal{NP}$. If it were the case, Karp and Lipton[28] and Sipser showed that, the polynomial hierarchy would collapse and that is somewhat unexpected.

The fundamental difference between circuit model and a software program is that a circuit works only for inputs of a definite length, whereas a reasonable program works for inputs of arbitrary length. The software model is termed as an *uniform* computation model. Turing machine is a representative of such model whereas Boolean circuit is a representative of *nonuniform* computation model.

2.3 Nonexplicit lower bounds

The number of circuits with small circuit size or small depth grows much slower than the the number of Boolean functions which implies that almost all functions are hard. We describe the following result obtained by Muller[34] who, based on a counting argument of Shannon[47], proved an exponential lower bound on the circuit complexity of nonexplicit problems.

Definition 2.5 *A statement of the form "almost all functions f of a class $F_n \subseteq B_n$ have property P" stands for the assertion that*

$$|\{f \in F_n |\ f \text{ has } P\}|/|F_n| \to 1 \quad as \quad n \to \infty.$$

Theorem 2.6 *[34] Almost every Boolean function of n variables requires circuits of size $\Omega(2^n/n)$.*

Proof. At first we prove that the number of circuits with n variables and size s is bounded above by $(3\cdot(s+n+2)^k)^s$. Each gate in a circuit is assigned an AND or OR or NOT operator that acts on at most k previous nodes. Each previous node can either be a previous gate(at most s choices), a literal, i.e., a variable(n choices) or a constant(2 choices). Thus each gate has at most $3\cdot(s+n+2)^k$ choices. Compounding thsese choices for all s gates proves the claimed upper bound. Now for $s = 2^n/(5kn)$, the bound is approximately $2^{2^n/5} \ll 2^{2^n}$. Since there are 2^{2^n} Boolean function of n variables, almost every Boolean function requires circuits of size larger than $2^n/(5kn)$. Hence the result. ∎

This lower bound is optimal upto a constant factor. We know that every Boolean function can be expressed in disjunctive normal form. Thus every Boolean function has circuits of size $O(2^n \cdot n)$. Muller[34] and Lupanov[32] showed that it is possible to improve this upper bound to $O(2^n/n)$.

2.4 Explicit lower bounds in general circuits

Although for almost all functions the general upper bound on circuit complexity is optimal upto a constant factor, no one has so far been able to show a super-linear lower bound for any explicitly defined function $f \in B_n$. Blum[11], improving a bound of Paul[36], proved a $3n - o(n)$ lower bound over B_2 basis and fanin restricted to 2. The best known lower bound over U_2 basis is $4n - O(1)$ due to Zwick[58].
The method used to prove these bounds is referred as the *gate elimination technique.* To illustrate this technique we describe the following simple bound for threshold functions. Let $TH_{k,n}$ be the function that outputs 1 iff at least k of its n variables are 1.

Theorem 2.7 *For $n \geq 2$, the function $TH_{2,n}$ requires circuits of size at least $2n-4$.*

Proof. (*From* [13]) The proof is by induction on n. For $n = 2$ and $n = 3$, the bound is trivial. Otherwise, let C be an optimal circuit for $TH_{2,n}$ and suppose w.l.o.g that the bottom most gate of C acts on variables x_i and x_j $(i \neq j)$. Now under the four possible settings of x_i and x_j, the function $TH_{2,n}$ has three possible subfunctions, namely, $TH_{0,n-2}$, $TH_{1,n-2}$, and $TH_{2,n-2}$. Thus, one of x_i, x_j (suppose x_i) must feed another gate of C, for otherwise C would have at most two inequivalent subcircuits. Setting x_i to 0 will now eliminate the need for at least 2 gates from C. The resulting function is $TH_{2,n-1}$, which by induction requires circuits of size $2(n-1)-4$. Adding the two eliminated gates to this bound shows that C has at least $2n-4$ gates which completes the induction. ∎

We state some lower bound results for explicit functions from [55], which were obtained by using the gate elimination technique.

Result 2.1 *For* $n \geq 2$ *and* $c \in \{0,1\}$ *the* B_2 *circuit complexity of the* PARITY *function* $x_1 \oplus \cdots \oplus x_n \oplus c$ *is* $(n-1)$, *the* U_2 *circuit complexity equals* $3(n-1)$ *and the DeMorgan circuit complexity is* $4(n-1)$.

Result 2.2 *Consider the equality test function* $f_n^{=} \in B_{2n}$ *defined by,* $f_n^{=}(x_1, \ldots, x_n, y_1, \ldots, y_n) = 1$ *iff* $(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$. B_2 *circuit complexity of* $f_n^{=}$ *is* $(2n-1)$, U_2 *circuit complexity is* $(4n-1)$ *and DeMorgan circuit complexity[45] is* $(5n-1)$.

The best known lower bound over B_2 was proved by Blum[11] by considering the following function.

Result 2.3 *Let* $n = 2^k$, $a = (a_{k-1}, \ldots, a_0)$, $b = (b_{k-1}, \ldots, b_0)$, $c = (c_{k-1}, \ldots, c_0)$, $x = (x_0, \ldots, x_{n-1})$ *and* p, q, r *be Boolean variables. If* $f \in B_{n+3k+3}$ *is defined by,*

$$f(a,b,c,p,q,r,x) = [q \wedge ((x_{|a|} \wedge x_{|b|}) \vee (p \wedge x_{|b|} \wedge x_{|c|}{}^{r}))] \vee [\overline{q} \wedge (x_{|a|} \oplus x_{|b|})],$$

then $C_{B_2}(f) \geq (3n-3)$.

Note : $x_{|c|}^{r} = x_{|c|} \oplus r$

Chapter 3

Restricted circuit models

The apparent difficulty in proving strong lower bounds for explicitly defined functions in general circuit model, led researchers to explore the situation in restricted circuit models.

> We want to prove that the computer cannot do something quickly. We cannot(*have not been able to*) do this. But if we tie the hands and feet of the computer together maybe we will have better luck. The hope being of course that we eventually will be able to remove the ropes and prove that the full powered computer needs a long time.
>
> To be more technical, we want to study a weaker model of computation and develop techniques for proving lower bounds in this weaker model, and may eventually be able to extend these techniques to the general situation.
>
> – J. Håstad[21].

Some examples of the restricted circuit models are *bounded depth circuits*, *monotone circuits* and *formulas*. We discuss about these models in this chapter.

3.1 Bounded-depth circuit model

This model, also called *small-depth circuit model*, restricts the depth of computing circuit to be either a constant independent of the input length, or a slowly growing function of the input length. The first strong lower bound in this model was proved by Furst, Saxe and Sipser[19] and independently by Ajtai[1]. Furst et.al.[19] proved that circuits of depth k that compute PARITY require size $\Omega(n^{\log^{[3(k-2)]} n})$ where $\log^{[i]} n$ denotes the logarithm function iterated i-times. Ajtai, using a probabilistic-combinatorial method, proved a stronger bound $\Omega(n^{c_k \log n})$. These results implied that $\mathcal{AC}^0 \subset \mathcal{NC}^1$.

Subsequently, Yao[57] gave a deeper analysis using the method of [19] to prove an exponential lower bound of $\Omega(2^{n^{1/4k}})$. Hastad[21] further strengthened and simplified the arguments in his Ph.D thesis. We state the results obtained by him.

Theorem 3.1 *[21] There is no depth-k* PARITY *circuit of size* $2^{(1/10)^{k/(k-1)}n^{1/(k-1)}}$ *for* $n > {n_0}^k$ *for some absolute constant* n_0.

Corollary 3.1 *[21] Polynomial size* PARITY *circuits must have depth at least* $\frac{\log n}{c+\log\log n}$ *for some constant* c.

The bound on the size can not be significantly improved as it is quite close to the easily obtained upper bound.

Theorem 3.2 *[21]* PARITY *can be computed by circuits of size* $O(n^{(k-2)/(k-1)}2^{n^{1/(k-1)}})$ *and depth* k.

The depth bound given in the corollary is tight since for every constant c, there exists a polynomial size circuit.

Complexity of MAJORITY

MAJORITY($x_1,\ldots,x_n$) is 1 iff at least half of the x_i are 1.
The lower bounds for PARITY are also valid for MAJORITY and are also close to optimal. If we can compute MAJORITY we can also compute "at least k" and "at most k", for any k, in almost similar complexity. Taking AND of these two functions we get the function "exactly k" and finally OR of circuits computing "exactly k" for odd $k \leq n$ gives PARITY of n variables. So given a gate which computes MAJORITY one can construct constant depth circuits of linear size which computes PARITY. Thus one can say that MAJORITY is at least as hard as PARITY and in general as hard as any symmetric function. Hastad[21] proved that constant depth polynomial size circuits computing MAJORITY that contain PARITY gates need at least $\Omega((\log n)^{3/2})$ PARITY gates.
The technique used by Hastad to arrive at the above-mentioned results was the method of probabilistic restriction which was first introduced by Sipser in [19]. This, in fact, was preceded and motivated by an infinitary version of a result in [48] in which he suggested a finite/infinite analogy :

Polynomial growth is to exponential growth as countability is to uncountability.

This analogy hints at a suggestive correspondence between finite complexity and definability in descriptive set theory. The class $\mathcal{AC}^0$ corresponds to the class of Borel sets and the class $\mathcal{NP}$ corresponds to the class of analytic sets. It was proved in [48], that there is no countable depth-d circuit computing a parity function on ω many variables. Barua[8] showed that Sipser's result on ω-parity functions can be viewed as some sort of Ramsey property.

3.1.1 Probabilistic bounded-depth circuits

A probabilistic circuit C is one in which, in addition to its standard inputs $x_1, \ldots, x_n$, some specially designated inputs $y_1, \ldots, y_n$, called *random* inputs, are provided. These random inputs are chosen from a uniform distribution. So the output of the circuit $C(x)$, is a random variable. A probabilistic circuit is said to (a, b)-*accept* a language if it outputs 1 with probability at most a for strings outside this language and outputs 1 with probability at least b for strings in the language. The circuit accepts a language with ε-*error* if it $(\varepsilon, 1-\varepsilon)$- *accepts* and it accepts with ε-*advantage* if it $(0.5, 0.5+\varepsilon)$-*accepts*. Ajtai and Ben-Or[2] proved the following theorem.

Theorem 3.3 *[2] Every probabilistic circuit of size s and depth d that accepts the language with $(\log^{-k} n)$-advantage (for fixed k) has an equivalent deterministic circuit of size $\mathbf{poly}(n) \cdot s$ and depth $(d + 2k + 2)$.*

3.1.2 Circuits with MOD_p gates

Razborov[40] extended the results of bounded-depth circuits to obtain exponential lower bound for MAJORITY function on more powerful small-depth circuit model having AND, OR and PARITY gates. Subsequently, Smolensky[51] showed that for any p and q powers of distinct primes, the MOD_p function cannot be computed with AND, OR, NOT, MOD_q circuits of polynomial size and constant depth. The method used by them is briefly as follows.
The gates of circuit C are thought as operating on functions rather than on Boolean variables. The results of each of AND and OR operators are slightly adjusted in such a way that (1) each adjustment alters the output of C on few input settings, while (2) the end result is a function which differs from MOD_p in many input settings. Hence many adjustments must occur, thereby giving a strong lower bound on circuit size.

3.2 Monotone circuits

A *monotone* circuit is a Boolean circuit with AND gates and OR gates, but with no NOT gate and the only functions computable by monotone circuits are monotone functions. Many important graph-theoretic functions like CLIQUE, Hamilton-circuit etc., are monotone.
The monontone circuit complexities of several single-output or multi-output symmetric functions are well studied in the literature.

Boolean sorting : Given n Boolean variables output their values in non-decreasing order. Ajtai, Komlos and Szemeredi[3] gave a very clever construction of monotone circuits of size $O(n \log n)$ for this function. This result is tight since Lamagna and

Savage[31] established an $\Omega(n \log n)$ lower bound for size of monotone circuits computing Boolean sorting. It is interesting to note that Boolean sorting has general circuits of linear size from the observation of Muller and Preparata[35].

Majority : This function has monotone circuits of size $O(n \log n)$ as a consequence of the result of Ajtai et. al[3] for Boolean sorting. Dunne[17] proved a $3.5n$ lower bound on the monotone circuit complexity of MAJORITY.

Boolean matrix multiplication : This problem takes two n by n matrices as input, and outputs their n by n Boolean matrix product. Mehlhorn and Galil[33] independently showed that its monotone circuit complexity is exactly $2n^3 - n^2$. Its general circuit complexity is known to be asymptotically smaller. Coppersmith and Winograd[16] constructed circuits of size $O(n^{2.38})$ for this function.

Boolean sum : For this class of n input, n output functions, each output is an OR of some subset of inputs. Andreev[6] has explicitly constructed, for avery $\epsilon > 0$, a Boolean sum with monotone complexity $\Omega(n^{2-\epsilon})$.

The first strong lower bound in this model is due to Razborov[42] who showed that detecting cliques of size s in a graph with m vertices requires monotone circuits of size $\Omega(m^s/(\log m)^{2s})$ for fixed s and size $m^{\Omega(\log m)}$ for $s = \lfloor \log(m/4) \rfloor$. Shortly, after this, Andreev[5], using methods similar to Razborov, proved an exponential lower bound for a monotone problem in $\mathcal{NP}$. This implies an exponential lower bound for the CLIQUE function since clique is complete (with respect to polynomial monotone projections) for "monotone $\mathcal{NP}$". Alon and Boppana [4] further improved the bound by showing that for $s = \lfloor \frac{1}{4}(m/\log m)^{2/3} \rfloor$, the monotone circuit complexity of CLIQUE(m,s) is $\exp(\Omega((m/\log m)^{1/3}))$. Strong lower bound for monotone complexity of a function does not necessarily imply strong lower bound in general circuit model since Razborov[41] showed that the problem of testing for a perfect matching requires super-polynomial size monotone circuits, whereas this problem is in $\mathcal{P}$. However, for some class of functions, called *slice functions* introduced by Berkowitz[10], the two complexities are polynomially related. A proof of this result is presented in the next section using communication complexity approach. There are some $\mathcal{NP}$-complete slice functions. Thus, super-polynomial lower bound on the monotone circuit complexity of an explicit slice function in $\mathcal{NP}$ would mean $\mathcal{P} \neq \mathcal{NP}$.

3.3 Formulas and circuit depth

A *formula* is a restricted circuit model in which fanout of each internal gate is one. The size of a formula is defined by the number of occurrences of literals in its corresponding Boolean formula. Thus, the size is precisely one more than the number of gates in it. The *formula complexity* of a function, $L_\Omega(f)$ is defined to be the size of the smallest

formula over Ω. The primary motivation for studying formulas is their close relationship to circuit depth which is analogous to parallel time. The following facts are well-known ([52],[54]), both for general and monotone complexities.

Fact 3.1 $s(f) \leq L(f)$.

Fact 3.2 $d(f) = \Theta(\log L(f))$.

Though it is known that almost every function has depth $n - \Theta(\log n)$[47], it is still a major challenge in circuit complexity to construct an explicit function (say in $\mathcal{NP}$) with depth $\omega(\log n)$. The best known general lower bound for formula is $\Omega(n^{5/2-\epsilon})$ due to Andreev[6].
Over *DeMorgan* basis, Khrapchenko[29] presented a method for obtaining size lower bounds of formulas. We describe it here.
Let A and B be two disjoint subsets of $\{0,1\}^n$. A Boolean formula is said to separate A and B, if it outputs 0 for every input from A and outputs 1 for every input from B. Suppose the set $A \otimes B$ is defined as

$$A \otimes B = \{(a,b) : a \in A \text{ and } b \in B \text{ and } a \sim b\},$$

where $a \sim b$ means that the inputs a and b differ on exactly one bit. If $A \otimes B$ is large then it is expected that any formula separating A and B should also be large, since the formula must distinguish many pairs of adjacent inputs.

Theorem 3.4 *[29] Let F be a DeMorgan formula that separates A and B. Then*

$$size(F) \geq \frac{|A \otimes B|^2}{|A| \cdot |B|}.$$

Proof. (Paterson). The proof is by induction on the size of F. If the size of F is 1, then F is just a single literal. Clearly, $|A \otimes B| \leq |A|$ and $|A \otimes B| \leq |B|$. This settles the base case.
Suppose $F = F_1 \wedge F_2$ (the case $F = F_1 \vee F_2$ is handled similarly). Define, $A_1 = \{a \in A : F_1(a) = 0\}$, $A_2 = A \setminus A_1$. Note that F_i is a separator of A_i and B for $i = 1, 2$. Applying the induction hypothesis to the subformula F_i yields

$$size(F_i) \geq \frac{|A_i \otimes B|^2}{(|A_i| \cdot |B|)}.$$

Thus,

$$size(F) = size(F_1) + size(F_2) \geq \frac{|A_1 \otimes B|^2}{|A_1| \cdot |B|} + \frac{|A_2 \otimes B|^2}{|A_2| \cdot |B|} \geq \frac{(|A_1 \otimes B| + |A_2 \otimes B|)^2}{(|A_1| + |A_2|) \cdot |B|}.$$

where the last inequality can be established by cross-multiplication. Since $|A_1 \otimes B| + |A_2 \otimes B| = |A \otimes B|$ and $|A_1| + |A_2| = |A|$, this completes the induction step for F.

■

Khrapchenko's theorem shows that the parity function of n variables requires *DeMorgan* formulas of size $\Omega(n^2)$. It is easy to see that this bound is tight. The theorem also shows that the threshold function $TH_{k,n}$ requires *DeMorgan* formulas of size $\Omega(k \cdot (n-k+1))$. In monotone circuit model, the first super-logarithmic depth lower bound (with respect to circuit size) was proved by Karchmer and Wigderson[25] by considering the CONN(s,t) function. It is known that there exists monotone circuit for CONN(s,t) (connectivity function of size $O(n^3 \log n)$. Karchmer, proposed a new approach, called the *communication complexity approach*, which is based on an equivalence between the circuit depth of a given function, and communication complexity of a related problem. Yannakakis independently discovered this equivalence (implicit in [30]). Using this method, Karchmer gave a tight $\Omega(\log^2 n)$ depth lower bound for CONN(s,t) function. We now discuss the communication complexity approach as described in [23]

3.3.1 Communication complexity approach

We give first an informal description of this approach. Let $\mathcal{L}$ be a language. In the *communication game* for $\mathcal{L}$, there are two players, one having a string in $\mathcal{L}$ and the other having a string of the same length which is not in $\mathcal{L}$. In the game, the players communicate with each other to find a position where the two strings differ. The minimum number of bits that they require to do this over all strings of the same length is the complexity of the game. Karchmer[23] showed that this is exactly equal to the minimum circuit depth necessary for $\mathcal{L}$.

To be more precise : There are two players, I and II, both with unlimited computing power, communicate through a flawless binary channel. The players follow a deterministic protocol, and it is required that they communicate using prefix-free codes so that so that at each point of time, history uniquely determines the players' turn to send a message.

Consider three finite sets X, Y, Z and a (*ternary*) relation, $R \subseteq X \times Y \times Z$. $S(R) \subseteq X \times Y$, called the *support* of R, is the set of pairs $(x, y) \in X \times Y$ such that for each such pair, $\exists z \in Z$ with $(x, y, z) \in R$. If $S(R) = X \times Y$, the relation R is said to be rectangular. Player I is given some x and player II is given some y, where $(x, y) \in S(R)$, so that I and II have to agree on a $z \in Z$ *s.t.*, $(x, y, z) \in R$ by communicating bits among themselves. The number of bits communicated by following a protocol D on x and y is denoted by $D(x, y)$, while the communication pattern or *history* of D is denoted by $\alpha(x, y)$.

Definition 3.1 Communication complexity *of* R, $C(R)$ *is defined as,*
$C(R) = \min_D \max_{(x,y) \in S(R)} \{D(x, y)\}$, *where the* min *is taken over all protocols for* R.

Without loss of generality, if we assume that for every history α of D, $\exists$ an $(x,y) \in S(R)$ s.t., $\alpha = \alpha(x,y)$, then in Definition 3.1, we can maximize over the entire $X \times Y$ plane and we can extend R to $\overline{R}$ by

$$\overline{R} = R \cup \{(x,y,z) : (x,y) \notin S(R), z \in Z\}.$$

Clearly, $C(R) = C(\overline{R})$. It is also possible to look at the above game from the point of view of a third party. A third party can infer the answer to the game by just listening I and II's conversation, even if he cannot see either x or y. This is because every history α is associated with a cartesian product $X' \times Y'$, $X' \in X$ and $Y' \in Y$ and for every $(x,y) \in X' \times Y'$, $\alpha(x,y) = \alpha$.

Restrictions and reductions

Definition 3.2 *The restriction of R into $I \subseteq S(R)$,*

$$R|_I = \{(x,y,z) \in R : (x,y) \in I\}.$$

Lemma 3.1 *[23]* $C(R|_I) \leq C(R)$

Proof. Any protocol for R can be used as a protocol for $R|_I$. ∎

Definition 3.3 *Let $R \subseteq X \times Y \times Z$ and $R' \subseteq X' \times Y' \times Z'$. R is said to be reducible to R', $R \leq R'$, if there exist functions $\phi_I : X \to X'$, $\phi_{II} : Y \to Y'$ and $\psi : Z' \to Z$ such that for every $(x,y) \in S(R)$,*

1. $(\phi_I(x), \phi_{II}(y)) \in S(R')$ *and*
2. $(\phi_I(x), \phi_{II}(y), z') \in R' \Rightarrow (x, y, \psi(z')) \in R$.

Lemma 3.2 *[23]* $C(R) \leq C(R')$.

Proof. A protocol D for R can be constructed out of a protocol D' for R' in such a way that on (x,y), D simulates D' on $(\phi_I(x), \phi_{II}(y))$ at first. If D' answers z' then D answers $\psi(z')$. ∎

If $R \leq R'$ and $R' \leq R$, the two relations R and R' are called equivalent, written as, $R \equiv R'$. We now describe a general game applicable to our present context.

Let $B_0, B_1 \subseteq \{0,1\}^n$ be such that $B_0 \cap B_1 = \emptyset$. Consider the rectangular relation $R(B_1, B_0) \subseteq B_1 \times B_0 \times [n]$ where $(x,y,i) \in R(B_1, B_0)$ iff $x_i \neq y_i$, and the game for $R(B_1, B_0)$ is as follows :

Player I gets x while player II gets y; their goal is to agree on a coordinate i, s.t., $x_i \neq y_i$. $C(B_1, B_0)$ denotes the minimum number of bits they have to communicate inorder for both to agree on such a coordinate. For a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, R_f (or $R[f]$) denotes $R(f^{-1}(1), f^{-1}(0))$.

Theorem 3.5 *[23] For every function* $f : \{0,1\}^n \rightarrow \{0,1\}$,

$$d(f) = C(R_f).$$

Proof. Follows from the following two lemmas. ∎

Lemma 3.3 *[23] For all functions* $f : \{0,1\}^n \rightarrow \{0,1\}$ *and all* $B_0, B_1 \subseteq \{0,1\}^n$ *such that* $B_0 \subseteq f^{-1}(0)$ *and* $B_1 \subseteq f^{-1}(1)$, *we have*

$$C(B_1, B_0) \leq d(f).$$

Proof. In words, this lemma shows how to make a protocol for $R(B_1, B_0)$ out of a circuit for f. We proceed by induction on $d(f)$. If $d(f) = 0$, then f is either x_i or $\overline{x_i}$ or constants 0 or 1. For constants, the game is meaningless. In case, f is either x_i or $\overline{x_i}$, $\forall x \in B_1$ and $\forall y \in B_0$, $x_i \neq y_i$, so that i is always an answer and $C(B_1, B_0) = 0$.
For the induction step, we suppose that $f = f_1 \vee f_2$ (the case, $f = f_1 \wedge f_2$ is treated similarly), and that $d(f) = \max(d(f_1), d(f_2)) + 1$. Let $B_1{}^j = B_1 \cap f_j{}^{-1}(1)$ for $j = 1, 2$. By induction we have, $C(B_1{}^j, B_0) \leq d(f_j)$ for $j = 1, 2$. Consider the following protocol for B_1 and B_0 : I sends a 0 if $x \in B_1{}^1$, otherwise he sends a 1; the players then follow the best protocol in each of the subclasses. We have

$$C(B_1, B_0) \leq 1 + \max_{j=1,2}(C(B_1{}^j, B_0)) \leq 1 + \max_{j=1,2}(d(f_j)) = d(f).$$

∎

Lemma 3.4 *[23] Let* $B_0, B_1 \subseteq \{0,1\}^n$ *be such that* $B_0 \cap B_1 = \emptyset$. *Then* $\exists f : \{0,1\} \rightarrow \{0,1\}$ *with* $B_0 \subseteq f^{-1}(0)$ *and* $B_1 \subseteq f^{-1}(1)$ *s.t.*,

$$d(f) \leq C(B_1, B_0).$$

Proof. The lemma shows how to define a function f, and a circuit computing it out of a protocol $R(B_1, B_0)$. We proceed by induction on $C(B_1, B_0)$. If $C(B_1, B_0) = 0$ then $\exists$ an i s.t., for every $x \in B_1$ and for every $y \in B_0$, $x_i \neq y_i$. Also it is clear that for every $x', x'' \in B_1$ we have $x_i{}' = x_i{}''$ and the same holds for every $y', y'' \in B_0$. Without loss of generality, $x_i = 1$ so letting $f = x_i$, we get $B_0 \subseteq f^{-1}(0)$ and $B_1 \subseteq f^{-1}(1)$.
To prove the induction step, we assume that I sends the first bit (the other case is

treated similarly). For some partition, $B_1 = B_1^1 \cup B_1^2$, I sends a 0 if $x \in B_1^1$ and 1 otherwise; the players then follow the best protocol for each of the subcases and

$$C(B_1, B_0) = 1 + \max_{j=1,2}(C(B_1^j, B_0)).$$

By induction, there exists f_1, f_2 so that $B_1^j \subseteq f_j^{-1}(1)$ and $B_0 \subseteq f_j^{-1}(0)$ and $d(f_j) \leq C(B_1^j, B_0)$ for $j = 1, 2$. We now take, $f = f_1 \vee f_2$. $B_0 \subseteq f_1^{-1}(0) \cap f_2^{-1}(0) = f^{-1}(0)$. $B_1 = B_1^1 \cup B_1^2 \subseteq f_1^{-1}(1) \cup f_2^{-1}(1) = f^{-1}(1)$ and $d(f) \leq 1 + \max_{j=1,2}(d(f_j)) \leq 1 + \max_{j=1,2}(C(B_1^j, B_0)) = C(B_1, B_0)$. ∎

The monotone game

For monotone circuits a modified formulation of Theorem 3.5 was presented in [23]. Consider P, $Q \subseteq \mathcal{P}([n])$ be such that for every $p \in P$ and for every $q \in Q$, $p \cap q \neq \emptyset$. A rectangular relation $R(P, Q) \subseteq P \times Q \times [n]$ is defined such that $(p, q, i) \in R(P, Q)$ iff $i \in p \cap q$. The game for R is to find an element in $p \cap q$. The minimum number of bits that need to be communicated is denoted as $C(P, Q)$. For a monotone function f, R_f^m denotes $R(min(f), max(f))$. Also let $R_f^1 \subseteq f^{-1}(1) \times f^{-1}(0) \times [n]$ where for $x \in f^{-1}(1)$, $y \in f^{-1}(0)$ and $i \in [n]$, $(x, y, i) \in R_f^1$ iff $x_i = 1$ and $y_i = 0$.

Theorem 3.6 *[23] For every monotone function f, we have $d_m(f) = C(R_f^1) = C(R_f^m)$.*

Proof. We first show that $d_m(f) = C(R_f^1)$. In the base case of lemma 3.3, if the circuit is monotone, we can always find an i, s.t., $x_i = 1$ while $y_i = 0$. On the other hand if the protocol always gives a coordinate i with the above property, lemma 3.4 gives a monotone circuit. The induction steps are identical to that of those two lemmas.

We next prove that $R_f^1 \equiv R_f^m$:
i) $R_f^m \leq R_f^1$: Let $p \in min(f)$ and $q \in max(f)$. Let $\phi_I(p)_i = 1$ iff $i \in p$ and let $\phi_{II}(q)_i = 0$ iff $i \in q$. Clearly, $\phi_I(p) \in f^{-1}(1)$ and $\phi_{II}(q) \in f^{-1}(0)$. Thus, $(\phi_I(p), \phi_{II}(q), i) \in R_f^1$ iff $i \in p \cap q$ so that we can take $\psi(i) = i$.
ii) $R_f^1 \leq R_f^m$: Let $x \in f^{-1}(1)$ and $y \in f^{-1}(0)$. Let $p_x = i : x_i = 1$ and $q_y = \{i : y_i = 0\}$. It is easy to see that $\exists p \subseteq p_x$ and $q \subseteq q_y$ such that $p \in min(f)$ and $q \in max(f)$. Moreover, it is clear that $i \in p \cap q$ iff $x_i = 1$ and $y_i = 0$. The conditions of reduction are met by taking $\phi_I(x) = p$, $\phi_{II}(y) = q$ and $\psi(i) = i$. ∎

Inorder to demonstrate the usefulness of Theorem 3.5, Karchmer[23] gave some intuitive and new proofs of old results, improved upper bounds of depth complexity for some functions and proved super-logarithmic monotone depth lower bound for CONN(s,t)

function. We briefly state some of these results below.
By the results of Valiant[53], Boppana[12] and Ajtai et.al.[3], it is known that polynomial size monotone functions exist for all threshold functions. This implies by Theorem 3.5 that $C(R^m[TH_k{}^n]) = O(\log n)$ for every $k = 0, \ldots, n$.
Let f be a function and let $0 \leq k \leq n$. Suppose $w(x)$ denotes the number of 1's in x. The *slice* k of f, f_k, is a function such that for $x \in \{0,1\}^n$, $f_k(x) = 0$ if $w(x) < k$; $f_k(x) = f(x)$ if $w(x) = k$; and $f_k(x) = 1$ if $w(x) > k$. The result of Berkowitz[10] which states that the monotone and non-monotone depth complexities of slice functions are very close to each other can be proved very easily by communication complexity approach.

Theorem 3.7 *[23] Let B_1, $B_0 \subseteq \{0,1\}^n$ be such that $B_1 \cap B_0 = \emptyset$ and for every $x \in B_1 \cup B_0$, $w(x) = k$. Let $R(B_1, B_0)$ be as defined previously and let $R^1(B_1, B_0)$ be a relation such that $(x, y, i) \in R^1(B_1, B_0)$ iff $x_i > y_i$. Then,*

$$C(R^1(B_1, B_0)) \leq C(R(B_1, B_0)) + O(\log n).$$

Proof. A protocol D^1 for $R^1(B_1, B_0)$ can be constructed out of a protocol D for $R(B_1, B_0)$ in such a way that the complexity relationships mentioned in the theorem hold. Let D^1, on (x, y), follow D on (x, y) until it gives an answer i. If $x_i > y_i$, then D^1 terminate. Otherwise, player I thinks of x as if it had $x_i = 1$ so that $k + 1 = w(x) > w(y) = k$. The players then follow the corresponding protocol for $R^m[TH_{k+1}{}^n]$. ∎

From this theorem the following corollary clearly follows.

Corollary 3.2 *[10] Let f_k be a slice function over $\{0,1\}^n$. Then*

$$C(R_{f_k}) \leq C(R_{f_k}{}^m) \leq C(R_{f_k}) + O(\log n).$$

By constructing explicit protocols the following upper bounds were proved in [23].

Theorem 3.8 *[23] For a given bipartite graph $G = (X \cup Y, E)$ with $|X| = |Y| = n$, if* MATCH *denotes a function which is* 1 *iff G has a perfect matching, then*

$$d_m(\text{MATCH}) = C(R^m[\text{MATCH}]) \leq n + O(\log n).$$

Theorem 3.9 *[23] For a given undirected graph $G = (V, E)$ with two distinguished nodes s and t, if* CONN*(s,t) denotes a function which is* 1 *iff G has* a path from s to t, *then*

$$d_m(\text{CONN}(s,t)) = C(R^m[\text{CONN}(s,t)]) \leq \log^2 n + \log n.$$

The following lower bound result was also first presented using communication complexity approach in [23].

Theorem 3.10 *[23] The function* CONN*(s,t) requires monotone circuits of depth* $\Omega((\log n)^2)$.

Raz and Wigderson[39] used the communication game and a lower bound on the probabilistic communication complexity of *the set disjointness problem* due to Kalyanasundaram and Schnitger[22], to get a linear depth lower bound for monotone circuits computing matching on n by n bipartite graph.

3.3.2 The fusion method

The framework that has been developed by extending the analogy between Razborov's generalized approximation method and ultraproduct construction in model theory is known as the *fusion method* which has been studied in different variations[56]. Razborov originally used it to characterize the classes P-nonuniform and $\mathcal{NL}$ and proved a super-linear lower bound for MAJORITY on *switching and rectifying* networks in this framework. Using this approach, Karchmer[24] presented a new proof for the exponential monotone size lower bound of CLIQUE and gave a characterization of $\mathcal{NP}$ through this method. Karchmer & Wigderson[26] used the method to give another characterization of $\mathcal{NP}$ and introduced a linear algebraic model of computation, the span programs, and proved several upper and lower bounds on it. We describe the fusion method according to the approaches in ([24],[9]) as follows.

Background

The circuit model considered in the present section assumes the source nodes to be labeled by literals from $\{x_1, \ldots, x_\alpha\} \cup \{\overline{x_1}, \ldots, \overline{x_\alpha}\}$ and the remaining nodes have fanin 2 (if $\alpha < \omega$) or arbitrary (possibly infinite) fanin in the infinite case ($\alpha = \omega$) and are labeled by one of the two Boolean operations AND($\wedge$) and OR($\vee$)[1]. We only consider circuits of finite depth. A Boolean circuit C computes a function $f \in B_\alpha$ ($\alpha \leq \omega$) in the natural way.
For a Boolean circuit C, $\Lambda(C)$ denotes the set of $\wedge$-gates of C. The number of $\wedge$-gates in C, i.e., $|\Lambda(C)|$ is denoted by $s_\wedge(C)$. For a function $f \in B_\alpha$, $s_\wedge(f)$ denotes the number of $\wedge$-gates of an optimal circuit that computes f. If f is a monotone function, $s^+(f)$ and $s_\wedge{}^+(f)$ denote the size and number of $\wedge$-gates of an optimal monotone circuit for f.

A *non-deterministic* circuit is a circuit with sources labeled by $\{x_1, \overline{x_1}, \ldots, x_\alpha, \overline{x_\alpha}\} \cup \{y_1, \overline{y_1}, \ldots, y_\beta, \overline{y_\beta}\}$ and is said to compute a function $f \in B_\alpha$ such that, for $x \in \{0,1\}^\alpha$, $f(x) = 1$ iff there exists a setting of the non-deterministic variables $\{y_1, \ldots, y_\beta\}$ that makes the circuit output 1. For a function f, $ns(f)$ and $ns_\wedge(f)$ are the size and the

[1]In the finite case, bringing the NOT($\neg$) gates down to source level does not increase the circuit depth and can increase the circuit size by at most a factor of two.

number of $\wedge$-gates of an optimal non-deterministic circuit for f.

A *co-non-deterministic* circuit is a circuit with sources labeled by $\{x_1, \overline{x_1}, \ldots, x_\alpha, \overline{x_\alpha}\} \cup \{y_1, \overline{y_1}, \ldots, y_\beta, \overline{y_\beta}\}$ and is said to compute a function $f \in B_\alpha$ such that, for $x \in \{0,1\}^\alpha$, $f(x) = 0$ iff there exists a setting of the non-deterministic variables $\{y_1, \ldots, y_\beta\}$ that makes the circuit output 0. For a function f, $\overline{n}s(f)$ and $\overline{n}s_\wedge(f)$ are the size and the number of $\wedge$-gates of an optimal co-non-deterministic circuit for f.
Suppose, f is an explicit function in $\mathcal{NP}$. We have already seen that if one could prove super-polynomial circuit complexity of f one would get $\mathcal{P} \neq \mathcal{NP}$. Similarly, if anybody can prove a super-polynomial lower bound of co-non-deterministic circuit size computing f it would mean that $\mathcal{NP} \neq co\text{-}\mathcal{NP}$. This is because, $\overline{n}s(f) = n^{\omega(1)}$ implies $\overline{f}$ is not in $\mathcal{NP}$.
The version of the fusion method which we describe next[24], presents a framework to lower bound the number of $\wedge$-gates needed to compute f. The following lemma due to Alon and Boppana[4] shows that the number of $\wedge$-gates cannot be much smaller than the total number of gates. Thus, if $s(f)$ is super-polynomial then so is $s_\wedge(f)$.

Lemma 3.5 *[4] For any function f, $s(f) = O(s_\wedge(f)^2)$.*

The main idea

Consider a "hard" function f and assume that C is a "small" circuit which allegedly computes f. Let $U \subseteq f^{-1}(0)$ be a subset of choice. Thus C should reject all vectors of U. Now, if it is possible to prove that rejecting computations for the vectors in U can be *combined* (in a sense to be explained later), to get a rejecting computation for a vector in $f^{-1}(1)$, it would mean that C does not compute f correctly.
A more formal description : There are two stages in this method. In the first stage, to each node g of C, a subset $[|g|] \rightleftharpoons \{u \in U : g(u) = 1\}$ is assigned.

Fact 3.3 *For any $g, h \in B_n$*
1. $[|g \wedge h|] = [|g|] \cap [|h|]$.
2. $[|g \vee h|] = [|g|] \cup [|h|]$.
3. $[|\overline{g}|] = [|g|]^c$.

In the second stage, a *filter* $\mathcal{F}$ over $\mathcal{P}(U)$ is chosen and g is given the value 1 if and only if $[|g|] \in \mathcal{F}$.

Definition 3.4 *A filter $\mathcal{F}$ over $\mathcal{P}(U)$ is a non-trivial upward-closed collection of subsets of U, i.e. $\emptyset \notin \mathcal{F}$ and if $A \in \mathcal{F}$ and $A \subseteq B$ then $B \in \mathcal{F}$.*

It is to be noted that the output of C will be assigned 0 value in the second stage since $\emptyset \notin \mathcal{F}$. In contrast to the usual definition of a filter over a Boolean algebra, here, the

requirement of closure under $\wedge$ is ommitted. Conventional filters are quite trivial for finite Boolean algebras.

Definition 3.5 *A filter $\mathcal{F}$ is said to preserve a pair (A, B) of subsets of U if $A, B \in \mathcal{F}$ implies that $A \cap B \in \mathcal{F}$. Similarly, $\mathcal{F}$ preserves an $\wedge$-gate, fed by functions g and h, if it preserves the pair $([|g|], [|h|])$. Preserving a collection Λ of $\wedge$-gates means preserving each of its members.*

Definition 3.6 *A filter $\mathcal{F}$ is above a vector $v \in \{0,1\}^n$ if for every $i = 1, \ldots, n$, $v_i = 1 \Rightarrow [|x_i|] \in \mathcal{F}$ and $v_i = 0 \Rightarrow [|\overline{x_i}|] \in \mathcal{F}$.*

Definition 3.7 *A filter $\mathcal{F}$ is said to majorize a computation of a vector v if for every subfunction g of C, $g(v) = 1 \Rightarrow [|g|] \in \mathcal{F}$.*

Lemma 3.6 *[24] If $\mathcal{F}$ is above a vector v and $\mathcal{F}$ preserves Λ then $\mathcal{F}$ majorizes the computation of v.*

Proof. Suppose $\mathcal{F}$ does not majorize the computation of v. Now let us consider the first node of C that is not majorized computes g. Clearly, g cannot be an input node as $\mathcal{F}$ is above v. Again, as $\mathcal{F}$ preserves Λ, g cannot be output of an $\wedge$-gate. Finally, the upward closure property of $\mathcal{F}$ prevents g from being the output of an $\vee$-gate. This yields contradiction. ∎

Definition 3.8 *For a function f, let $\rho(f)$ denote the minimum size of a collection Λ of pairs of subsets of $f^{-1}(0)$ such that there is no filter above a vector in $f^{-1}(1)$ which preserves Λ.*

Theorem 3.11 *[24],[44] For any $f \in B_n$, $s_\wedge(f) \geq \rho(f)$.*

Proof. Consider, on the contrary to the theorem statement, that there exists a circuit C with fewer than $\rho(f)$ $\wedge$-gates which allegedly computes f. By Definition 3.8 there is a filter $\mathcal{F}$ which preserves Λ and is above a vector $v \in f^{-1}(1)$. By Lemma 3.6 $\mathcal{F}$ majorizes the computation of v and the circuit C must reject v. This is because, if output(f') of C is 1 then as $\mathcal{F}$ majorizes v, $[|f'|] \in \mathcal{F}$. But $[|f'|]$ is $\emptyset$ and $\emptyset \notin \mathcal{F}$. Thus C does not compute f correctly. ∎

We state the converse of this theorem below.

Theorem 3.12 *[24],[44] For any $f \in B_n$, $s_\wedge(f) = O(\rho(f)^2)$.*

So, if $s_\wedge(f)$ is super-polynomial then so is $\rho(f)$.

For monotone circuits

Monotone circuits do not have any source node labeled by a negated variable. So, while proving lower bounds in monotone circuits, we need to work with only those filters which are *weakly* above some vector $v \in f^{-1}(1)$ (a filter $\mathcal{F}$ is weakly above a vector v if $v_i = 1 \Rightarrow [|x_i|] \in \mathcal{F}$). Using this approach Karchmer[24] proved $\Omega(n^3/\log^4 n)$ lower bound for CLIQUE(n,3) and provided hints on how to generalize it to CLIQUE(n,k).

Ultrafilters and $\mathcal{NP}$

In [24], the following definition of *ultrafilter* has been used inorder to propose a framework for proving lower bounds on co-non-deterministic circuit size.

Definition 3.9 *An ultrafilter $\mathcal{U}$ over U is a filter such that for every $A \subseteq U$, at least one of A or A^c is in $\mathcal{U}$.*

By this definition, any ultrafilter is above some vector.

Definition 3.10 *For a function $f \in B_n$, $\rho_{\mathcal{U}}(f)$ denotes the minimum size of a collection Λ of pairs of subsets of $f^{-1}(0)$ such that there is no ultrafilter above a vector in $f^{-1}(1)$ which preserves Λ.*

Wigderson and Karchmer observed the following.

Theorem 3.13 *For any $f \in B_n$, $\overline{ns}_\wedge(f) \geq \rho_{\mathcal{U}}(f)$. Conversely, $\overline{ns}_\wedge(f) = O(\rho_{\mathcal{U}}(f))$.*

Ultrafilters and countable circuits

Karchmer extended the fusion method to a framework which can be used to show that for some given hard function $f : \{0,1\}^\omega \rightarrow \{0,1\}$, there does not exist any countable co-non-deterministic circuit computing f. This framework is based on conventional ultrafilters.

Definition 3.11 *An ultrafilter $\mathcal{U}$ over $\mathcal{P}(U)$ is an upward closed collection of subsets of U which is closed under finite intersections and for every $A \subseteq U$, exactly one of A or A^c is in $\mathcal{U}$. An ultrafilter is called* principal *if it is* generated *by some $u_0 \in U$ (i.e., the collection of all subsets of U that contains u_0). A* non-principal *ultrafilter is one which is not principal.*

The following facts are immediate from the definition.

Fact 3.4 *An ultrafilter is non-principal iff it contains a finite set.*

Fact 3.5 *For every ultrafilter $\mathcal{U}$ over $\mathcal{P}(U)$, $U \in \mathcal{U}$ (due to upward-closure property) and $\emptyset \notin \mathcal{U}$ (as $U \in \mathcal{U}$).*

It is known that if U is infinite then there exist non-principal ultrafilters over $\mathcal{P}(U)$. The following definition and lemma are crucial in this proof technique.

Definition 3.12 *A collection C of subsets of U has the finite intersection property (f.i.p) iff any finite number of subsets from C has a non-empty intersection.*

Lemma 3.7 *[9] A collection C of subsets of U can be extended to an ultrafilter over $\mathcal{P}(U)$ iff C has the finite intersection property.*

If U is infinite then the collection of all co-finite subsets has the f.i.p and by the lemma can be extended to a (non-principal) ultrafilter.
Similar to the method described using filters, here also we procced in two stages, the only difference being in the first stage, a rejecting witness w_u is fixed for each $u \in U$ inorder to set the non-deterministic inputs so that for each sub-function g of C, $[|g|]$ denotes $\{u \in U : g(u, w_u) = 1\}$. An ultrafilter $\mathcal{U}$ *defines* a Boolean vector $z \in \{0,1\}^\omega$ by letting $z_i = 1$ iff $[|x_i|] \in \mathcal{U}$. The second stage remains identical.
Any ultrafilter $\mathcal{U}$, which gives *consistent* (to be explained next) values to the sub-functions of C and which defines a vector of $f^{-1}(1)$ constitutes a proof that C does not compute f correctly. The values assigned are said to be consistent if they are not less than the actual values. As any ultrafilter is an upward-closed collection, values assigned to outputs of $\vee$-gates are always consistent. So, we have to worry about the consistency of $\wedge$-gates only.
An $\wedge$-gate with countable fanin $\wedge_{j<\omega} g_j$ can be viewed as a countable collection of subsets $\{[|g_j|]\}_{j<\omega}$. An ultrafilter is said to *preserve* the $\wedge$-gate if it gives consistent values to the sub-functions associated with its inputs and output. This happens when the ultrafilter $\mathcal{U}$ contains either $\cap_{j<\omega}[|g_j|]$ or one of the sets $[|g_j|]^c$ for some $j < \omega$. These sets constitute a countable cover of U. The following lemma follows from this discussion.

Lemma 3.8 *[9] A countable co-non-deterministic circuit for f induces a countable collection of countable covers of $U = f^{-1}(0)$, one for each gate. An ultrafilter preserves the $\wedge$-gates of C iff it contains at least one set from each cover.*

Ben-David et.al.[9] gave the following characterization of co-non-deterministic circuit size in terms of sets of covers of the zeroes of a function.

Theorem 3.14 *[9] For every function $f \in B_\omega$, the following conditions are equivalent :*

1. For any countable collection of countable covers of $U = f^{-1}(0)$, there exists some $x \in f^{-1}(1)$ and a collection of subsets one from each cover, which, together with the sets $\{\{u \in U : u_i = x_i\} : i \in \omega\}$ have the f.i.p.

2. There exists no countable co-non-deterministic circuit for f.

The technique discussed above also works if U is taken as some subset of $f^{-1}(0)$. Using the characterization mentioned above, Karchmer proved the following theorem.

Lower bound for clique : Let $\mathcal{G}$ denote the class of all countable undirected graphs. Let us consider these graphs to be encoded as a set set of variables $\{x_{i,j} : i,j \in \mathcal{N}\}$ where $x_{i,j} = 1$ iff the graph has the edge $\{i,j\}$. Let the function $\text{CLIQUE}_\omega \in B_\omega$ be defined as : $\text{CLIQUE}_\omega(x) = 1$ iff $\exists$ an infinite $\mathcal{S} \subseteq \mathcal{N}$ so that $\forall i,j \in \mathcal{S}$, $x_{i,j} = 1$.

Theorem 3.15 *[9] There is no countable co-non-deterministic circuit for* CLIQUE_ω.

In [9] a finite analogue of this method was also proposed to obtain a characterization of co-$\mathcal{NP}$. It was shown that this characterization yields an alternative proof of $\mathcal{NP}$-completeness of CLIQUE function. In another paper[26], Karchmer and Wigderson proposed a linear algebraic variant to the general approximation method of Razborov and derived four different combinatorial problems that characterize non-uniform $\mathcal{NP}$. All these variations of the fusion method present frameworks for studying the $\mathcal{P}$ vs. $\mathcal{NP}$ vs. co-$\mathcal{NP}$ question.

Chapter 4

Some lower and upper bound results

4.1 Using the gate elimination approach

4.1.1 Threshold function

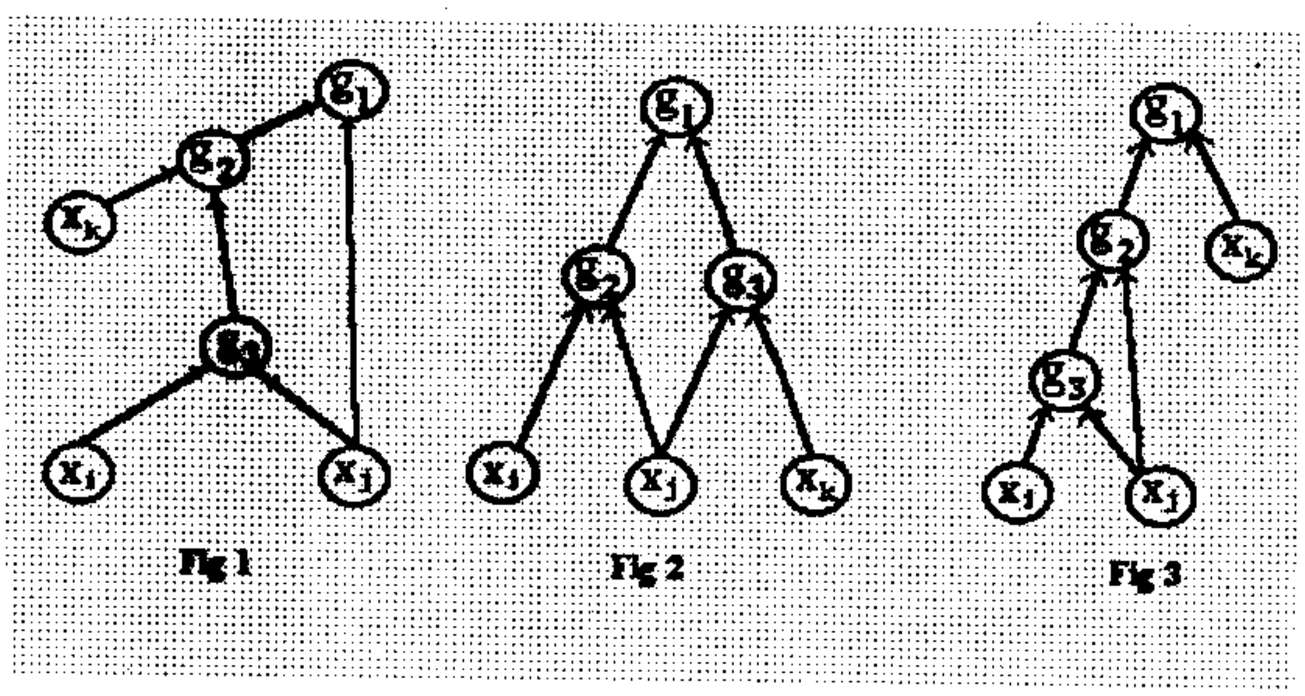
Let us recall that $TH_{k,n}$ represents a function which outputs 1 if and only if at least k out of n input variables are 1. We show here that Theorem 2.7 of Chapter 2 can be slightly improved using the following fact.

Lemma 4.1 $C_{B_2}(TH_{2,2}) = 1$ *and* $C_{B_2}(TH_{2,3}) = 4$.

Proof. $C_{B_2}(TH_{2,2}) = 1$, because the smallest circuit that realizes $TH_{2,2}(x_1, x_2)$ is $(x_1 \wedge x_2)$.
To prove $C_{B_{2,3}} = 4$ we first observe that $TH_{2,3}$ can be realized by a circuit consisting of 4 B_2 gates as
$TH_{2,3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$.
This shows that

$$C_{B_2}(TH_{2,3}) \leq 4 \qquad (a)$$

We now prove that $C_{B_2}(TH_{2,3}) \geq 4$.
Let us consider an optimal circuit C that realizes $TH_{2,3}$ and suppose input variables x_i and x_j $(i \neq j)$ act as inputs to a bottom level gate g of C. One of these two variables (say x_j) must have outdegree two or more since under the four possible settings of x_i and x_j to constants, $TH_{2,3}$ has three subfunctions namely $TH_{1,1}$ and the constant functions 0 and 1.



Claim 4.1 *If size of* C *is less than 4 then* x_j *has outdegree 2 and each of the other two variables* ***has outdegree 1.***

Proof. We know that x_j has outdegree at least 2. So the number of input lines (i.e., the total outdegree of 3 variables) is at least 4. So C must have at least 3 gates. Also, for a 3-gate circuit

total fanin $\geq$ (*the number of input lines* + *the number of gates* $-$ 1). So the number input lines $\leq (6-3+1) = 4$. Thus the claim. ∎

So if size of C is 3 then its topology must fall in one of the three categories shown above (fig. 1-3).

Claim 4.2 *No Boolean circuit over* B_2 *with 3 gates can realize* $TH_{2,3}$.

Proof. As we are concerned with optimal circuits, g_1, g_2 and g_3 can be taken to represent non-degenerate 2-input functions. We deal with three topologies separately.
Case 1 : We consider circuit topology of fig.1. The gate g_1 must be of type $\oplus$ or $\equiv$. Otherwise by choosing appropriate constant for x_j we can force the output to some constant irrespective of the values on x_i and x_k. Similarly, $g_2(g_3)$ can not be of type $(x^a \wedge x^b)^c$ since otherwise by choosing an appropriate constant for $x_k(x_j)$, we can make C independent of x_i. So we are left with the possibility where each of the gates represents functionality $\oplus$ or $\equiv$. This type of circuit can not realize $TH_{2,3}$.
Case 2 : We consider circuit topology of fig.2. Let us try to fit correct functionality into this structure as follows. We assume that when $x_i = 1, x_j = 0 and x_k = 1$ then g_2 and g_3 output c_2 and c_3 respectively where c_2, $c_3 \in \{0, 1\}$. In this case g_1 outputs 1.

From this assumption the entries of the first row of the following table follows and the entries of other rows follows logically from the previous rows.

x_i	x_j	x_k	g_1	g_2	g_3	Comments
1	0	0	0	c_2	$\overline{c_3}$	Output of g_3 must change
0	0	1	0	$\overline{c_2}$	c_3	Output of g_2 must change
0	0	0	0	$\overline{c_2}$	$\overline{c_3}$	Follows from previous rows
0	1	1	1	c_2	c_3	Since only for this combination of values of g_2 and g_3, the circuit ouputs 1
0	1	0	0	c_2	$\overline{c_3}$	Follows from the previous row
1	1	1	1	c_2	c_3	
1	1	0	1	c_2	c_3	

Now we observe that entries of 6th and 8th rows of the above table are contradictory to each other as in both cases input combination to g_3 is identical but outputs of g_3 are different. So this topology is not possible.

Case 3 : We consider the topology of fig.3. Here g_1 can not be of type $(x^a \wedge y^b)^c$ since otherwise by choosing appropriate constant for x_k we can force the circuit output to a constant. Also g_1 can not be of type $\oplus$ or $\equiv$. This is because when $x_i = 1$ and $x_j = 1$ are assigned the circuit output can be made 0 by choosing an appropriate constant for x_k. So C can not be of this topology. This proves that

$$C_{B_2}(TH_{2,3}) \geq 4 \qquad (b)$$

Combining (a) and (b) we get
$C_{B_2}(TH_{2,3}) = 4$.

■

■

Theorem 4.1 *For $n \geq 3$, $TH_{2,n}$ requires circuits of size at least $(2n-2)$.*

Proof. We prove it by induction.
The basis case fo $n = 3$ follows directly from Fact 4.1. The induction step is identical to that of Theorem 2.7 of Chapter 2.

■

4.1.2 $MN_{k,l}$ class of symmetric functions

In this section we derive a lower bound for the class of symmetric Boolean functions considered by Zwick[58] for which he obtained a $4n - O(1)$ lower bound over U_2 basis. The bases considered by us are $\{\wedge, \neg\}$ and $\{\vee, \neg\}$ and the variation of gate elimination method used here is similar to that in [58].

Definition 4.1 *If β is a circuit and A is a gate in β then $d_\beta(A)$ denotes outdegree of A in β, $res_\beta(A)$ denotes the function computed at A and $d_1(\beta)$ denotes number of variables whose outdegree in β is exactly 1 and which are either feeding to two-input gates or to $\neg$-gates having single fanout.*

It is to be noted that the meaning of $d_1(\beta)$ in our case is slightly different from that in [58].

Simplification steps : A simplification step in a circuit(β) is one of the following.
1. If an internal $\wedge$-gate A of β is fed by constant 0 at one of its inputs then this gate is removed from β and the gates originally fed by A are now directly fed by 0.
2. If an internal $\wedge$-gate A of β is fed by constant 1 at one of its inputs then this gate is removed from β and the gates originally fed by A are now directly fed by the other input of A.
3. If an internal $\neg$-gate of β is fed by constant($c \in \{0,1\}$) at one of its inputs then this gate is removed from β and the gates originally fed by this $\neg$-gate are now directly fed by constant $\bar{c}$.
4. If a gate G is fed by a $\neg$-gate A, which in turn is fed by another $\neg$-gate B, then A is removed from β and G is fed directly by the input of B.
5. If an internal $\wedge$-gate A is fed by the same function at both the inputs, then this gate is removed from β and the gates originally fed by its fanout are fed by one input of A.
6. Any internal gate which is not feeding any other gate of β is removed.
7. If the inputs to an internal $\wedge$-gate A of β represent the functions f and $f \wedge g$, then A is removed and the gates originally fed by it are now fed by the line carrying $f \wedge g$ function.
8. If the inputs to an internal $\wedge$-gate A of β represent the functions f and $f \vee g$, then A is removed and the gates originally fed by it are now fed by the line carrying f function.
9. If the inputs to an internal $\wedge$-gate A of β represent the functions f and $\bar{f} \wedge g$, then A is removed and the gates originally fed by it are now fed by 0.
10. If the input of a $\neg$-gate A of β feeds to another $\neg$-gate B, then A is removed and the gates fed by its output are now fed by the output of B.

After a simplification step the changed circuit γ contains at least one less number of gates, $d_1(\gamma) \geq d_1(\beta) - 1$, and it continues to compute the same function. We call a circuit ***simplified*** when no further simplication step can be applied on it.

We work with the same class of functions as was defined in [58].

Definition 4.2 *The sets $S(n)$, $M_k(n)$, $N_l(n)$, $MN_{k,l}(n)$ are defined as follows :*

(1) $f \in S(n)$ if and only if $f(x_1, \ldots, x_n)$ depends only on $\sum_{i=1}^{n} x_i$. Functions belonging to $S(n)$ are called *symmetric functions*. If $f \in S(n)$ and $f(x_1, \ldots, x_n) = v_k$,

where $k = \sum_{i=1}^{n} x_i$, we associate with f the binary word $v(f) = v_0 v_1 \cdots v_n$. The word $v(f)$ is called the value vector of f.

(2) $f \in M_k(n)$ if and only if $f \in S(n)$ and every restriction of f to a subset of k variables is not constant. It is easy to see that $f \in M_k(n)$ if and only if $v(f)$ does not have a constant subword (i.e., $000\cdots$ or $111\cdots$) of length $k+1$.

(3) $f \in N_l(n)$ if and only if $f \in S(n)$ and every restriction of f to a subset $\{y_1, \ldots, y_n\}$ of f's variables is not linear, i.e., not $y_1 \oplus \cdots \oplus y_l$ or its complement. It is easy to see that $f \in N_l(n)$ if and only if $v(f)$ does not have an alternating subword (i.e., $0101\cdots$ or $1010\cdots$) of length $l+1$.

(4) $MN_{k,l}(n) = M_k(n) \cap N_l(n)$. In other words, $f \in MN_{k,l}(n)$ if and only if $v(f)$ does not have a constant subword of length $k+1$ or an alternating subword of length $l+1$.

The sets $MN_{k,1}(n)$ for $n \geq k \geq 1$ and $MN_{1,l}(n)$ for $n \geq l \geq 1$ are empty. The first non-empty set of these classes of functions is $MN_{2,2}(n)$. We will consider $k, l > 2$. Clearly, if $f \in MN_{k,l}(n)$ for some $n > k, l$ then every restriction of f obtained by fixing the value of one variable belongs to $MN_{k,l}(n-1)$. Also, for $n > k+1, l$ and a function $f \in MN_{k,l}(n)$, by fixing the values of any two variables we cannot restrict the function to a constant since the length of the value vector of the induced function can be at least $k+1$. For the same reason it is not possible by fixing the values of any two variables to make the induced function independent of some other variable.

Lemma 4.2 *Let β be a circuit over the complete basis $\{\wedge, \neg\}$ that computes a function $f \in MN_{k,l}(n)$ for $n > k+1, l$. There exists a circuit δ which computes a function $f' \in MN_{k,l}(n-1)$ and satisfies $[C(\delta) - d_1(\delta)] \leq [C(\beta) - d_1(\beta)] - 6$.*

Proof. From β we first obtain a simplified circuit(γ), if β were not already simplified. After simplification, $[C(\gamma) - d_1(\gamma)] \leq [C(\beta) - d_1(\beta)]$, because we have noted that at each simplification step the circuit size decreases by 1 and d_1 can decrease by at most 1.
We prove the lemma by considering the following cases.
Case 1. For some variable x, $d_\gamma(x) \geq 3$.
Suppose three such gates be A, B and C. Clearly, none of them can be the output gate, since otherwise by fixing an appropriate constant for x (0 if the gate is $\wedge$-type and 0 or 1 if it is $\neg$-type) the circuit output can be forced. These gates cannot feed among themselves since γ is simplified. So, these gates must feed to some other gates.
Case 1.1. Each of A, B and C feeds distinct gates (fig. 4.1).
Among A, B and C at most one can be a $\neg$-gate (suppose it is C). So A and B are $\wedge$-gates. We assign 0 to x. So A, B, D, E and C can be deleted since they output constants. F can also be deleted and its other input is directly connected to the fanout lines. Note that if $d_\gamma(z) = 1$, then z cannot feed A, B, C, D, E or F in γ. Otherwise,

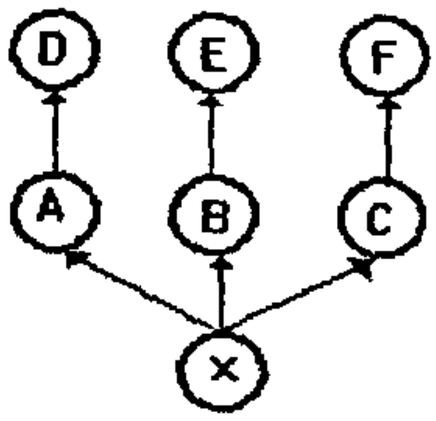


Fig 4.1

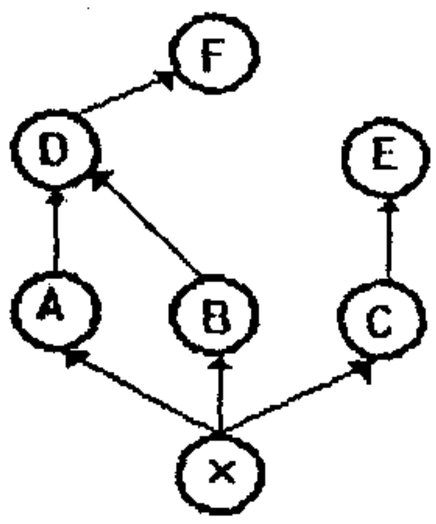


Fig 4.2

by choosing the right value for x we can make the output of γ independent of z and this is a contradiction as $n > k+1$. Thus, we get $C(\delta) = C(\gamma) - 6$ and $d_1(\delta) \geq d_1(\gamma)$ as required. If each of A, B and C is of type $\wedge$, we can prove in a similar fashion.

Case 1.2. A and B feed the same gate D (fig. 4.2).

Clearly, D cannot be the output gate since in that case by fixing a 0 at x, γ can be forced. Thus D must feed another gate (F). Assume that E and F are distinct. So by fixing 0 value at x all 6 gates can be eliminated. Now consider that E and F are same. C cannot be $\neg$ since γ is already simplified. So C is an $\wedge$-gate and thus F must feed some other gate. So, here also by putting $x = 0$, 6 gates are eliminated. Also, d_1 cannot decrease (for similar reason as described previously). Thus the lemma holds in this case.

Case 2. For any variable x, $d_\gamma(x) \leq 2$ and farthest gate from the output is an $\wedge$-gate fed by variables x and y.

The outdegrees of x and y must 2 (since if it were less we could have assigned a value to one of them and make the output independent of the other and if it is more it will fall under Case 1).

Case 2.1. At least one among x and y (say x) feeds to a $\neg$-gate having multiple fanout (fig. 4.3).

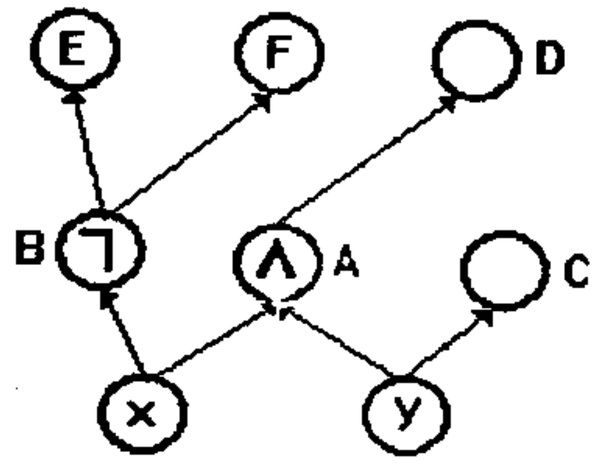


Fig 4.3

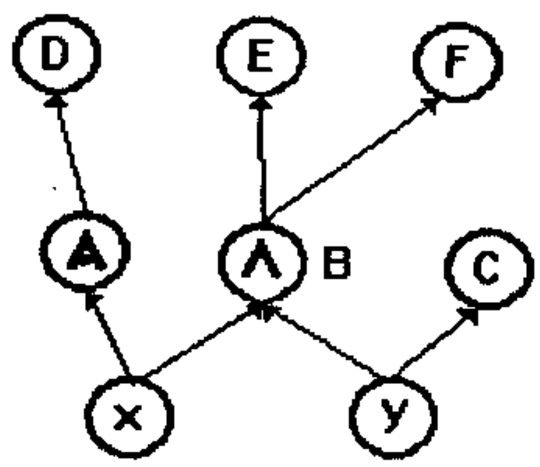


Fig 4.4

Since γ is simplified E and F must be $\wedge$-gates and none of them can be the same gate as D. We put $x = 0$ to delete A, B, D, E and F. If for some variable z, $d_\gamma(z) = 1$, z cannot feed D, E or F. Also $d_\delta(y) = 1$. Therefore, $C(\delta) \leq C(\gamma) - 5$ and $d_1(\delta) \geq d_1(\gamma) + 1$, as required.

Case 2.2. $d_\gamma(B) \geq 2$ (fig. 4.4).

This case is similar to Case 2.1.

Case 2.3. B feeds a $\neg$-gate (E) (fig. 4.5).

F must be of type $\wedge$ since γ is simplified. Assume first that F is different from A or D. In this case the lemma holds by putting $x = 0$ since if for some variable z, $d_\gamma(z) = 1$ it cannot feed F. If F is same as either A or D, we put $y = 0$ and the lemma holds in that case.

Case 2.4. B feeds an $\wedge$-gate (E) (fig. 4.6).

A or D cannot be E or F as γ is simplified. So by putting $x = 0$ the lemma holds in this case.

Case 3. The farthest gate from the output is a $\neg$-gate B fed by variable x.

If $d_\gamma(B) \geq 3$, we can handle in the same manner as Case 1. So we now consider the case where $d_\gamma(B) \leq 2$. Since B is farthest from the output, the gate fed by it (say E),

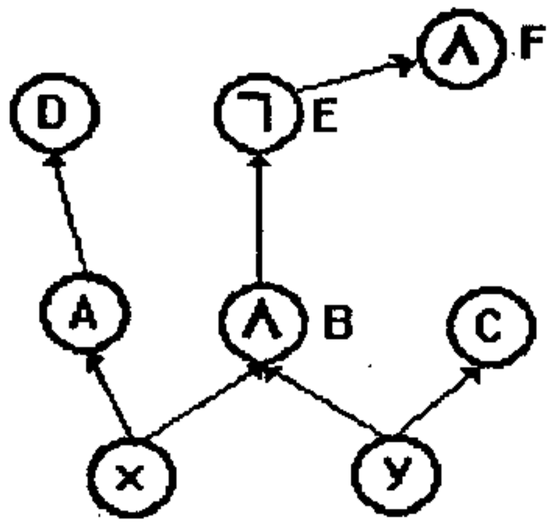


Fig 4.5

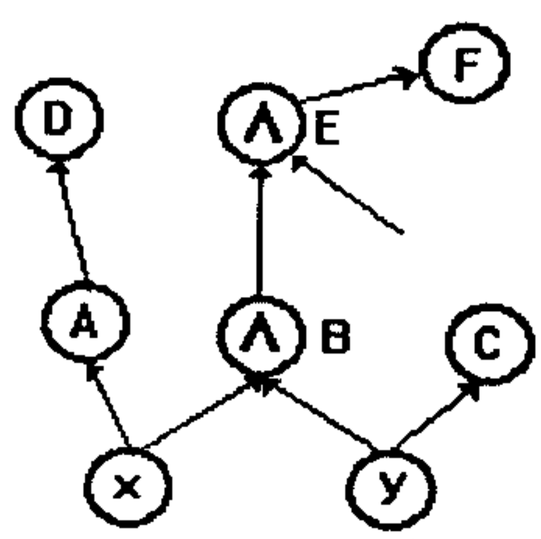


Fig 4.6

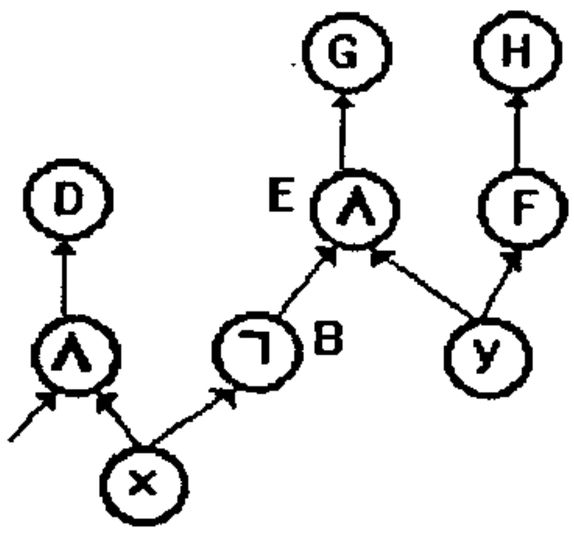


Fig 4.7

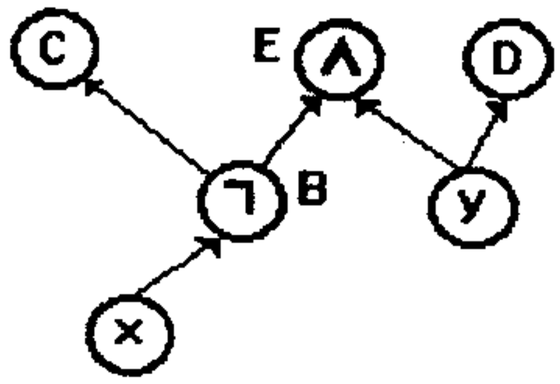


Fig 4.8

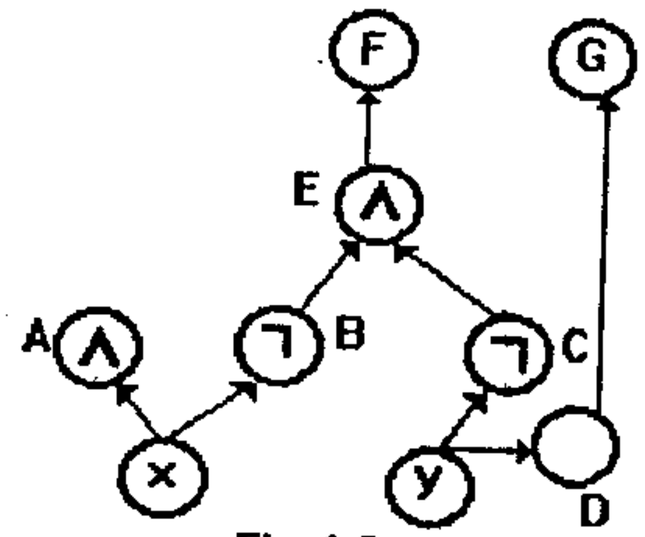


Fig 4.9

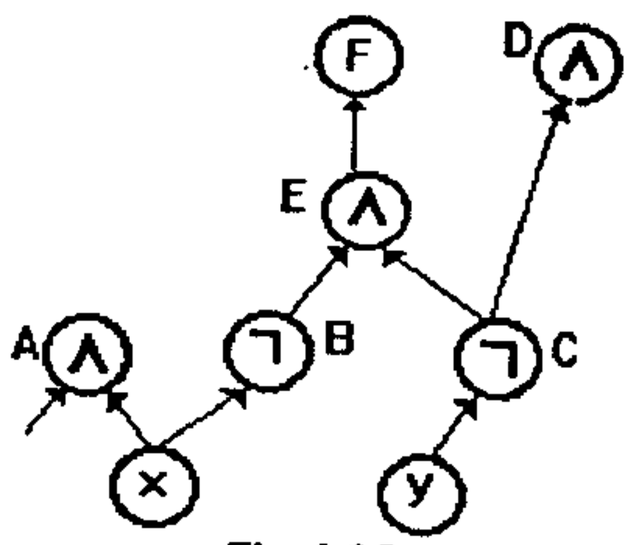


Fig 4.10

which must be of type $\wedge$, is fed either directly by an input node y having fanout 2 or by a $\neg$-gate that is fed by another variable y (if it is an $\wedge$-gate we have already handled it in Case 2.).

Case 3.1. E is fed by a variable y and outdegree of x is 2 (fig. 4.7).
If number of fanouts of B is 2 or more, we can put $x = 1$ and can easily show that the lemma holds. By putting $y = 0$, 5 gates can be eliminated (E, G, F, H and B). If G and H is the same gate the gate fed by it can be eliminated. Also, $d_\delta(x) = 1$. Thus increase in the count of d_1 is at least 1. So, the lemma holds in this case.

Case 3.2. Outdegree of x is 1 and E is fed by a variable y. (fig. 4.8).
B must have multiple fanout otherwise y can force the circuit to become independent of x which is not possible. This case can be handled in a similar fasion as Case 2, by considering B as a variable taking complementary values to that of x.

Case 3.3. E is fed by another $\neg$-gate which in turn is fed by a variable y (fig. 4.9, 4.10, 4.11, 4.12).
Clearly in the first 3 cases (fig. 4.9, 4.10, 4.11) the lemma can be shown to hold by setting $y = 1$. For the remaining case, we can handle in an analogous fashion as Case 2. So in each of the subcases we obtained a circuit that satisfies the conditions required by the lemma and thus the proof is complete. ∎

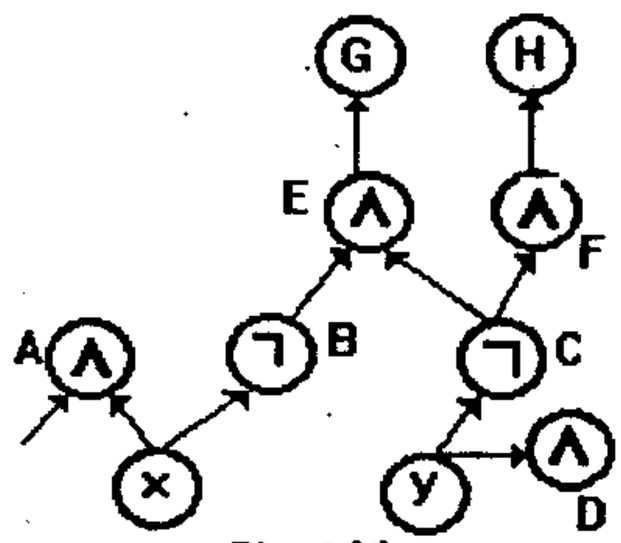


Fig 4.11

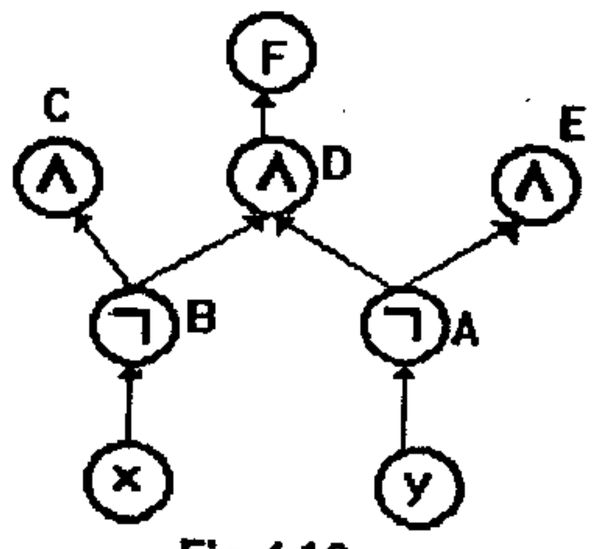


Fig 4.12

Similar result can also be proved over the complete basis $\{\vee, \neg\}$.

Lemma 4.3 *Let β be a circuit over the complete basis $\{\vee, \neg\}$ that computes a function $f \in MN_{k,l}(n)$ for $n > k+1, l$. There exists a circuit δ which computes a function $f' \in MN_{k,l}(n-1)$ and satisfies $[C(\delta) - d_1(\delta)] \leq [C(\beta) - d_1(\beta)] - 6$.*

The following lemma is similar to a result in [58].

Lemma 4.4 *[58] If $f \in M_k(n)$ and β is a simplified circuit computing f, then $d_1(\beta) < k$.*

Proof. Suppose on the contrary that $d_\beta(x_1) = \ldots = d_\beta(x_k) = 1$. If x_i feeds an $\wedge$-gate denote it by A_i, otherwise if it is a $\neg$-gate denote the unique gate fed by this gate by A_i (it must be of type $\wedge$ since β is simplified) for $i \leq k$. Since β is a simplified circuit the second input of A_i is not a constant. Denote by V_i, the set of variables on which this second input depends. Since by assigning appropriate values to the variables of V_i we can block x_i and thus obtain a constant restriction, we immediately get that $|V_i| \geq n - k + 1$. Thus each V_i contains at least one variable from the set $\{x_1, \ldots, x_k\}$. Denote one such variable by $x_{\pi(i)}$. For each $1 \leq i \leq k$ there exists a directed path in β from $x_{\pi(i)}$ to A_i and therefore also from $A_{\pi(i)}$ to A_i. But this is a contradiction since it implies a directed circuit in β. ∎

Theorem 4.2 *If $f \in MN_{k,l}(n)$, then $C(f) \geq 6(n-m)+(m-k)$ over the basis $\{\wedge, \neg\}$ as well as $\{\vee, \neg\}$, where $m = max\{k+1, l\}$ is fixed.*

Proof. We prove by induction on n that if β is a circuit which computes $f \in MN_{k,l}(n)$ then $[C(\beta) - d_1(\beta)] \geq 6(n-m)+(m-k)$.
The basis for $n = m$ is settled by the fact that $C(\beta) \geq m-1$, $d_1(\beta) < k$ (by Lemma 4.4).
The induction step follows immediately from Lemma 4.2. ∎

4.2 Using the communication complexity approach

4.2.1 Graph connectedness function

In this section we present a simple upper bound of monotone depth complexity of a monotone function using the framework of communication complexity.

Problem 4.1 GRAPHCON : *Given the adjacency matrix of an undirected graph $G = (V, E)$, with n vertices, test whether G is connected or not (a graph is said to be connected if between every pair of vertices there exsits a path).*

Theorem 4.3 $d_m(\text{GRAPHCON}) = C(R^m[\text{GRAPHCON}]) \leq (\lceil \log n \rceil)^2 + 2 \cdot \lceil \log n \rceil$.

Proof. We present a protocol for $R^m[GRAPHCON]$. Clearly, minterms correspond to spanning trees and maxterms correspond to $V_1 \times V_2$ for some partition of V into two non-empty sets V_1 and V_2. Assume, at the beginning of the game, both the players choose a fixed vertex s. Player-II now identifies in which partition (V_1 or V_2), s lies (suppose it is V_1) and sends the identification of any vertex t from the other partition (here V_2) to player-I. Since there are n nodes in the graph, this communication requires $\lceil \log n \rceil$ bits. After this, we follow the protocol presented in [23] for *st-connectivity* function. Player-II now considers the maxterm as an (s,t)-cut and assumes a coloring $q : V \rightarrow \{0,1\}$ where $q(v) = 0$ iff $v \in V_1$, while player-I identifies the (s,t)-path from the minterm. An edge in the intersection between the (s,t)-path and the (s,t)-cut is just a bichromatic edge. When the two players agree upon this bichromatic edge the game stops.

The players now perform a binary search for this edge along the (s,t)-path. The protocol consists of $\lceil \log n \rceil$ stages. Player-I sends the identification of the middle vertex v in the current portion p of the (s,t)-path using $\lceil \log n \rceil$ bits. Player-II responds with the color of v. The current path for the next stage is the portion of p where a bichromatic edge is ensured to exist. When the path is a single edge the protocol terminates. ∎

4.3 A combinatorial lemma

Alon and Boppana[4] showed that the number of AND gates and OR gates in a monotone circuit can always be somewhat balanced without increasing the complexity of the circuit. In this section we show that a similar result can be proved for circuits over some complete bases.

Lemma 4.5 *Let f be a function of n Boolean variables and suppose there exists a circuit with input literals from $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$ and gates from $\{\wedge, \vee\}$ computing f and contains k $\wedge$-gates ($\vee$-gates). Then there exists a circuit computing f containing k $\wedge$-gates ($\vee$-gates) and at most $(k+1)(2n-2)+({}^{k+1}C_2)+k-n$ $\vee$-gates ($\wedge$-gates).*

Proof. We consider a straight-line program for computing f and suppose $f_1, \ldots, f_k$ be the k-outputs of the k $\wedge$-gates ($\vee$-gates), in the order of computation. We first prove, by induction on i, that there is a circuit that computes $f_1, \ldots, f_i$ containing i $\wedge$-gates ($\vee$-gates) and at most $i(2n-2)+({}^{i}C_2)+(i-1)$ $\vee$-gates ($\wedge$-gates). For $i = 1$, f_1 is and AND (OR) of two operands, each of which is either a constant or an OR (AND) of a subset of $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$. These two operands can be computed by at most $(2n-2)$ $\vee$-gates ($\wedge$-gates). So the basis case is settled.
In the induction step, the function f_i is an AND (OR) of two operands, each of which is either a constant or an OR (AND) of a subset of $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\} \cup \{f_1, \ldots, f_{i-1}\}$. These two operands can be computed with at most $2n+i-2$ $\vee$-gates ($\wedge$-gates). and since $(i-1)(2n-2)+({}^{i-1}C_2)+(i-2)+(2n+i-2) = i(2n-2)+({}^{i}C_2)+(i-1)$, the induction step is proved.
So $f_1, \ldots, f_k$ can be computed with a monotone circuit containing k $\wedge$-gates ($\vee$-gates) and at most $k(2n-2)+({}^{k}C_2)+(k-1)$ $\vee$-gates ($\wedge$-gates). The function f is either a constant or an OR (AND) of a subset of $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\} \cup \{f_1, \ldots, f_k\}$ which can be computed by at most $n+k-1$ additional $\vee$-gates ($\wedge$-gates). Hence the result follows. ∎

It is to be noted that similar results can be proved over the bases $\{\wedge, \neg\}$ and $\{\vee, \neg\}$.

Chapter 5

Conclusion

We have presented a brief survey on circuit complexity and some of our results related to this area. In the survey, we have focussed on the different approaches for proving size and depth lower bounds. The approaches described are mainly of two types : *Top-down* and *Bottom-up*. In the Top-down approach progress of computation is measured from the output down to the inputs, while in the Bottom-up approach it is viewed in the reverse fashion. The gate elimination method, the approximation method and the method of random restrictions are of Bottom-up type. An example of Top-down method is the communication complexity approach. The fusion method, which borrows ideas from the model theory, is an exception and views the computation in a more global way. We have not described the works in the following areas of circuit complexity : Relativization and complexity hierarchy, Parallel random access computation, Synchronous, planar and probabilistic circuit model, Arithmetic circuits, Branching program model etc. The book by Wegener[55] may be referred for getting acquainted in these topics.

Bibliography

[1] M. Ajtai, Σ_1^1-formulae on finite structures, *Ann. Pure Appl. Logic* **24** (1983) 1-48.

[2] M. Ajtai and M. Ben-Or, A theorem on probabilistic constant depth circuits, in: *Proc. 16th Ann. ACM Symp. on Theory of Computing* (1984) 471-474.

[3] M. Ajtai, J. Komlós and E. Szemerédi, An $O(n \log n)$ sorting network, in: *Proc. 15th Ann. ACM Symp. on Theory of Computing* (1983) 1-9; revised version in *Combinatorica* 3(1) (1983) 1-19.

[4] N. Alon and R.B. Boppana, The monotone circuit complexity of Boolean functions, *Combinatorica* 7(1) (1987) 1-22.

[5] A.E. Andreev, On a method for obtaining lower bounds for the complexity of individual monotone functions, *Dokl. Acad. Nauk SSSR* 282 (5) (1985) 1033-1037 (in Russian); English translation in: *Soviet Math. Dokl.* 31(3) (1985) 530-534.

[6] A.E. Andreev, On a family of Boolean matrices, em Vestnik Moskov. Univ. Mat. 41(2) (1986) 97-100 (in Russian); English translation in: *Moscow Univ. Math. Bull.* 41(2) (1986) 79-82.

[7] A.E. Andreev, On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes, *Vestnik Moscov. Univ. Mat.* 42(1) (1987) 70-73 (in Russian); English translation in: *Moscow. Univ. Math. Bull.* 42(1) (1987) 63-66.

[8] R. Barua, Restrictions of Boolean functions and Ramsey property, *Proc. of 1st National Seminar on Theoretical Computer Science* (ed. P.S. Thiagarajan) (1991).

[9] S. Ben-David, M. Karchmer and E. Kushilevitz, On ultrafilters and NP, in: *Proc. of the 9th. Ann. Symp. on Structure in Complexity Theory*, (1994) 97-105.

[10] S.J. Berkowitz, On some relationships between monotone and non-monotone circuit complexity, Tech. Report, Comput. Sci. Dept., Univ. of Toronto, (1982).

[11] N. Blum, A Boolean function requiring $3n$ network size, *Theoret. Comput. Sci.* 28 (1984) 337-345.

[12] R.B. Boppana, Amplification of probabilistic Boolean formulas, in: S. Micali, ed., *Advances in Computer Research, Vol. 5:Randomness and Computation*, JAI Press, Greenwich, CT.

[13] R.B. Boppana and M. Sipser, The Complexity of finite functions, in: *The Handbook of Theoretical Computer Science*, (J. van Leeuwen, ed.), Elsevier Science Publishers B.V., (1990) 759-804.

[14] A.K. Chandra, L.J. Stockmeyer and U. Vishkin, A complexity theory for unbounded fan-in parallelism, in: *Proc. 23rd Ann. IEEE Symp. on Found. of Comp. Sci*, (1982) 1-12.

[15] S.A. Cook, The complexity of theorem proving procedures, *Proc. of the 3rd. Ann. ACM Symp. on Theory of Computing* (1971) 151-158.

[16] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, in: *Proc. 19th Ann. ACM Symp. on Theory of Computing* (1987) 1-6.

[17] P.E. Dunne, Lower bounds on the monotone network complexity of threshold functions, in: *Proc. 22nd Ann. Allerton Conf. on Communication, Control and Computing* (1984) 911-920.

[18] J. Edmonds, Paths, trees and flowers, *Canad. J. Math.* 17 (1965) 449-467.

[19] M. Furst, J. Saxe and M. Sipser, Parity, circuits and the polynomial time hierarchy, *Math. Systems Thoery* 17 (1984) 13-27.

[20] J. Hartmanis and R.E Stearns, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* 117 (1965) 285-306.

[21] J. Håstad, Computational limitations of small-depth circuits, *ACM Doctoral Dissertation award winner, originally presented as author's Ph.D thesis, MIT* (1986), The MIT Press, Cambridge, Massachusetts.

[22] B. Kalyanasundaram and G. Schnitger, The probabilistic communication complexity of set intersection, in: *Proc. of the 2nd. Ann. Conf. on Structure in Complexity Theory*, (1987) 41-49.

[23] M. Karchmer, Communication complexity : A new approach to circuit depth, *ACM Distinguished Doctoral Dissertation award winner, originally presented as author's Ph.D thesis, Hebrew University* (1988), The MIT Press, Cambridge, Massachusetts.

[24] M. Karchmer, On proving lower bounds for circuit size, in: *Proc. of the 8th Ann. Symp. on Structure in Complexity Theory*, (1993) 112-118.

[25] M. Karchmer and A. Wigderson, Monotone circuits for connectivity require superlogarithmic depth, in: *Proc. of the 20th Ann. ACM Symp. on Theory of Computing*, (1988) 539-550.

[26] M. Karchmer and A. Wigderson, Characterizing non-deterministic circuit size, in: *Proc. of 25th Ann. ACM Symp. on Theory of Computing*, (1993) 532-539.

[27] M. Karchmer and A. Wigderson, On span programs, in: *Proc. of the 8th Ann. Symp. on Structure in Complexity Theory*, (1993).

[28] R.M. Karp and R. Lipton, Turing machines that take advice, *Enseign. Math.* 28 (1982) 191-209.

[29] V.M. Khrapchenko, A method for determining lower bounds for the complexity of Π-schemes, *Mat. Zametki* 10(1) (1971) 83-92 (in Russian); English translation in: *Math. Notes* 10(1) (1971) 474-479.

[30] M. Klawe, W.J. Paul, N. Pippenger, M. Yannakakis, On monotone formulae with restricted depth, *Proc. of the 16th Ann. ACM Symp. on Theory of Computing* (1984) 480-487.

[31] E.A. Lamagna and J.E. Savage, Combinational complexity of some monotone functions, in: *Proc. 15th Ann. IEEE. Symp. on Switching and Automata Theory* (1974) 140-144.

[32] Lupanov, A method of circuit synthesis, *Izv. VUZ Radiofiz*, 1 (1958) 120-140.

[33] K. Mehlhorn and Z. Galil, Monotone switching circuits and Boolean matrix product, *Computing* 16 (1976) 99-111.

[34] D.E. Muller, Complexity in electronic switching circuits, *IRE Trans. Electronic Computers* 5 (1956) 15-19.

[35] D.E. Muller and F.P. Preparata, Bounds to complexities of networks for sorting and switching, *J. Assoc. Comput. Mach.* 22(2) (1975) 195-201.

[36] W.J. Paul, A $2.5n$ lower bound on the combinational complexity of Boolean functions, *SIAM J. Comput.* 6(3) (1977) 427-443.

[37] N. Pippenger and M.J. Fischer, Relations among complexity measures, *J. Assoc. Comput. Mach.* 26 (1979) 361-381.

[38] J. Radhakrishnan and S. Saluja, Interactive Proof Systems (Lecture Notes), Technical Report No. TCS-95/4, TCS Group, TIFR Bombay, (February 1995).

[39] R. Raz and A. Wigderson, Monotone circuits for matching require linear depth, in: *Proc. 22nd. Ann. ACM Symp. on Theory of Computing* (1990).

[40] A.A. Razborov, Lower bounds on the size of bounded depth networks over a complete basis with logical addition, *Mat. Zametki* 41(4) (1987) 598-607 (in Russian); English translation in: *Math. Notes* 41(4) (1987) 333-338.

[41] A.A. Razborov, A lower bound on the monotone network complexity of the logical parmanent, *Mat. Zametki* 37(6) (1985) 887-900 (in Russian); English translation in: *Math. Notes* 37(6) (1985) 485-493.

[42] A.A. Razborov, Lower bounds for the monotone complexity of some Boolean functions, *Dokl. Ak. Nauk. SSR*, 281 (1985) 798-801 (in Russian); English translation in: *Sov. Math. Dokl.* 31 (1985), 354-357.

[43] A.A. Razborov, Lower bounds on the size of switching-and-rectifying networks for symmetric Boolean functions, *Math. Notes of the Academy of Sciences of the USSR*, 48(6) (1990) 79-91.

[44] A.A. Razborov, On the method of approximations, in: *Proc. of 21st Ann. ACM Symp. on Theory of Computing*, (1989), 167-176.

[45] N.P. Redkin, Proof of minimality of circuits consisting of functional elements, *Sys. Th. Res.*, 23 (1973) 85-103.

[46] J.E. Savage, Computational work and time on finite machines, *J. Assoc. Comput. Mach.*, 19(4) (1972) 660-674.

[47] C.E. Shannon, The synthesis of two terminal switching circuits, *Bell Systems Tech. J.* 28(1) (1949) 59-98.

[48] M. Sipser, On polynomial vs. exponential growth, Unpublished manuscript, 1981.

[49] M. Sipser, A topological view of some problems in complexity theory, *Colloquia Mathematica Societatis János Bolyai* 44 (1984) 387-391.

[50] M. Sipser, The history and status of the P versus NP question, in: *Proc. of the 24th Ann. ACM Symp. on Theory of Computing* (1992) 603-618.

[51] R. Smolensky, Algebraic methods in the theory of lower bounds for Boolean circuit complexity, in: *Proc. 19th Ann. ACM Symp. on Theory of Computing* (1987) 77-82.

[52] P.M. Spira, On time-hardware complexity tradeoffs for Boolean functions, in: *Proc. 4th Hawaii Symp. on Systems Sciences* (Western Periodicals Company, North Hollywood, 1971) 525-527.

[53] L.G. Valiant, Short monotone formulae for the majority function, *J. of Algorithms*, 5 (1984) 363-366.

[54] I. Wegener, Relating monotone formula size and monotone depth of Boolean functions, *Inform. Process. Lett.*, 16 (1983) 41-42.

[55] I. Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner series in Computer Science (Teubner, Stuttgart/Wiley, Chichester, 1987).

[56] A. Wigderson, The fusion method for lower bounds in circuit complexity, *Combinatorics, Paul Erdös is Eighty*, (Vol 1), Miklós, Sós and Szönyi (eds.), Bolyai Math. Soc. (1993), 453-468.

[57] A.C. Yao, Separating the polynomial-time hierarchy by oracles, in: *Proc. 26th Ann. IEEE Symp. on Foundations of Computer Science* (1985) 1-10.

[58] U. Zwick, A $4n$ lower bound on the combinatorial complexity of certain symmetric Boolean functions over the basis of unary dyadic Boolean functions, *SIAM J. on Computing*, 20(3) (1991) 499-505.