Boolean Function Complexity

Advances and Frontiers

Stasys Jukna

Preliminary Version[1]

Vilnius/Frankfurt

2009

[1]Compiled: September 26, 2009. Typeset using amsbook.cls

Contents

Preface

Computational complexity theory is the study of the inherent hardness or easiness of computational tasks. Research in this theory has two main strands.

One of these strands—called also *structural complexity*—deals with "high level" complexity questions: is space a more powerful resource than time? Does randomness enhance the power of efficient computation? Is it easier to verify a proof than to construct one? So far we do not know the answers to any of these questions; thus most results in "high level" complexity are *conditional* results that rely on various unproven assumptions, like $\mathbf{P} \neq \mathbf{NP}$. While many interesting connections have been established between different computational problems and computational resources, and many beautiful and important results have been achieved, the major open problems here remain widely open.

The second strand of research in complexity theory—called also *concrete complexity* or *circuit complexity*—deals with establishing concrete lower bounds, that is, lower bounds on the computational complexity of specific problems, like multiplication of numbers or their factorization. This is essentially a "low level" study of computation; it typically centers around particular models of computation such as decision trees, boolean formulas, restricted classes of boolean circuits, and the like. In this line of research *unconditional* lower bounds are established which rely on no unproven assumptions.

Research in circuit complexity began about 60 years ago starting from a seminal work of Claude Shannon. A burst of activity in circuit complexity exploded about 25 years ago with first exponential lower bounds for some circuit models, like bounded depth circuits, monotone circuits, restricted branching programs, etc. Since then there has been steady progress made over the years using a range of techniques from combinatorics, algebra, analysis, and other branches of mathematics. In fact, circuit complexity is the “most combinatorial” part of the whole computer science.

The focus of this book[1] is on the second stream: concrete, "low-level" complexity, with a special focus on lower bounds. I give self-contained proofs of a wide range of unconditional lower bounds for interesting and important models of computation, covering many of the "gems" of the field that have been discovered over the past several decades, right up to results from the last year or two.

The book is not an all-inclusive *historical* survey—bibliographical references are only given for results that are actually described here. Instead, the book is an almost all-inclusive survey of known lower bounds *techniques* with full proofs.

The reason to write this book was threefold.

First, 20 years passed since the well known books on circuit complexity of Savage (1976), Nigmatullin (1983), Wegener (1987) and Dunne (1988), as well as a famous

[1] This is a draft version. Any critics, detected errors in proofs, missing references, comments on topics worth to be discussed, etc. are more than welcome!

survey paper of Boppana and Sipser (1990), were written. It came the time to summarize the development in circuit complexity during these two decades.

Second, newly written nice books on computational complexity are mainly devoted to research in the first direction—structural complexity and, quite naturally, the treatment of "low level" complexity is there only fragmentary.

Finally, the discovery of "natural proofs" waked an impression that almost nothing is possible in this field. Roughly speaking, this result says that it is (apparently) impossible to separate complexity classes like **P** and **NP** using properties of boolean functions that are easily verifiable and are shared by random functions. As such, this is a serious warning: circuit lower bounds are *indeed* very hard to prove. It also says that, like in other in other fields of mathematics, too general and too constructive ideas cannot solve too difficult problems. But separating **P** from **NP** is not the main goal of circuit complexity—this will be probably done by a cute diagonalization.

Circuits and Turing machines are very different models: the former is *non-uniform*, and hence, much stronger. According to Leonid Levin, the co-founder of **NP**-completeness phenomenon, Andrey Nikolaevich Kolmogorov, one of the greatest mathematicians of the last century, even suspected that all **NP** can be apparently done by linear size circuits! Decades passed, and this belief ist still not refuted. There are even some indications that this prediction (or something similar) could be indeed true. Say, Mayer auf der Heide (1984) shows that, for each n, the n–dimensional knapsack problem is solvable in $n^4 \log n$ time. Another indication is given by Allender and Koucky (2008): in a class of constant-depth threshold circuits, some boolean functions cannot have circuits of polynomial size, if they do not have such circuits of size $n^{1+\varepsilon}$ for an arbitrary small constant $\varepsilon > 0$. These (and some other) indications show that circuits of superlinear size may indeed accumulate an unexpected power. So large that current mathematics is unable to engage such circuits. So, the goals of circuit complexity are much more "pragmatic:" prove lower bounds in—restricted, but practically important—circuit models. When trying to do this for harder and harder models many nice mathematical ideas emerged, and my goal was to describe some of them here.

Just like proving lower bounds is a self-defeating task—prove that this was hard to prove—the goals of this book are somewhat self-defeating as well. My goal was:

- to cover main developments in circuit complexity during the last two decades, but also to be fairly compact;
- to give full proofs of core results, but also to be as concise and as intuitive as possible.
- to write a text which can be relatively easily grasped by graduate students, but remains of some interest for researchers, as well.

I've done my best to achieve a fair balance between these contradicting goals. The seek for the balance has also influenced the choice of the material: the focus is on classical models of circuits—results on their randomized, quantum or algebraic versions receive less attention here. My goal was to give a "big picture" of existing most powerful lower bound methods for classical circuit models, in a hope that the reader will be motivated to find a new one. Many open problems, marked as "Research Problem", are mentioned along the way.

The text is *self-contained*. It assumes a certain mathematical maturity at an undergraduate level but *no* special knowledge in theory of computing. Like in combinatorics or in number theory, the problems here are usually quite easy to state and explain, even for the layman. Their solutions often require a cute idea, but rarely an involved

mathematics. The book may be used for a graduate course on circuit complexity or as a supplement material in a more general course on computational complexity. The material is split into mostly *independent* chapters, each devoted to a particular model for computing boolean functions, so that the reader can choose her/his own order to follow the book. The order of chapter has nothing to do with the "importance" of the circuit model and/or techniques dealt with in them—it just reflects the chronological order in which the results about these models were achieved. Say, the last chapter "Propositional Proof Complexity" (Chapter 18) is nowadays one of the "hottest" places of action.

Some features of the book (as I see them) include:

- It is the first book covering the happening in circuit complexity during the past 20 years. A part of this happening—the communication complexity—was already covered in an excelent book by Nisan and Kushilevitz (1997).
- It includes some topics, like graph complexity or method of finite limits, that are not known well enough even for specialists.
- Gives full and intuitive proofs of basic lower bounds.
- Gives new proofs of classical results, like lower bounds for monotone and for constant-depth circuits.
- Presents some topics never touched in existing books, like circuits with arbitrary boolean functions as gates.
- Relates the circuit complexity with one of the "hottest" nowadays topics – the proof complexity.

Two apologies may be in order. The first of them goes to students: although an attempt is made to keep the exposition as simple as possible, some proofs will still require a considerable effort to get them. But remember: original proofs were even more complicated. The second apology goes to purists: many of the estimations in our arguments can be numerically improved by making more careful computations. The reason for my carelessness was the desire to make the exposition as simple as possible. So, my stress is on arguments and ideas used in the lower bound proofs rather than on the numerical form of resulting bounds, unless the jump in the rate of growth is really important, like linear to quadratic, polynomial to super-polynomial.

..... to be finished .....

Frankfurt/Vilnius, Juli, 2009 Stasys Jukna

CHAPTER 1

Our Adversary – The Circuit

Boolean or switching functions $f : \{0,1\}^n \to \{0,1\}$ map each sequence of bits to a single bit 0 or 1. Simplest of such functions are the product $x \cdot y$, sum $x \oplus y$ mod 2, non-exclusive Or $x \vee y$, negation $\neg x = x \oplus 1$. The central problem of Boolean function complexity—the lower bounds problem—is: Given a boolean function how many these simplest operations do we need to compute the function on all input vectors? This is an extremal problem per se: how large boolean circuits for a given function must be? The problem lies on the border between mathematics and computer science: lower bounds themselves are of great importance for computer science but their proofs require techniques from combinatorics, algebra, analysis, and other branches of mathematics.

Mathematics is full of non-existence results. Circuit lower bounds are also non-existence results, only on a "low level." We restrict our world by circuits of reasonable (say, linear or polynomial) size, and ask if a given boolean function belongs to this world.

But is proving lower bounds important at all? Would it not be better to invest our energy into proving good *upper* bounds, that is, into the design of efficient circuits? Yes and no. Yes, because cute algorithms detect some "singularity" in a given problem making it efficiently solvable by a circuit. No, because lower bounds do just the same! They detect singularities making the problem unsolvable by any circuit. That is, proving upper bounds is a cooperative game with algorithms, whereas proving lower bounds is an adversary game against algorithms. A progress in any direction is a step towards the solution of the main problem of computer Science: understand what algorithms can and what they can not.

1.1. Circuit models

Before we start with proving lower bounds, let us first recall the most fundamental models for computing boolean functions.

Let Ω be a set of some boolean functions (elementary or basis operations). A *circuit* (or a *stright line program*) over the basis Ω is just a sequence $g_1, \ldots, g_m$ of boolean functions such that the first n functions are input variables $g_1 = x_1, \ldots, g_n = x_n$, and each subsequent g_i is an application $g_i = \varphi(g_{i_1}, \ldots, g_{i_d})$ of some basis function $\varphi \in \Omega$ (called the *gate* of g_i) to some previous functions.

That is, the value $g_i(a)$ of the ith gate g_i on a given input $a \in \{0,1\}^n$ is the value of the boolean function φ applied to the values $g_{i_1}(a), \ldots, g_{i_d}(a)$ computed at the previous gates. A circuit computes a boolean function (or a set of boolean functions) if it (or they) are among the g_i.

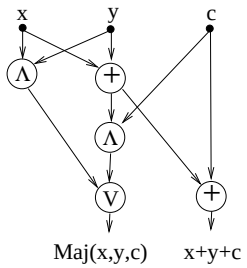


FIGURE 1. A circuit with five gates over the basis $\{\wedge, \vee, \oplus\}$ computing the sum $x \oplus y \oplus c$ of three bits modulo 2, and a carry bit $Maj(x, y, c) =$ 1 iff at least two of there input bits are 1's.

Each circuit can be looked at as a directed acyclic graph [1] whose fanin-0 nodes (those of zero in-degree) correspond to variables, and each other node v corresponds to a function φ from Ω. One (or more) nodes are distinguished as outputs. The value at a node is computed by applying the corresponding function to the values of the preceding nodes (see Fig. 1). The *size* of the circuit is the total number of its gates.

A *formula* is a circuit whose all gates have fanout at most 1. Hence, the underlying graph of a formula is a tree. The *leafsize* of a formula is the number of input gates, that is, the number of leaves in its tree, and the *depth* of a formula is the depth of its tree. Note that the only (but crucial) difference of formulas from circuits is that, in the later model, a result computed at some gate can be used many times with no need to recompute it again and again, as in the case of formulas.

A *DeMorgan circuit* is a circuit whose inputs a variables and their negation, and gates are fanin-2 AND and OR functions. That is, these are the circuits over the basis $\{\wedge, \vee, \neg\}$ where NOT gates are only applied to input variables. Such circuits are also called *circuits with tight negations*. It can be easily shown (do this!) that any circuits over $\{\wedge, \vee, \neg\}$ can be reduced to this form by at most doubling the total number of gates.

Circuits and formulas are "parallel" models: given an input vector x, we process some pieces of x in parallel and join the results by AND or OR gates. The oldest "sequential" model for computing boolean functions, introduced already in pioneering work of Shannon (1949) and extensively studied in the Russian literature since about 1950, is that of switching networks; a "modern" name for these networks is nondeterministic branching programs.

A *nondeterministic branching program* (or a *switching-and-rectifair network*) is a directed graph $G = (V, E)$ with two specified vertices $s, t \in V$, *some* of whose edges are labeled by variables x_i or their negations $\neg x_i$. A labeled edge is also called a *contact*. The graph may have multiple edges, i.e., several edges may have the same endpoints. The *size* of G is defined as the number of contacts (labeled edges).

Each input $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ switches the labeled edges on or off by the following rule: the edge, labeled by x_i, is switched on if $a_i = 1$ and is switched off if $a_i = 0$; the edge, labeled by $\neg x_i$, is switched on if $a_i = 0$ and is switched off if $a_i = 1$.

[1] The graphs of circuits are often drawn starting from the output gate(s) and going down to inputs. But then we must let trees grow from sky to the earth. I will therefore sometimes use a more "nature friendly" way in pictures, and will draw circuits starting from inputs.

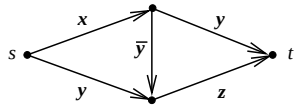


FIGURE 2. A switching network for the threshold-2 function $Th_2^3(x,y,z)$ in three variables, which outputs 1 iff $x+y+z \geq 2$.

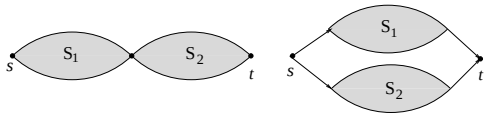


FIGURE 3. Sequential connection corresponds to AND, and parallel connection to OR.

A switching network G computes a boolean function in a natural way: it accepts the input a if and only if there is a path from s to t along which all edges are switched on by a. That is, each input switches the edges on or off, and we accept that input if and only if after that there is a nonzero conductivity between the vertices s and t (see Fig. 2).

It is important to note that switching networks include DeMorgan formulas as their special case. Namely, it can be easily shown that DeMorgan formulas correspond to a very special type of switching networks—so called Π-*schemes*—whose underlying graph consists of parallel-sequential components.

PROPOSITION 1.1. *Every DeMorgan formula can be simulated by a* Π*-scheme of the same size, and vice versa.*

PROOF. This can be shown by induction on the leafsize of a DeMorgan formula F. If F is a variable x_i or its negation $\neg x_i$, then F is equivalent to a switching network consisting of just one contact. If $F = F_1 \wedge F_2$ then, having switching networks S_1 and S_2 for subformulas F_1 and F_2, we can obtain a switching network for F by just identifying the target node of S_1 with the source node of S_2. If $F = F_1 \vee F_2$ then, having switching networks S_1 and S_2 for subformulas F_1 and F_2, we can obtain a switching network for F by placing these two networks in parallel by gluing their source nodes and their target nodes (see Fig. 3). □

Another special version of switching networks is the model of "deterministic branching programs". In the past decades, this model deserved much more attention than switching networks. The reason for this interest is that the logarithm of the number of nodes in such programs captures the space of deterministic Turing machines, and the model itself is easier to analyze.

A *deterministic branching program* for a given boolean function f in n variables $x_1,\ldots,x_n$ is a directed acyclic graph with one source node and two sinks, i.e., nodes of out-degree 0. The sinks are labeled by 1 (accept) and by 0 (reject). Each non-sink node has out-degree 2, and the two outgoing edges are labeled by the tests $x_i = 0$ and $x_i = 1$ for some $i \in \{1,\ldots,n\}$. Such a program computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ in a natural way: given an input vector $a \in \{0,1\}^n$, we start in the source node and follow the unique path whose tests are consistent with the

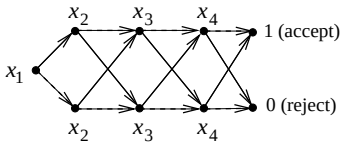


FIGURE 4. A deterministic branching program computing $x_1 \oplus x_2 \oplus x_3 \oplus x_4$. Each node is labeled by a variable x_i. Dashed arrows correspond to tests $x_i = 0$ and the remaining arrows to test $x_i = 1$. This program is very specific: along every path each variable is tested only once, and the variables are tested in the same order. Programs with these two restrictions are known as *ordered binary decision diagrams* (OBDDs).

corresponding bits of a; this path is the *computation* on a. This way we reach a sink, and the input a is accepted iff this is the 1-sink (see Fig. 4).

Thus, if we remove the 0-sink (together with all edges entering it) in a deterministic branching program, we obtain a switching network with two restrictions:

a. every node has fanout at most 2, and
b. the two edges leaving the same node must be labeled by a variable and its negation.

It is the second condition which makes such a network deterministic: every input vector has a unique computation path. By the same reason, general switching networks are also called "nondeterministic" branching programs: here one accepted input may have many accepting paths from s to t.

A *parity branching program* is a nondeterministic branching program with the "counting" mode of acceptance: an input vector a is accepted iff the number s-t paths consistent with a is odd.

Branching programs are also called in the literature *binary decision diagrams* or shortly BDDs. This term is especially often used in the circuit design theory as well as in other fields where branching programs are used to represent boolean functions. Be however warned that the term "BDD" in such papers is often used to denote much weaker model—that of OBDD, meaning oblivious read-once branching programs. These are deterministic branching programs of a very restricted structure as shown in Fig. 4: along every computation path all variables are tested in the same order, and no variable is tested more than once.

1.2. Random functions are complex

As mentioned above, we still cannot prove super-linear lower bounds for circuits with AND, OR and NOT gates. This is in sharp contrast with the fact, proved 60 years ago by Claude Elwood Shannon (1949), that most boolean functions require about $2^n/n$ elementary operations. His argument was the first application of counting arguments in boolean function complexity: count how many *different* boolean functions in n variables can be computed using a given number of elementary operations, and compare this number with the total number 2^{2^n} of all boolean functions.

Most of lower bounds in circuit complexity are asymptotic, that is, ignore the constant multiplicative factors. Moreover, boolean function $f : \{0,1\}^n \to \{0,1\}$ are parameterized by their number of variables n. Hence, under a boolean function f we actually understand an infinite sequence $\{f_n \mid n = 1,2,\ldots\}$ of boolean functions. So, the claim

"f requires $\Omega(\varphi(n))$ gates" means that there is an absolute constant $\varepsilon > 0$ such that, for infinitely many values of n, the function f_n cannot be computed using fewer than $\varepsilon \cdot \varphi(n)$ gates.

THEOREM 1.2 (Circuits). *Almost every Boolean function of n variables requires De-Morgan circuits of size $\Omega(2^n/n)$.*

PROOF. Let $F(n,t)$ be the number of circuits with n variables which have size $\leq t$. We will first show that

$$F(n,t) \leq ((2(t+2n)^2)^t . \tag{1.1}$$

Indeed, each gate in a circuit is assigned an AND or OR operator (2 possibilities) that acts on two previous nodes, and each previous node can either be a previous gate ($\leq t$ choices), or a variable or its negation ($\leq 2n$ choices) Thus each gate in a circuit has at most $2(t+2n)^2$ choices. Since we have t gates, (1.1) follows.

Notice that for $t = 2^n/(10n)$, the right-hand of (1.1) is approximately $2^{2^n/5}$ which is $\ll 2^{2^n}$. Since there are exactly 2^{2^n} Boolean functions of n variables, almost every Boolean function requires circuits of size large than $2^n/(10n)$. □

By this theorem, the average circuit complexity of boolean function in n variables is exponential in n. But, so far, nobody was able to prove that some *specific* boolean function requires more than $5n$ gates!

In the class of formulas (fanout-1 circuits) some boolean function require even more that $2^n/n$ leaves.

THEOREM 1.3 (Formulas). *Almost every boolean function of n variables requires De-Morgan formulas of leafsize $\Omega\left(2^n/\log n\right)$.*

PROOF. There are at most $2^{O(t)}$ binary trees with at most t leaves, and for each such tree, there are at most $(2n+2)^t$ possibilities to turn it into a DeMorgan formula ($2n$ input literals and two types of gates, AND and OR). Hence, the number of different formulas of leafsize at most t is at most $n^{O(t)}$. Since, we have 2^{2^n} different boolean functions, the lower bound $t = \Omega\left(2^n/\log n\right)$ follows. □

THEOREM 1.4 (Switching Networks). *Almost every boolean function of n variables requires switching networks with $\Omega(2^n/n)$ contacts.*

PROOF. Every set of t edges is incident with at most $2t$ nodes. Using these nodes, at most $r = (2t)^2$ their pairs (potential contacts) can be built. Hence, the number of switching networks with t edges is at most $\binom{r+t}{t} = O(t)^t$. Since there are at most $(2n)^t$ ways to turn a graph with t edges into a switching network, at most $(nt)^{O(t)}$ different boolean functions can be computed by switching networks with at most t contacts. Comparing this number with the total number of all boolean functions, yields the result. □

It is also known that all three lower bounds above are optimal up to constant factors. These results, however, do not solve the problem: we know that almost all boolean functions are complex, but no specific complex function is known. The highest known lower bounds for circuits computing explicit boolean functions in n variables have the form:

- $4n-4$ for circuits over $\{\wedge, \vee, \neg\}$ computing $x_1 \oplus x_2 \oplus \cdots \oplus x_n$, Redkin (1973);
- $5n - o(n)$ for circuits over the basis with all fanin-2 gates, except the parity and its negation, Iwama et al. (2001);

- $3n-o(n)$ for general circuits over the basis with all fanin-2 gates, Blum (1984);
- $n^{3-o(1)}$ for DeMorgan formulas, Hastad (1993);
- $\Omega(n^2/\log^2 n)$ for deterministic and $\Omega(n^{3/2}/\log n)$ for nondeterministic branching programs, Nechiporuk (1966).

We have only listed highest bounds we currently have. The bounds for circuits and formulas were obtained by gradually increasing previous lower bounds. A lower bound $2n$ for general circuits was first proved by Schnorr (1974). Then Paul (1977) proved a $2.5n$ lower bound, Stockmayer (1977) gave the same $2.5n$ lower bound for a larger family of boolean functions, and finally Blum (1984) proved the lower bound $3n-o(n)$. For circuits over the basis with all fanin-2 gates, except the parity and its negation, a lower bound of $4n$ was earlier obtained by Zwick (1991b). For formulas, the first nontrivial lower bound $\Omega(n^{3/2})$ was proved by Subbotovskaya (1961), then a lower bound $\Omega(n^2)$ was proved by Khrapchenko (1971), and a lower bound of $\Omega(n^{5/2})$ by Andreev (1985).

All the lower bounds for general circuits were proved using the so-called "gate-elimination" argument. The proofs themselves consist of a rather involved case analysis, and we will not present them here. Instead of that we will demonstrate the main idea by proving weaker lower bounds.

1.3. A $3n$ lower bound for circuits

The *gate-elimination* argument does the following. Given a circuit for the function in question, we first argue that some variable (or set of variables) must fan out to several gates. Setting this variable to a constant will eliminate several gates. By repeatedly applying this process, we conclude that the original circuit must have had many gates.

To illustrate the basic idea, we apply the gate-elimination argument to threshold functions

$$Th_k^n(x_1,\ldots,x_n)=1 \quad \text{iff} \quad x_1+x_2+\cdots+x_n \geq k\,.$$

THEOREM 1.5. *Even if all boolean functions in at most two variables are allowed as gates, the function Th_2^n requires at least $2n-4$ gates.*

PROOF. The proof is by induction on n.

For $n=2$ and $n=3$ the bound is trivial.

For the inductions step, take an optimal circuit for Th_2^n, and suppose w.l.o.g. that the bottom-most gate g acts on variables x_i and x_j (where $i \neq j$), i.e. that this gate has the form $g=\varphi(x_i,x_j)$ for some $\varphi:\{0,1\}^2\to\{0,1\}$. Notice that under the four possible settings of these two variables, the function Th_k^n has *three* different subfunctions Th_0^{n-2}, Th_1^{n-2} and Th_2^{n-2}. It follows that either x_i or x_j fans out to another gate h, for otherwise our circuit would have only *two* inequivalent sub-circuits under the settings of x_i and x_j. Why? Just because the gate $g=\varphi(x_i,x_j)$ can only take *two* values, 0 and 1.

Suppose now that it is x_j that fans out to h. Setting x_j to 0 eliminates the need of both gates g and h. The resulting circuit computes Th_2^{n-1}, and by induction, has at least $2(n-1)-4$ gates. Adding the two eliminated gates to this bound shows that the original circuit has at least $2n-4$ gates, as desired. □

Theorem 1.5 holds for circuits whose gates are any boolean functions in at most two variables. For circuits over the basis $\{\wedge,\vee,\neg\}$ one can prove a slightly higher lower

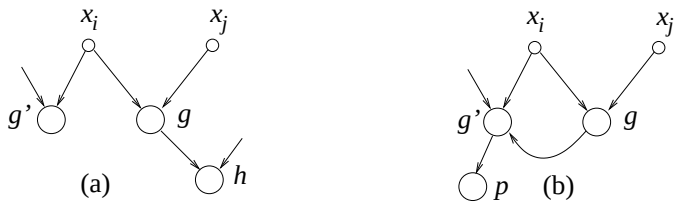


FIGURE 5. The two cases in the proof of Theorem 1.6.

bound. For this we consider the parity function

$$\oplus_n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n\,.$$

THEOREM 1.6. *The minimal number of AND and OR gates in a circuit over* $\{\wedge, \vee, \neg\}$ *computing* $\oplus_n$ *is* $3(n-1)$.

PROOF. The upper bound follows since $x \oplus y$ is equal to $(x \wedge \neg y) \vee (\neg x \wedge y)$. For the lower bound we prove the existence of some x_i whose replacement by a suitable constant eliminates 3 gates. This implies the assertion for $n = 1$ directly and for $n \geq 3$ by induction.

Let g be the first gate of an optimal circuit for $\oplus_n(x)$. Its inputs are different variables x_i and x_j (see Fig. 5). If x_i would have fanout 1, that is, if g would be the only gate which x_i is feeding in, then we could replace x_j by a constant so that gate g would be replaced by a constant. This would imply that the output became independent of the ith variable x_i in contradiction to the definition of parity. Hence, x_i must have fanout at least 2. Let g' be the other gate feeded in by x_i. We now replace x_i by such a constant that g becomes replaced by a constant. Since under this setting of x_i the parity is not replaced by a constant, the gate g cannot be an output gate. Let h be a successor of g. We only have two possibilities: either h coincides with g' or not.

Case (a): $g' \neq h$. Then we can set x_i to a constant so that g will become set to a constant. This will eliminate the need of all three gates g, g' and h.

Case (b): $g' = h$. In this case g has fanout 1. We can set x_i to a constant so that g' will become set to a constant. This will eliminate the need of all three gates g, g' and p.

In either case we eliminate at least 3 gates. □

1.4. Coin-flipping in circuits is useless

Probabilistic circuits have, besides standard inputs $x_1, \ldots, x_n$, some specially designed inputs $r_1, \ldots, r_m$ called random inputs. When these random inputs are chosen from a uniform distribution on $\{0,1\}$, the output of the circuit is a random variable. A probabilistic circuit $C(x)$ computes a boolean function $f(x)$ if

$$\Pr[C(x) = f(x)] \geq 3/4 \quad \text{for each} \quad x \in \{0,1\}^n\,.$$

There is nothing special about using the constant $3/4$ here—one can take any constant $> 1/2$ instead.

Can probabilistic circuits have much smaller size than usual (deterministic) circuits? A negative answer is given by the following

THEOREM 1.7 (Adleman 1977). *If a boolean function f in n variables can be computed by a probabilistic circuit of size ℓ, then f can be computed by a deterministic circuit of size $O(n\ell)$.*

PROOF. Let C be a probabilistic circuit that computes $f(x)$. Take k independent copies of this circuit (each with its own random inputs), and consider the probabilistic circuit C_k that computes the majority of the outputs of these k circuits. Since each of these outputs is correct with probability $p \geq 3/4$, Chernoff inequality yields that $\Pr[C_k(x) \neq f(x)] \leq e^{-\Omega(k)}$ for each $x \in \{0,1\}^n$. Therefore, for $k = \Theta(n)$, there is a setting of the random inputs, which always gives the correct answer. □

1.5. Recap: Normal forms and restrictions

A boolean function can be represented in several manners. The most commonly used one is by means of a boolean (or propositional) formula in conjunctive (CNF) or disjunctive (DNF) normal form. Let us shortly recall these concepts.

A *literal* is a boolean variable x_i or its negation $\neg x_i$; a negated variable is also written as $\overline{x_i}$ instead of $\neg x_i$. Literals are also denoted as x_i^σ where x_i^1 stands for x_i and x_i^0 stands for $\neg x_i$.

A *clause* is an OR of literals, whereas a *monomial* is an AND of literals. Clauses c and monomials m containing a contradicting pair of literals are trivial: for them we have $c \equiv 1$ and $m \equiv 0$. We will often identify clauses and monomials with the *sets* of their literals. The *length* of a clause or a monomial is the number of literals in it.

A *conjunctive normal form* or *CNF* is an AND of clauses, whereas a *disjunctive normal form* or *DNF* is an OR of monomials. A CNF is a *k-CNF* if each its clause has length at most k.

Let f be a boolean function, and $X = \{x_1, \ldots, x_n\}$ the set of its variables. A *partial assignment* (or *restriction*) is a function $\varrho : X \to \{0, 1, *\}$, where we understand $*$ to mean that the corresponding variable is unassigned. Each such partial assignment ϱ yields a *restriction* (or a *subfunction*) $f\restriction_\varrho$ of f in a natural way:

$$f\restriction_\varrho = f(\varrho(x_1), \ldots, \varrho(x_n)).$$

Note that $f\restriction_\varrho$ is a function of the variables x_i for which $\varrho(x_i) = *$. For example, if

$$f = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2) \wedge (x_1 \vee \overline{x}_3)$$

and $\varrho(x_1) = 1$, $\varrho(x_2) = \varrho(x_3) = *$ then $f\restriction_\varrho = x_2$.

A *1-term* of f is a partial assignment ϱ such that $f\restriction_\varrho \equiv 1$. That is, a 1-term of f is a consistent set of literals such that evaluating these literals to 1 forces the function to output 1 independent on the values of the (possibly) remaining free variables. *0-terms* of f are defined dually: such are all partial assignments ϱ such that $f\restriction_\varrho \equiv 0$. Minimal under set inclusion 1-terms (resp., 0-terms) are called minterms (resp., maxterms) of f.

Namely, a *minterm* of f is a restriction ϱ such that $f\restriction_\varrho \equiv 1$ and which is minimal in the sense that un-specifying every single value $\varrho(x_i) \in \{0,1\}$ already violates this property. The *length* of a minterm is the number $n - |\varrho^{-1}(*)|$ of assigned variables. Each restriction ϱ can be looked at as monomial. For example, the restriction $\varrho(x_1) = 0$, $\varrho(x_3) = 1$ can be looked at as a monomial $m = \overline{x_1} \wedge x_3$. Hence, minterms of f are monomials m such that $m(a) \leq f(a)$ for all input vectors $a \in \{0,1\}^n$, but this does not hold anymore if we remove at least one literal from m. Minterms of $\neg f$ are called *maxterms* of f.

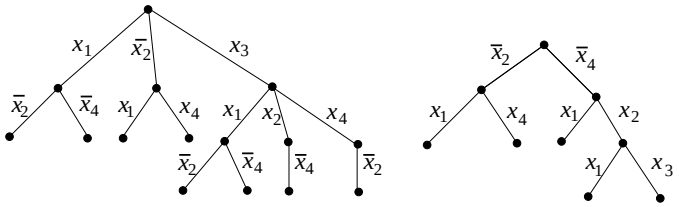


FIGURE 6. Two DNF-trees of the same CNF $f = (x_1 \vee \overline{x_2} \vee x_3)(x_1 \vee x_2 \vee x_4)(\overline{x_2} \vee \overline{x_4})$. The second tree is obtained by parsing the clauses of f in the inverse order.

If all the minterms of f have length at most k then, clearly, f has a k-DNF (take the Or of all these minterms). But the opposite is false! Namely, f *can* have a k-DNF even though some of its minterms are much longer than k, see Exercise 1.5.

In some lower bound arguments—like those for monotone circuits or constant depth circuits—we need a way to switch between k-CNFs and k-DNFs; this is done by so-called "switching lemmas." A useful way to visualize this "switching" is via transversal trees.

Let $f = c_1 \wedge \cdots \wedge c_\ell$ be a CNF. It will be convenient to identify clauses and monomials with the *sets* of their literals. The *DNF-tree* T_f of a CNF f is defined inductively as follows (see Fig. 6).

a. The first node of T_f corresponds to the first clause c_1, and the outgoing $|c_1|$ edges are labeled by the literals of c_1.
b. Suppose we have reached a node v, and let m be the monomial consisting of the labels of edges from the root to v.
   - If $m \cap c_i \neq \emptyset$ for all clauses c_i of F, then v is a leaf.
   - Otherwise, let c_i be the *first* clause such that $m \cap c_i = \emptyset$. Remove from c_i all literals whose negations belong to m (if there are any) to obtain a clause c_i'. Then the node v has $|c_i'|$ outgoing edges labeled by the literals in c_i'.

Each path from the root to a leaf of T_f corresponds to a monomial of f (since each such path intersects all its clauses). Hence, the OR over all paths gives us a DNF formula for f. Note, however, that different orderings of clauses in a given CNF may lead to DNF-trees of entirely different form (cf. Fig. 6).

Using such a tree representation we can, for example, immediately show that every k-CNF f can be represented as DNF containing at most k^i (instead of all $2^i\binom{n}{i}$ possible) monomials of length i, for each i. This holds just because a DNF-tree of every k-CNF has fanout at most k.

A boolean function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $x \leq y$ implies $f(x) \leq f(y)$, where $x \leq y$ means that $x_i \leq y_i$ for all positions i (see Fig. 7).

Note that a minterm of a monotone boolean function is a minimal set of variables which, if assigned the value 1, forces the function to take the value 1 regardless of the values assigned to the remaining variables. Similarly, a maxterm of such a function is a minimal set of variables which, if assigned the value 0, forces the function to take the value 0 regardless of the values assigned to the remaining variables.

Note also that one set S can be both minterm and maxterm of the same function! For example, if $f(x_1, x_2, x_3)$ outputs 1 iff $x_1 + x_2 + x_3 \geq 2$, then $S = \{1, 2\}$ is both a minterm and a maxterm of f, because $f(1, 1, x_3) = 1$ and $f(0, 0, x_3) = 0$.

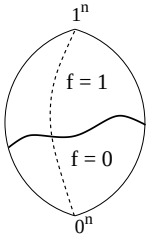


FIGURE 7. A distribution of the values of a monotone boolean function in the binary n-cube $\{0,1\}^n$. Along any path (or chain) from the all-0 vector 0^n to the all-1 vector 1^n, the function can only change its value from 0 to 1 (not from 1 to 0), and can do this at most once.

Yet another special property of monotone functions, not shared by other boolean functions, is that each such function f can be written as a monotone CNF $f = c_1 \wedge \cdots \wedge c_s$ as well as a monotone DNF $f = m_1 \vee \cdots \vee m_t$ in a *unique* way. Here $c_1, \ldots, c_2$ are all maxterms of f, and $m_1, \ldots, m_t$ are all minterms of f; all they are monotone (have no negated variables). Moreover, if we look at clauses/monomials as *sets* of their variables, then these two families have the following *cross-intersection property*: $c_i \cap m_j \neq \emptyset$ for all i, j: Would we have $c_i \cap m_j = \emptyset$ for some i and j, then we could set all variables of c_i to 0 and all remaining variables to 1 so that the resulting input vector $\boldsymbol{a} \in \{0,1\}^n$ would be forced to satisfy $f(\boldsymbol{a}) = 0$, because $c_i(\boldsymbol{a}) = 0$, as well as $f(\boldsymbol{a}) = 1$, because $m_j(\boldsymbol{a}) = 1$, a contradiction.

Finally, note that there is a 1-to-1 correspondence between monotone boolean functions $f : \{0,1\}^n \to \{0,1\}$ and anti-chains in $[n]$, that is, families $\mathscr{F}$ of subsets of $[n]$ no member of which is contained in another. Namely, given such an anti-chain $\mathscr{F}$, we can associate with each its member $S \in \mathscr{F}$ a monomial $m = \bigwedge_{i \in S} x_i$, and the OR of all these monomials gives a DNF of a monotone boolean function. The other direction follows from the uniqueness of DNFs.

1.6. Combinatorial rectangles

Important objects when analyzing boolean circuits are so-called "combinatorial rectangles." These are special subsets of $\{0,1\}^n \times \{0,1\}^n$, and are important when dealing with formula size and circuit depth. We will use this concept quite often.

An n-dimensional *combinatorial rectangle*, or just a *rectangle*, is a non-empty Cartesian product $S = S^0 \times S^1$ of two disjoint subsets S^0 and S^1 of vectors in $\{0,1\}^n$. Vector pairs $e = (x, y)$ with $x \neq y$ will be referred to as *edges*. A *subrectangle* of S is a subset $R \subseteq S$ which itself forms a rectangle. A boolean function $f : \{0,1\}^n \to \{0,1\}$ *separates* the rectangle $S = S^0 \times S^1$ if

$$f(x) = \begin{cases} 0 & \text{for } x \in S^0, \\ 1 & \text{for } x \in S^1. \end{cases}$$

If the sets S^0 and S^1 form a partition of $\{0,1\}^n$, then the rectangle $S = S^0 \times S^1$ is called a *full rectangle*. Note that every boolean function $f : \{0,1\}^n \to \{0,1\}$ defines a unique full rectangle

$$S_f := f^{-1}(0) \times f^{-1}(1),$$

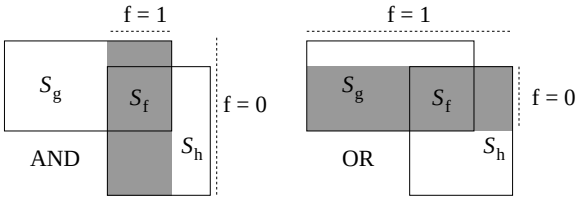


FIGURE 8. The rectangle S_f of f when $f = g \wedge h$, and when $f = g \vee h$.

which we will call the *rectangle of* f. Note also that we have much more rectangles than boolean functions.

Having rectangles $S_g := g^{-1}(0) \times g^{-1}(1)$ and $S_h := h^{-1}(0) \times h^{-1}(1)$ of two boolean functions g and h, we can compute the rectangle $S_f := f^{-1}(0) \times f^{-1}(1)$ of their AND $f = g \wedge h$ by (see Fig. 8):

$$S_f^0 = S_g^0 \cup S_h^0 \quad \text{and} \quad S_f^1 = S_g^1 \cap S_h^1, \tag{1.2}$$

as well as of their OR $f = g \vee h$ by:

$$S_f^0 = S_g^0 \cap S_h^0 \quad \text{and} \quad S_f^1 = S_g^1 \cup S_h^1. \tag{1.3}$$

Important class of rectangles are monochromatic rectangles which are the rectangles that can be separated by a single variable x_i or by a negated variable $\neg x_i$. That is, a rectangle $M = M^0 \times M^1$ is *monochromatic*, if there exists an $i \in \{1, \ldots, n\}$ such that $x_i \neq y_i$ for all edges $(x, y) \in M$; here x_i is the i-th bit in x.

The *partition number* $\mathsf{D}(S)$ of a rectangle S is the smallest number t such that S can be decomposed into t disjoint monochromatic rectangles. Note that $\mathsf{D}(S)$ is monotone under taking subrectangles: if $R \subseteq S$ is a subrectangle of S then $\mathsf{D}(R) \leq \mathsf{D}(S)$.

The following lemma reduces the (computational) problem of proving a lower bound on the formula size to a (combinatorial) problem about decomposition of rectangles.

Let $L(f)$ be the smallest leafsize of a DeMorgan formula computing f.

LEMMA 1.8 (Rychkov 1985). *For every boolean function f we have that*

$$L(f) \geq \mathsf{D}(S_f).$$

PROOF. Our goal is to show that the full rectangle S_f of f can be decomposed into at most $L(f)$ disjoint monochromatic rectangles. We argue by the induction on $L(f)$.

Base case. If $L(f) = 1$ then f is just a single variable x_i or its negation. In this case S_f itself is a monochromatic rectangle.

Induction step. Let $t = L(f)$ and assume that the theorem holds for all boolean functions g with $L(g) \leq t - 1$. Take a minimal formula for f, and assume that its last gate is an And gate (the case of an Or gate is similar). Then $f = g \wedge h$ for some boolean functions g and h such that $L(g) + L(h) = L(f)$. On the other hand, the rectangle S_f of f can be computed from the rectangles S_g and S_h for functions g and h using (1.2). By the induction hypothesis, the rectangle S_g can be covered by at most $L(g)$ disjoint monochromatic rectangles, and S_h can be also covered by at most $L(h)$ such rectangles. By restricting these coverings to S_f, we obtain a covering of S_f by at most $L(g) + L(h) = L(f)$ disjoint monochromatic rectangles, as desired. □

It is not known whether some polynomial inverse of Rychkov's lemma holds.

RESEARCH PROBLEM 1.9. *Does* $L(f) \le \mathsf{D}(S_f)^{O(1)}$*?*

What we know is only a "quasi-polynomial" inverse

$$L(f) \le \mathsf{D}(S_f)^{2\log_2 \mathsf{D}(S_f)}$$

which we will prove later in the part devoted to communication complexity (see Lemma 8.9). Still, the latter inequality implies that boolean functions f in n variables such that $\mathsf{D}(S_f) \ge 2^{(1-o(1))\sqrt{n}}$ exist. Hence, in principle, the partition number $\mathsf{D}(S)$ can also achieve super-polynomial lower bounds on the formula size.

1.7. Matrix complexity

As pointed by Sipser (1992), one of the impediments in the lower bounds area is a shortage of problems of *intermediate* difficulty which lend insight into the harder problems. Most of known problems (boolean functions) are either "easy" (parity, majority, etc.) or are "very hard" (clique problem, satisfiability of CNFs, and all other **NP**-hard problems). On the other hand, there are fields—like graph theory or matrix theory—with much richer spectrum of known objects. It makes therefore sense to look more carefully at the graph structure of boolean functions. That is, to move from a "bit level" to a more global one and look at a boolean function as a matrix or as a bipartite graph. This results into a concept of "graph complexity."

We can look at every boolean function $f(u,v)$ in $2m$ variables as an $n \times n$ $(0,1)$ matrix M_f with $n = 2^m$ whose rows and columns are indexed by vectors in $\{0,1\}^m$ and entries are the values of f: $M_f[u,v] = f(u,v)$. We call M_f the *truth matrix of* f; such a matrix is also called in the literature the "communication matrix of f."

The truth matrix M_f of $f(x,y)$ should not be mixed with the rectangle S_f of f! The first is a mapping

$$M_f : \{0,1\}^m \times \{0,1\}^m \to \{0,1\},$$

whereas the second is a subset

$$S_f = f^{-1}(0) \times f^{-1}(1) \subseteq \{0,1\}^{2m} \times \{0,1\}^{2m}.$$

Now, instead of computing a boolean function f starting from input literals, we can consider the computation of its truth matrix M_f starting from some "simplest" matrices. As these simplest matrices we take "rectangular" matrices .

A *rectangular matrix* is a $(0,1)$ matrix of rank 1. Each such matrix can be described by a Cartesian product $I \times J$ corresponding to its all-1 submatrix. Boolean operation on $(0,1)$ matrices are computed component-wise.

The relation between boolean functions and matrices is given by the following simple lemma. Here by a circuit we mean an arbitrary boolean circuit with literals—variables and their negations—as inputs.

LEMMA 1.10 (Magnification Lemma for Matrices). *In every circuit computing* $f(x,y)$ *it is possible to replace its input literals by rectangular matrices so that the resulting circuit computes the matrix* M_f*.*

PROOF. Take an arbitrary circuit $C(x,y)$ computing f. Replace each input literal x_i^σ with $\sigma \in \{0,1\}$ by the rectangular $|I| \times 2^m$ matrix, where $I = \{u \in \{0,1\}^m \mid u_i = \sigma\}$, and replace each input literal y_i^σ by the rectangular $2^m \times |J|$ matrix, where $J = \{v \in \{0,1\}^m \mid v_i = \sigma\}$. That the resulting circuit computes the matrix M_f can be shown by induction on the size c of our circuit.

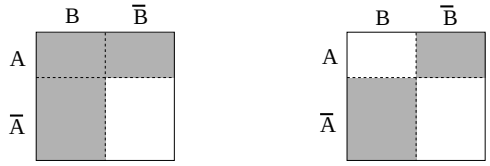


FIGURE 9. The adjacency matrix of a bipartite graph represented by an OR function $g = \bigvee_{v \in A \cup B} z_v$, and the adjacency matrix of a bipartite graph represented by a Parity function $g = \bigoplus_{v \in A \cup B} z_v$.

If $c = 0$ then f itself is a literal, and M_f in this case is one of the rectangular matrices we just defined. The induction step follows from the fact that all boolean operations (this time gates) operate on matrices component-wise. □

REMARK 1.11. Note that rectangular matrices used in the proof are very special: we only have $4m$ such matrices, and each of them either consists of a half of rows and all columns, or of a half of columns and all rows. Namely, each of them is just the truth matrix of a corresponding input literal, if we view literals as boolean functions of all $2m$ variables. Would we allow only these $4m$ rectangular matrices as inputs, then we would also have a converse.

1.8. Graph complexity

In a similar way one can consider computations of graphs when inputs are some simplest graphs, like stars or cliques.

Let $G = (V, E)$ be an n-vertex graph, and let $Z = \{z_v \mid v \in V\}$ be a set of boolean variables, one for each vertex. For two vertices $u \neq v \in V$, let $\boldsymbol{a}_{u,v} \in \{0,1\}^n$ be a vector with exactly two 1's in positions u and v.

Say that a boolean function $g(Z)$ in these variables *represents* the graph G if, for every two vertices $u \neq v \in V$, we have that $g(\boldsymbol{a}_{uv}) = 1$ iff u and v are adjacent. If the graph is bipartite then we only require that this must hold for vertices u and v from different color classes. Note that in both cases (bipartite or not), on input vectors with fewer that two 1's as well as on vectors with more than two 1's the function can take arbitrary values!

Another way to treat this concept is to look at edges as 2-element sets of vertices, and boolean functions as accepting/rejecting subsets of vertices. Then a boolean function represents a graph if it accept all edges and rejects all non-edges.

For example, a single variable z_v represents a complete star around the vertex v, that is, the graph consisting of all edges connecting v with the remaining vertices. If $A, B \subseteq V$ and $A \cap B = \emptyset$, then the boolean functions $\left(\bigvee_{u \in A} z_u\right) \wedge \left(\bigvee_{v \in B} z_v\right)$ represents a complete bipartite graph $A \times B$. In particular, every graph $G = (V, E)$ is represented by

$$\bigvee_{uv \in E} z_u z_v \quad \text{as well as by} \quad \bigoplus_{uv \in E} z_u z_v \, .$$

But these representations of n-vertex graphs are not quite compact: the number of AND gates in them may be as large as $\Theta(n^2)$. If we allow unbounded fanin OR gates then already $2n - 1$ AND gates are enough:

$$\bigvee_{u \in S} z_u \wedge \left(\bigvee_{v : uv \in E} z_v \right),$$

where $S \subseteq V$ is an arbitrary vertex cover of G, that is, a set of vertices such that every edge of G has is endpoint in S.

Now, given a model of boolean circuits, we can ask how many gates do we need to represent a given graph? It turns out that in the case of bipartite graphs, this question is related the circuit complexity of boolean functions. Namely, each boolean function $f(x,y)$ in $2m$ variables can be looked at as a bipartite graph $G_f = (V_1 \cup V_2, E)$ with color classes $V_1 = V_2 = \{0,1\}^m$, in which two vertices (vectors) x and y are adjacent iff $f(x,y) = 1$. Similarly, by fixing an encoding of vertices of a bipartite $2^m \times 2^m$ graph G by binary vectors, we obtain a boolean function f_G in $2m$ variables, a *characteristic function* of this graph defined by: $f_G(x,y) = 1$ iff x and y are adjacent in G.

Magnification Lemma!for graphs

LEMMA 1.12 (Magnification Lemma for Graphs). *In every circuit computing $f(x,y)$ it is possible to replace its input literals by ORs of new variables so that the resulting circuit represents the graph G_f.*

Instead of ORs one can take other boolean functions $g(Z)$. We only need that g computes 0 on the all-0 vector, and computes 1 on any input vector with exactly one 1. In particular, parity functions also have this property, as well as any function $g(Z) = \varphi(\sum_{w \in S} z_w)$ with $\varphi : \mathbb{N} \to \{0,1\}$, $\varphi(0) = 0$ and $\varphi(1) = 1$ does.

PROOF. Just replace input literals in the circuit computing $f(x,y)$ by ORs as follows:

$$x_i^\sigma \mapsto \bigvee_{u \in V_1, u_i = \sigma} z_u \quad \text{and} \quad y_i^\sigma \mapsto \bigvee_{v \in V_2, v_i = \sigma} z_u \,.$$

Observe that the adjacency matrices of the graphs, represented by these ORs, are precisely the rectangular matrices used in the proof of Lemma 1.10. □

This lemma is particularly appealing when dealing with circuits containing *unbounded* fanin OR (or unbounded fanin Parity gates) on the bottom, next to the inputs layer. In this case the total number of gates in the circuit computing f and in the obtained circuit representing the graph G_f is just the same! Thus, if we could prove that some explicit bipartite $n \times n$ graph with $n = 2^m$ cannot be represented by a such circuit of size n^ε, then this would immediately imply that the corresponding boolean function $f(x,y)$ in $2m$ variables cannot be computed by a (non-monotone!) circuit of size $n^\varepsilon = 2^{\varepsilon m}$, which is already exponential in the number of variables of f. This is where the term "magnification" comes from.

We will use Lemma 1.12 in Section 10.4.1 to prove truly exponential lower bounds for unbounded fanin depth-3 circuits with parity gates on the bottom layer. Now we show that, even in the class of monotone circuits with fanin-2 AND and OR gates, any lower bound larger than $12n$ for graphs would yield an *exponential* (in the number of their variables) lower bound for boolean functions in the class of non-monotone circuits with AND, OR and NOT gates.

1.9. Monotone $12n$ lower bound for graphs implies P $\neq$ NP

Recall that a DeMorgan circuit consist of fanin-2 AND and OR gates, and has all variables as well as their negations as inputs. A circuit is monotone if it has no negated inputs. For a graph G let $C_+(G)$ be the smallest number of gates in a monotone DeMorgan circuit representing G.

PROPOSITION 1.13. *For almost all bipartite $n \times n$ graphs G, we have*

$$C_+(G) = \Omega\left(\frac{n^2}{\log n}\right).$$

PROOF. Easy counting (as in the proof of Theorem 1.2) shows that there are at most $t^{O(t)}$ DeMorgan circuits with at most t gates. Since we have 2^{n^2} graphs, and different graph require different circuit, the lower bound follows. □

Thus, overwhelming majority of graphs requires almost quadratic number of gates to represent them. On the other hand, we are now going to show (Corollary 1.16 below) that any *explicit* graph G with $C_+(G) \geq 12n + \phi(n)$ would give us an explicit boolean function f in $2m$ variables which cannot be computed by a non-monotone(!) DeMorgan circuit with fewer than $\phi(2^m)$ gates. That is, linear lower bounds on the monotone complexity of graphs imply exponential lower bounds on the non-monotone complexity of boolean functions.

When constructing the circuit for the graph G, as in the Magnification Lemma, we replace $4m$ input literals in a circuit for f_G by $4m = 4\log_2 n$ disjunctions of $2n = 2^{m+1}$ (new) variables. If we compute these disjunctions separately then we need about $mn = n\log n$ fanin-2 OR gates. The disjunctions can, however, be computed much more efficiently—using only $12n$ OR gates—if we compute all these disjunctions *simultaneously*.

LEMMA 1.14. *Any collection of k disjunctions over n variables can be computed using at most $n + 2^{k+1} - k - 2$ fanin-2 OR gates.*

The proof of this lemma is a bit technical, and we postpone it to the end of this section. For us is interesting the following its consequence.

COROLLARY 1.15. *Let n be a power of two. Then any collection of $p\log_2 n$ disjunctions of variables $x_1, \ldots, x_n$ can be simultaneously computed by a circuit consisting solely of at most $3pn$ fanin-2 OR gates.*

PROOF. We want to compute $m = p\log_2 n$ disjunctions. Split these disjunctions into p groups, each containing $k = \log_2 n$ disjunctions. Applying Lemma 1.14 to each group separately, we get a circuit of size

$$p(n + 2^{k+1} - k - 2) \leq pn + 2p2^{\log_2 n} = 3pn\,. \qquad \square$$

Let now $f(x, y)$ be a boolean function in $2m$ variables, and $G_f = (V_1, V_2, E)$ the corresponding bipartite $n \times n$ graph with $V_1 = V_2 = \{0,1\}^m$. Let $C(f)$ be the smallest size of a DeMorgan circuit computing f, and $C_+(G)$ the smallest size of a monotone DeMorgan circuit representing the graph G.

COROLLARY 1.16. $C(f) \geq C_+(G_f) - 12n$.

PROOF. By Magnification Lemma, all $2m = 2\log_2 n$ x-literals are replaced by a disjunctions on the set $\{z_u \mid u \in V_1\}$ of n variables. By Corollary 1.15 (with $p = 2$), all these disjunctions can be computed using at most $6n$ fanin-2 OR gates. Since the same holds also for y-literals, we are done. □

Hence, proving even linear lower bounds $C_+(G) \geq cn$ for graphs is a very difficult task. Still, Exercise 1.4 shows that at least for $c = 2$ this can be easily done.

PROOF OF LEMMA 1.14. Given a collection of k subsets $S_1, \ldots, S_k$ of $[n] = \{1, \ldots, n\}$, our goal is to compute k disjunctions

$$\bigvee_{i \in S_1} x_i \,, \ \bigvee_{i \in S_2} x_i, \ \ldots \ \bigvee_{i \in S_k} x_i \,, \tag{1.4}$$

by a circuit only containing fanin-2 OR gates. For each 0-1 string w of length[2] $|w| \leq k$ define an auxiliary set J_w by

$$J_w = \{j \mid \ j \in S_i \text{ iff } w(i) = 1\} \,.$$

That is, we look at the first $|w|$ sets $S_1, \ldots, S_{|w|}$ and include an element j in J_w iff w is the indicator vector for the occurrence of j in these sets. The sets J_w have the following properties:

$$J_w \cap J_{w'} = \emptyset \qquad \text{for } w \neq w' \text{ and } |w| = |w'|; \tag{1.5}$$

$$J_w = J_{w0} \cup J_{w1} \qquad \text{for } |w| < k; \tag{1.6}$$

$$S_i = \bigcup_{w:|w|=i-1} J_{w1} \qquad \text{for } i = 1, \ldots, k. \tag{1.7}$$

The first property (1.5) follows from the observation that $j \in J_w$ for some w of length $|w| = \ell$ iff w is the indicator vector for the occurrence of j in the first ℓ sets $S_1, \ldots, S_\ell$, and no element x can have two such vectors.

The second property (1.6) follows from the observation that an indicator vector w for an element j of length $|w| = i < k$ can be extended to two vectors $w0$ and $w1$ of length $i+1$, and at least one of them must be an indicator vector for j of length $i+1$, depending on whether j belongs to the $(i+1)$-th set S_{i+1} or not.

To show the third property (1.7), observe that an element j can only then belong to J_{w1} if it belongs to $S_{|w|+1} = S_i$. On the other hand, if $j \in S_i$ then $j \in J_{w1}$, where w is the indicator vector for j of length $i-1$.

We can now compute our k disjunctions (1.4) as follows. First compute all 2^k disjunctions $\bigvee_{i \in J_w} x_i$ with $|w| = k$. Since, by (1.5), the corresponding sets J_w in this case are disjoint, this can be done using at most n ORs. Next we use (1.6) to compute nonempty disjunctions $\bigvee_{i \in J_w} x_i$ for strings w of length $|w| < k$ using the (already computed) disjunctions $\bigvee_{i \in J_{w0}} x_i$ and $\bigvee_{i \in J_{w1}} x_i$ for longer strings:

$$\bigvee_{i \in J_w} x_i = \bigvee_{i \in J_{w0}} x_i \vee \bigvee_{i \in J_{w1}} x_i \,.$$

This can be done using $2^k - 1$ additional ORs. Finally, we use (1.7) to compute our original disjunction (1.4) by the formula

$$\bigvee_{j \in S_i} x_j = \bigvee_{w:|w|=i-1} \ \bigvee_{j \in J_{w1}} x_j \,.$$

This only requires $|\{w : |w| = i-1\}| - 1 = 2^{i-1} - 1$ new ORs, and thus, for $i = 1, \ldots, k$ all together

$$\sum_{i=1}^{k} (2^{i-1} - 1) = 2^k - 1 - k$$

[2] In this proof, $|w|$ denotes the total number of bits in w, not the number of 1's.

ORs. The number of ORs used in the entire circuit computing all disjunctions (1.4) does not exceed

$$n + (2^k - 1) + (2^k - 1 - k) = n + 2^{k+1} - k - 2\,. \qquad \square$$

Exercises

Ex. 1.1 (Minimal circuits are very unstable). Let F be a circuit over some basis computing a boolean functions f, and assume that F is *minimal*, that is, no circuit with a smaller number of gates can compute f. In particular, minimal circuits are "unstable" with respect to deletion of its gates: the resulting circuit must make an error. Prove that, in fact, minimal circuits are unstable in a much stronger sense: we cannot even *replace* a gate by another one; the size of the resulting circuit remains the same but, nevertheless, the function computed by a new circuit differs from that computed by the original one.

More precisely, write $g \le h$ for boolean functions in n variables, if $g(v) \le h(v)$ for all $v \in \{0,1\}^n$. Call a boolean function h a *neighbor* of a boolean function g if either (i) $g \oplus \delta \le h \oplus \delta \oplus 1$ for some $\delta \in \{0,1\}$, or $g \oplus x_i \le g \oplus h$ for some $i \in \{1,\ldots,n\}$.

(a) Show that constants 0 and 1 are neighbors of all non-constant functions.

(b) Show that neighbors of the OR gate $\vee$ are all the two variable boolean functions, except $\oplus$ and the function $\vee$ itself.

(c) Let F be a minimal circuit, e a gate in it of fanin m, and h be a boolean function in m variables. Let $F_{e\to h}$ be the circuit obtained from F as follows: replace the boolean function attached to the gate e by h and remove all the gates that become redundant in the resulting circuit. Prove that $F_{e\to h} \neq F$.

Hint: Case (i) can be proved as follows. Since F is optimal, we cannot replace the gate e by the constant δ, i.e. there must be at least one vector $v \in \{0,1\}^n$ such that $F_{e\to\delta}(v) \neq F(v)$. This, in particular, means that $g(f_1(v),\ldots,f_m(v)) = \delta \oplus 1$, where g is a boolean function attached to the gate e, and $f_1,\ldots,f_m$ are boolean function computed at its inputs. Since $g \oplus \delta \le h \oplus \delta \oplus 1$, we have that $h(f_1(v),\ldots,f_m(v)) = \delta$, and hence, $F_{e\to h}(v) = F_{e\to\delta}(v) \neq F(v)$.

Ex. 1.2 (Circuits as linear programs). Show that for every circuit $C(x)$ over $\{\wedge,\vee,\neg\}$ there is a system $L(x,y)$ of linear constraints (linear inequalities with coefficients ± 1) such that:

a. For every $x \in \{0,1\}^n$, $C(x) = 1$ iff there is an y such that all constraints in $L(x,y)$ are satisfied.

b. The number of constrains in $L(x,y)$ is by only a constant fraction larger than the number of gates in C.

c. The number of y-variables is at most the number of gates in C.

Hint: Introduce a variable for each gate. For an $\wedge$-gate $g = u \wedge v$ use the constraints $0 \le g \le u \le 1$, $0 \le g \le v \le 1$, $g \ge u + v - 1$. What constraints to take for $\neg$-gates and for $\vee$-gates? For the output gate g add the constraint $g = 1$. Show that, if the x-variables have values 0 and 1, then all other variables are forced to have value 0 or 1 equal to the output value of the corresponding gate.

Ex. 1.3. Let $G = ([n], E)$ be an n-vertex graph, and d_i be the degree of vertex i in G. Then G can be represented by a monotone formula

$$F(X) = \bigvee_{i\in[n]} x_i \wedge \left(\bigvee_{j:\{i,j\}\in E} x_j \right).$$

A special property of this formula is that the ith variable occurs at most $d_i + 1$ times.

Prove that, if G has no complete stars, then *any* minimal formula representing G must have this property.

Hint: Take a minimal formula F for G, and suppose that some variable x_i occurs $m_i > d+1$ times in it. Consider the formula

$$F' = F_{x_i=0} \vee F_i \text{ with } F_i = x_i \wedge \left(\bigvee_{j:ij\in E} x_j \right),$$

where $F_{x_i=0}$ is the formula obtained from F by setting to 0 all m_i occurrences of the variable x_i. Show that F' represents G, and compute its leafsize to get a contradiction with the minimality of F.

Ex. 1.4. Let $G_n = K_{n-1} + E_1$ be a complete graph on $n-1$ vertices $1, 2 \ldots, n-1$ plus one isolated vertex n. Let $F(x_1, \ldots, x_n)$ be an arbitrary monotone circuit with fanin-2 AND and OR gates representing G_n.

a. Show: If $n \geq 3$ then every input gate x_i for $i = 1, \ldots, n-1$ has fanout at least 2.
b. Use the previous claim to derive that G_n cannot be represented by a monotone circuit using fewer than $2n-6$ gates.

Ex. 1.5. The *storage access function* $f(x,y)$ is a boolean function in $n+k$ variables $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_k)$ where $n = 2^k$, and is defined as follows: $f(x,y) := x_{\text{bin}(y)}$, where $\text{bin}(y) = \sum_{i=1}^{k} y_i 2^{i-1}$ is the integer whose binary representation is vector y.

Show that the monomial $K = x_1 x_2 \cdots x_n$ is a minterm of f, but still f can be written as a $(k+1)$-DNF.

Hint: For the second claim, observe that the value of $f(x,y)$ depends only on $k+1$ bits $y_1, \ldots, y_k$ and $x_{\text{bin}(y)}$.

Bibliographic Notes

Theorem 1.6 was proved by Schnorr (1974). The concept of graph complexity was first introduced by Pudlák, Rödl and Savický (1988), and Razborov (1990). Lemma 1.14 is from Pudlák et al. (1988).

Part 1

Formulas and Monotone Circuits

CHAPTER 2

Boolean Formulas: The Classics

If not stated otherwise, by a formula we will understand a DeMorgan formula, that is, a formula with fanin-2 AND and OR gates whose inputs are variables and their negations. By $L(f)$ we denote the minimal leafsize and by $\mathrm{Depth}(f)$ the minimal depth of a DeMorgan formula computing a given boolean function f.

2.1. Size versus depth

Since the underlying graph of a DeMorgan formula is a binary tree, any formula of depth d can have at most 2^d leaves. This implies that, for every boolean function f,

$$\mathrm{Depth}(f) \geq \log_2 L(f).$$

In fact, we also have a converse inequality:

THEOREM 2.1 (Spira 1971). *For any f,* $\mathrm{Depth}(f) \leq 1 + 3.5 \log_2 L(f)$.

PROOF. We will prove a slightly more general claim: any DeMorgan formula of leafsize m can be transformed into an equivalent formula of depth $1 + 2 \cdot \log_{3/2} m$; two formulas are *equivalent* if they compute the same boolean function.

We argue by induction on m. The basis case $m = 1$ it trivial. So, assume that the claim holds for all formulas of leafsize at most $m-1$, and take an arbitrary formula F of leafsize m. Recall that the underlying graph of this formula is a binary tree with m leaves.

CLAIM 2.2. In every binary tree with m leaves, there is a subtree with at least $m/3$ and at most $2m/3$ leaves.

PROOF. Define the weight of a node in a tree as the number of leaves of the subtree rooted in this node. Start from the root, and each time see whether some of two successors has weight at most $2m/3$. If not, then take any one of them and continue the walk. Since the weight of a node is at most the sum of the weights of its two successors, we will eventually find a desired subtree. □

By this claim, there must be a subformula G of F whose leafsize lies between $m/3$ and $2m/3$. Let F_0 (resp., F_1) be F with the distinguished subformula G replaced by constant 0 (resp., 1). It is not difficult to verify (do this!) that F is equivalent to

$$(F_0 \wedge \neg G) \vee (F_1 \wedge G).$$

By the choice of G, the formulas G and $\neg G$ have at most $2m/3$ leaves, and formulas F_0 and F_1 also have at most $m - m/3 = 2m/3$ leaves. By the induction hypothesis, $F_0, F_1, G, \neg G$ are equivalent to formulas $F_0', F_1', G', (\neg G)'$ all of depth 1 plus two times the logarithm base $3/2$ of their respective leafsizes. Hence, if $d(F)$ denotes the depth of a formula F, then the formula

$$(F_0' \wedge (\neg G)') \vee (F_1' \wedge G')$$

is equivalent to F and has depth

$$\begin{aligned} d(F) &\leq 2 + \max\{d(F_0'), d(F_1'), d(G'), d(\neg G)'\} \\ &\leq 2 + 1 + 2 \cdot \log_{3/2}\left(\frac{2m}{3}\right) \\ &= 1 + 2 \cdot \log_{3/2}\left(\frac{2m}{3} \cdot \frac{3}{2}\right) \\ &= 1 + 2 \cdot \log_{3/2} m \leq 1 + 3.5 \log_2 m\,. \end{aligned}$$

□

A DeMorgan formula is *monotone* if it has no negated variables as inputs. Let $L_+(f)$ and $\mathrm{Depth}(f)$ denote, respectively, the minimal leafsize and the minimal depth of a monotone DeMorgan formula computing a monotone boolean function f.

THEOREM 2.3. *For any monotone f, $\mathrm{Depth}_+(f) \leq 1 + 3.5 \log_2 L_+(f)$.*

PROOF. The proof is almost the same. Just take a formula $F_0 \vee (F_1 \wedge G)$ instead of $(F_0 \wedge \neg G) \vee (F_1 \wedge G)$ and use the monotonicity of $F(x)$. □

2.2. The effect of random restrictions

Already in 1961, Subbotovskaya has found an argument to show that some boolean functions require DeMorgan formulas of super-linear size. Her idea was, given a formula F computing some function f, to set randomly some of the variables to constants and show that this restriction reduces the size of F considerably whereas the resulting subfunction of f is not much easier.

Let us recall some notation. Let f be a boolean function, and $X = \{x_1, \ldots, x_n\}$ the set of its variables. A *partial assignment* (or *restriction*) is a function $\varrho : X \to \{0, 1, *\}$, where we understand $*$ to mean that the corresponding variable is unassigned. The function from f by applying the partial assignment ϱ is denoted by $f\restriction_\varrho$.

Let $\mathscr{R}_k$ be the set of all partial assignments which leave exactly k variables unassigned. What we will be interested in is the *random* restrictions $f\restriction_\varrho$ that results from choosing a *random* partial assignment from $\mathscr{R}_k$.

The probability distribution of restrictions in $\mathscr{R}_k$ we will use is the following: randomly assign k variables to be $*$, and assign all other variables to be 0 or 1 randomly and independently.

The following lemma shows that a random restriction may substantially reduce the size of a formula.

LEMMA 2.4 (Subbotovskaya 1961). *Let f be a boolean function of n variables, and let ϱ be a random restriction from $\mathscr{R}_k$. Then, with probability at least* 3/4*,*

$$L(f\restriction_\varrho) < 4 \cdot \left(\frac{k}{n}\right)^{3/2} \cdot L(f)\,.$$

PROOF. Let F be an optimal DeMorgan formula for the function f of size $s = L(f)$. Construct the restriction ϱ in $n - k$ stages as follows: At any stage, choose a variable randomly from the remaining ones, and assign it 0 or 1 randomly. We analyse the effect of this restriction to the formula F, stage-by-stage.

Suppose the first stage chooses the variable x_i. When this variable is set to a constant, then all the input gates $e \in F$, labeled by the literals x_i and $\overline{x}_i$ will disappear from the formula F. By averaging, the expected number of such literals is s/n.

In fact, the formula is likely to be reduced even further. For each of the input gates e, labeled by x_i or $\overline{x}_i$, consider the gate which e feeds into. For example, suppose the

gate is $e \wedge G$ for some subformula G. We may assume w.l.o.g. that G does not contain the literals x_i or $\overline{x}_i$ (show this!).

Now, if the variable x_i is assigned 0, then the subformula G will disappear from the formula F, thereby erasing at least one more input gate. Since x_i is assigned 0 or 1 randomly (with probability 1/2), we expect at least $\frac{1}{2} \cdot \frac{s}{n}$ input gates to disappear because of these secondary effects. In total, we thus expect at least

$$\frac{s}{n} + \frac{s}{2n} = \frac{3s}{2n}$$

input gates $e \in F$ to disappear in the first stage, yielding a new formula with expected size at most

$$s - \frac{3s}{2n} = s \cdot \left(1 - \frac{3}{2n}\right) \leq s \cdot \left(1 - \frac{1}{n}\right)^{3/2}.$$

The succeeding stages of the restriction can be analyzed in the same way. After each stage the number of variables decrements by one. Hence, after $n - k$ stages, the expected size $\mathrm{E}\left[L(f\restriction_\varrho)\right]$ of the final formula is at most

$$s \cdot \left(1 - \frac{1}{n}\right)^{3/2} \cdot \left(1 - \frac{1}{n-1}\right)^{3/2} \cdot \ldots \cdot \left(1 - \frac{1}{k+1}\right)^{3/2} = s \cdot \left(\frac{k}{n}\right)^{3/2}.$$

By Markov's inequality, the probability that the random variable $L(f\restriction_\varrho)$ is more than 4 times its expected value is less than $\frac{1}{4}$, which completes the proof. □

COROLLARY 2.5. *Let f be a boolean function of n variables, and $1 \leq k \leq n$ be an integer. Then there exists a restriction $\varrho \in \mathscr{R}_k$ such that*

$$L(f\restriction_\varrho) \leq 4 \cdot \left(\frac{k}{n}\right)^{3/2} \cdot L(f).$$

EXAMPLE 2.6. Let $f = x_1 \oplus x_2 \oplus \cdots \oplus x_n$. Applying Corollary 2.5 with $k = 1$ we have

$$1 \leq L(f\restriction_\varrho) \leq 4 \cdot \left(\frac{1}{n}\right)^{3/2} \cdot L(f),$$

which gives the lower bound $L(f) = \Omega(n^{3/2})$.

2.3. An $n^{2.5}$ lower bound

Andreev (1987) used Subbotovskaya's argument to prove the first super-quadratic lower bound for formula size.

Let X be a set of n boolean variables, where n is a power of 2. Take $b := \log_2 n$ and $m = n/b$, and arrange the variables in X into a $b \times m$ matrix

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ & & \cdots & \\ x_{b1} & x_{b2} & \cdots & x_{bm} \end{bmatrix}.$$

Given a boolean function $\varphi : \{0,1\}^b \to \{0,1\}$ on b variables, let $f_\varphi(X)$ denote the following boolean function on $n = bm$ variables X

$$f_\varphi(X) = \varphi\left(\bigoplus_{j=1}^{m} x_{1j}, \bigoplus_{j=1}^{m} x_{2j}, \ldots, \bigoplus_{i=j}^{m} x_{bj}\right).$$

That is, we compute the parity of bits in each row of X and apply φ to these parities.

LEMMA 2.7. *Let ϱ be a random restriction from $\mathscr{R}_k$ where $k = \lceil b \ln(4b) \rceil$. Then with probability at least $3/4$, the restriction ϱ assigns at least one $*$ to each of the b rows of X.*

PROOF. Observe that the restriction ϱ assigns a $*$ to each single variable with probability $\binom{n-1}{k-1}/\binom{n}{k} = \frac{k}{n}$. By the union bound, the probability that some of b rows will get no $*$ is at most

$$b \cdot \left(1 - \frac{k}{n}\right)^m \leq b \cdot e^{-\frac{km}{n}} \leq b \cdot e^{-\ln(4b)} = 1/4 = 3/4\,. \qquad \square$$

Let now A_n be a boolean function in $2n$ variables defined as follows. The first $n = 2^b$ variables, $b = \log_2 n$, specify the truth table of a boolean function φ of b variables. Then, the value of A_n is defined to be the value of $f_\varphi(X)$, where X is the set of remaining $n = bm$ variables.

THEOREM 2.8. $L(A_n) = \Omega\left(n^{5/2-o(1)}\right)$.

PROOF. Let $f = A_n$ and let φ be an arbitrary boolean function in b variables. By Lemma 2.7, we have that with probability at least $3/4$, the function φ is a subfunction of $f_\varphi\restriction_\varrho$, and hence,

$$\Pr[L(f_\varphi\restriction_\varrho) \geq L(\varphi)] \geq \frac{3}{4}\,.$$

On the other hand, by Lemma 2.4,

$$\Pr[L(f_\varphi\restriction_\varrho) \leq 4 \cdot \left(\frac{k}{bm}\right)^{3/2} \cdot L(f_\varphi)]\,.$$

Thus, there must be a restriction $\varrho \in \mathscr{R}_k$ for which both these events happen, implying that

$$L(f_\varphi) \geq \frac{1}{4} \cdot \left(\frac{bm}{k}\right)^{3/2} \cdot L(f_\varphi\restriction_\varrho) \geq \frac{1}{4} \cdot \left(\frac{bm}{k}\right)^{3/2} \cdot L(\varphi). \tag{2.1}$$

We already known (Theorem 1.3) that, for almost all boolean functions φ in b variables,

$$L(\varphi) = \Omega\left(\frac{2^b}{\log b}\right).$$

Taking any of these "most complicated" functions φ we get from (2.1) that $L(f_\varphi)$ is at least about

$$\left(\frac{bm}{k}\right)^{3/2} \cdot \frac{2^b}{\log b} = \Omega\left(\frac{n^{5/2}}{\log n \cdot \log\log n}\right). \tag{2.2}$$

That is, the function A_n has a subfunction whose leafsize is at least this number and, in particular, is at least $\Omega\left(n^{5/2-o(1)}\right)$. $\square$

2.4. Nechiporuk's theorem

The arguments above only work for DeMorgan formulas, that is, formulas over the basis $\{\wedge, \vee, \neg\}$. Nechiporuk (1966) has found another argument which works for *binary* formulas (or formulas over *universal basis*) where all $2^4 = 16$ boolean functions in two variables as gates are allowed as gates. Actually, his argument works for circuits using any c-variable boolean functions as gates, as long as c is an absolute constant, independent of the number n of variables of the boolean function we want to compute.

Nechiporuk's idea is a refinement of that used by Shannon: if a function f has many *different* subfunctions, then any formula computing f must also have many different sub-formulas, implying that the original formula for f must be large. To realize this idea, we have only relate the number of different subfunctions with the formula size.

A *subfunction* of a boolean function $f(X)$ on $Y \subseteq X$ is a function obtained from f by setting all the variables of $X - Y$ to constants. A subfunction of a formula F is the subfunction of the boolean function it computes. Note that the number of different subfunctions is at most $2^{|Y|}$ and at most $2^{|X-Y|}$. Intuitively, if f has many subfunctions, then it is complicated and hence should require large formulas. This intuition was made precise by Nechiporuk (1966). We will derive his theorem from the following more general result.

For a boolean function (or a formula) F, let $S_Y(F)$ be the collection of functions g on a variable set Y for which either g or $\neg g$ is a subfunction of F. That is, we include a function g is $S_Y(F)$ is *at least one* of g and $\neg g$ is a subfunction of F. Let $\text{size}_Y(F)$ be the number of occurrences of variables of Y in a boolean formula F. A *binary* formula is a formula where all boolean functions in at most 2 variables are allowed as gates.

LEMMA 2.9. *For every binary formula F and every variable set Y, we have*

$$2|S_Y(F)| + 1 \leq 5^{\text{size}_Y(F)}. \tag{2.3}$$

PROOF. The lemma is proved by the induction on the leafsize of F. The base case $F = x_i$ of $F = \neg x_i$ divides into two sub-cases ($i \in Y$ or $i \notin Y$). Both sub-cases satisfy the claim, because in both of them we have that $|S_Y(F)| \leq 2$.

Assume now by induction that $F = F_1 * F_2$, where F_1 and F_2 satisfy the claim, and $*$ is a binary operation. For brevity, let $S_1 = S_Y(F_1)$ and $S_2 = S_Y(F_2)$. Consider the following two collections of boolean functions on Y:

$$T = \{g_1 * g_2 \,:\, g_1 \in S_1 \text{ and } g_2 \in S_2\} \quad \text{and} \quad T' = \{\neg g \,:\, g \in T\}.$$

Since S_i $(i = 1, 2)$ contains every subfunction of F together with its negation, we have that

$$S_Y(F) \subseteq T \cup\ T' \cup S_1 \cup S_2.$$

Since $\text{size}_Y(F_1) + \text{size}_Y(F_2) = \text{size}_Y(F)$, we obtain

$$\begin{aligned} 2|S_Y(f)| + 1 &\leq 2 \cdot (|T| + |T'| + |S_1| + |S_2|) + 1 \\ &\leq 4|S_1||S_2| + 2|S_1| + 2|S_2| + 1 \\ &= (2|S_1| + 1) \cdot (2|S_2| + 1) \\ &\leq 5^{\text{size}_Y(F_1)} \cdot 5^{\text{size}_Y(F_2)} = 5^{\text{size}_Y(F)}. \end{aligned}$$

□

We can now give a general lower bound for formula size.

Let f be a boolean function in n variables, and let $L_U(f)$ denote the smallest leafsize of a binary formula computing f. Fix a partition of the variable set $[n]$ into m disjoint subsets $Y_1, \ldots, Y_m$. For every $i \in [m]$ let $c_i(f)$ be the number of distinct subfunctions of f on the variables Y_i obtained by fixing the remaining variables to constants in all possible ways.

THEOREM 2.10 (Nechiporuk 1966).

$$L_U(f) \geq \sum_{i=1}^{m} \log_5 \left(2c_i(f) + 1\right).$$

PROOF. Take an arbitrary binary formula F for f. Since the Y_i's are disjoint, the leafsize of F is equal to $\sum_{i=1}^{m} \text{size}_{Y_i}(F)$. Since clearly $|S_{Y_i}(F)| \geq c_i(f)$, the desired lower bound on this sum follows directly from (2.3). □

A standard example of a function with many subfunctions is the *element distinctness function*. This function takes a string $s_1, \ldots, s_m$ of m elements of the set $[m^2] = \{1, \ldots, m^2\}$ and outputs 1 iff all the s_i are distinct. If we encode the elements of $[m^2]$ by binary strings of length $2 \log m$, then we obtain a boolean version of this function in $n = 2m \log m$ variables. Consider the input vector $x \in \{0,1\}^n$ to represent m strings $s_1, \ldots, s_m$ each of length $2 \log m$ where $n = 2m \log m$. Define the function ED_n so that it is 1 iff all the s_i are distinct.

THEOREM 2.11 (Element Distinctness Function).

$$L_U(ED_n) = \Omega\Big(\frac{n^2}{\log n}\Big).$$

PROOF. Let $f = ED_n$ and take a partition $Y_1, \ldots, Y_m$ of the variables of f according to the blocks $s_1, \ldots, s_m$. We claim that for each of these m blocks we have

$$N \geq \binom{m^2}{m-1} \geq \left(\frac{m^2}{m-1}\right)^{m-1} = 2^{\Theta(m \log m)}$$

different subfunctions of our function f. Indeed, $\binom{m^2}{m-1}$ is the number of ways to chose a string $a = (a_2, \ldots, a_m)$ with all the a_i distinct. If $b = (b_2, \ldots, b_m)$ is another such string, then there must be an a_i such that $a_i \notin \{b_2, \ldots, b_m\}$. But for such an a_i, the subfunction defined by a outputs 0 on input a_i, whereas that defined by b outputs 1 on the same input a_i. Hence, all the subfunctions are distinct.

Since $\log_5 N = \Omega(m \log_2 m) = \Omega(n)$ and $m = \Omega(n/\log n)$, Nechiporuk's theorem yields the desired lower bound on the leafsize. □

2.5. Khrapchenko's theorem

For DeMorgan formulas, we have yet another lower bounds argument, due to Khrapchenko (1971). He used this argument to prove a lower bound n^2 for the parity function $f(x_1, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$. Later, Rychkov (1984) observed that the essence of Khrapchenko's argument is more general: it reduces the lower bounds problem for DeMorgan formulas to a combinatorial problem about the covering of the rectangle

$$S_f = f^{-1}(0) \times f^{-1}(1)$$

by pairwise disjoint monochromatic rectangles (see Lemma 1.8):

$$L(f) \geq \mathsf{D}(S_f),$$

where $\mathsf{D}(S)$ is the smallest number t such that S can be decomposed into t disjoint monochromatic rectangles. We can use Rychkov's lemma to derive the well-known lower bound due to Khrapchenko (1971).

The *Hamming distance* $\text{dist}(x, y)$ between two vectors x and y is the number of positions in which these two vectors differ. Intuitively, if S_f contains many edges (x, y) of distance 1, then every formula separating these edges must be large, since the formula must distinguish many pairs of "very similar" inputs (they differ in just one bit). The following theorem of Khrapchenko makes this intuition precise.

THEOREM 2.12 (Khrapchenko 1971). *For every boolean function f we have that*

$$L(f) \geq \mathsf{D}(S_f) \geq \frac{|Y|^2}{|S_f|} \quad \textit{where} \quad Y = \{(x,y) \in S_f \mid \mathrm{dist}(x,y) = 1\}\,.$$

PROOF. The main property of the set Y is accumulated in the following

CLAIM 2.13. If $M = M^0 \times M^1$ is a monochromatic subrectangle of S_f, then

$$|M \cap Y|^2 \leq |M^0| \cdot |M^1|\,.$$

PROOF. Since the rectangle $M = M^0 \times M^1$ is monochromatic, each element of M^0 differs from each element in M^1 in one particular position j, whereas (x,y) is in Y only if x and y differ in exactly one position. Hence, for any given $x \in M^0$, the only possible $y \in M^1$ for which $(x,y) \in Y$ is one which differs from x exactly in position j. As a result, we have $|M \cap Y| \leq |M^0|$ and $|M \cap Y| \leq |M^1|$, and the desired upper bound on $|M \cap Y|^2$ follows. □

Consider now a partition $M_1, \ldots, M_d$ of S_f into $d = \mathsf{D}(S_f)$ disjoint monochromatic rectangles, as in Rychkov's lemma. Since the rectangles are disjoint and cover the whole rectangle S_f, we have that $|Y| = \sum_{i=1}^{d} |M_i \cap Y|$ and hence,

$$\begin{aligned} |Y|^2 = \left(\sum_{i=1}^{d} |M_i \cap Y|\right)^2 &\leq d \sum_{i=1}^{d} |M_i \cap Y|^2 \\ &\leq d \cdot \sum_{i=1}^{d} |M_i^0| \cdot |M_i^1| = d \cdot \sum_{i=1}^{d} |M_i| = d \cdot |S_f|\,, \end{aligned}$$

where the first inequality follows from the Cauchy–Schwarz inequality

$$(\sum_{i=1}^{d} a_i b_i)^2 \leq (\sum_{i=1}^{d} a_i^2) \cdot (\sum_{i=1}^{d} b_i^2)\,. \qquad \square$$

Khrapchenko's theorem can be used to show that some explicit boolean functions require formulas of quadratic size. Consider, for example, the parity function $f = x_1 \oplus \cdots \oplus x_n$, where n is a power of 2. Then $|S_f| = 2^{n-1} \cdot 2^{n-1}$, whereas $|Y| = n2^{n-1}$. Hence,

$$L(x_1 \oplus \cdots \oplus x_n) \geq \frac{n^2 2^{2(n-1)}}{2^{2(n-1)}} = n^2\,.$$

2.6. Complexity is not convex

Khrapchenko's measure is of the form

$$\mu(R) := |R| \cdot \varphi\left(\frac{|Y \cap R|}{|R|}\right) \tag{2.4}$$

where $Y = \{(x,y) \in R \mid \mathrm{dist}(x,y) = 1\}$ and $\varphi(x) = x^2$. Exercise 2.2 shows that this measure cannot yield larger than $\Omega(n^2)$ lower bounds. All subsequent attempts to modify his measure with the goal to brake the "n^2 barrier" failed (so far). So, what is bad with this measure? Perhaps larger lower bounds can be obtained by taking other subsets Y of special entries and/or using some other functions $\varphi(x)$ instead of x^2?

The answer is somewhat disappointing. Namely, it turns out that the reason for the failure of Khrapchenko-type measures is much deeper than expected: for any choice of $Y \subseteq S$ and for every convex function $\varphi(x)$, the resulting measure is *convex*, and

convex measures cannot yield super-quadratic lower bounds. To show this, we first define what is meant under a "convex" rectangle measure.

Let R be a rectangle, and $M_1, \ldots, M_t$ be all monochromatic subrectangles of R. A *fractional partition* of R is just a sequence $r_1, \ldots, r_t \in [0,1]$ of real numbers such that

$$\sum_{i:e\in M_i} r_i = 1 \quad \text{for all} \quad e \in R\,.$$

If χ_S is the characteristic function of a rectange S, that is, $\chi_S(e) = 1$ iff $e \in S$, then this condition can be written as

$$\chi_R = \sum_{i=1}^{t} r_i \cdot \chi_{M_i}\,.$$

We will shorten this last condition as

$$R = \sum_{i=1}^{t} r_i \cdot M_i\,.$$

Note that every fractional partition with all r_i in $\{0,1\}$ is just a partition of R in a usual sense.

DEFINITION 2.14 (Convex measures). A *rectangle function* is a mapping μ that assigns to each rectangle R a real number $\mu(R)$. Such a function is *convex* if, for every sequence $r_1, \ldots, r_t$ of real numbers in $[0,1]$,

$$R = \sum_{i=1}^{t} r_i \cdot M_i \quad \text{implies} \quad \mu(R) \le \sum_{i=1}^{t} r_i \cdot \mu(M_i)\,. \tag{2.5}$$

A rectangle function μ is a *rectangle measure* if it is *normalized*, that is, if $\mu(M) \le 1$ for any monochromatic rectangle M.

Hence, if $\mu(R)$ is a rectangle *measure*, then its convexity just means that

$$\mu(R) \le \pi(R)\,,$$

where $\pi(R)$ is the *fractional partition number* of a rectangle R defined by:

$$\pi(R) = \min \sum_{i=1}^{t} r_i\,,$$

where the minimum is over all fractional partitions $r_1, \ldots, r_t$ of R.

The following is an analogon of Khrapchenko's theorem for fractional partition number.

THEOREM 2.15. *For every rectangle R we have that*

$$\pi(R) \ge \frac{|Y|^2}{|R|}, \quad \textit{where} \quad Y = \{(x,y) \in R \mid \mathrm{dist}(x,y) = 1\}\,.$$

PROOF. Applying the duality for linear programs, one can write the fractional partition number as

$$\pi(R) = \max_{w} \sum_{e\in S} w(e)\,,$$

where the maximum is over all functions $w : R \to \mathbb{R}$ satisfying the constraint

$$\sum_{e\in M} w(e) \le 1$$

for all monochromatic rectangles $M \subseteq R$. Hence, in order to prove a lower bound $\pi(R) \geq t$ it is enough to find at least one weight function $w : R \to \mathbb{R}$ such that $\sum_{e \in S} w(e) \geq t$, and the weight of each monochromatic rectangle does not exceed 1.

We define the weight $w(e)$ of each edge $e \in R$ by:

$$w(e) = \begin{cases} p^{-1} & \text{if } e \in Y, \\ -p^{-2} & \text{otherwise,} \end{cases}$$

where $p > 0$ is a parameter to be specified soon. Since only entries of Y have positive weights, the heaviest rectangles M are the square ones with exactly one entry from Y in each row and column. For a $k \times k$ such square we have

$$\sum_{e \in M} w(e) = \frac{k}{p} - \frac{k(k-1)}{p^2} \leq \frac{k}{p}\Big(1 - \frac{k-1}{p}\Big) \leq 1\,.$$

Indeed, if $k \geq p+1$ then the expression in the parenthesis is at most 0, and if $k \leq p$ then both terms are at most 1. Hence, w is a legal weight function, and we obtain

$$\pi(R) \geq \sum_{e \in S_f} w(e) = \frac{|Y|}{p} - \frac{|R|-|Y|}{p^2} = \frac{|Y|}{p}\Big(1 - \frac{|R|-|Y|}{p|Y|}\Big)\,.$$

For $p = 2|R|/|Y|$, the expression in the parenthesis is at least $1/2$, and we obtain

$$\pi(R) \geq \frac{|Y|^2}{4|R|}\,. \qquad \square$$

Hence, one can obtain quadratic lower bounds using the fractional partition number, as well. We now show that this is actually all what we can get using *any* convex rectangle measure.

THEOREM 2.16. *If a rectangle measure μ is convex, then $\mu(R) = O(n^2)$ for every n-dimensional rectangle R.*

PROOF. Associate with each subset $I \subseteq [n] = \{1, \ldots, n\}$ the following two *parity rectangles*.

$$S_I = \{x \mid \oplus_{i \in I} x_i = 0\} \times \{y \mid \oplus_{i \in I} y_i = 1\}$$

and

$$T_I = \{x \mid \oplus_{i \in I} x_i = 1\} \times \{y \mid \oplus_{i \in I} y_i = 0\}\,.$$

Hence, monochromatic rectangles correspond to the case when $|I| = 1$. There are exactly 2^{n+1} parity rectangles.

CLAIM 2.17. Every edge $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ such that $x \neq y$ belongs to 2^{n-1} parity rectangles.

PROOF. For $I \subseteq [n]$, let $v_I \in \{0,1\}^n$ be its incidence vector. If $x \neq y$, then $x \oplus y$ is not a zero vector. Since each nonzero vector is orthogonal over $GF(2)$ to exactly half of the vectors in $\{0,1\}^n$, this implies that precisely 2^{n-1} of the vectors v_I are non-orthogonal to $x \oplus y$. This means that (x, y) belongs to precisely 2^{n-1} of the sets $S_I \cup T_I$. Since $S_I \cap T_I = \emptyset$, we are done. $\square$

Let now R be an n-dimensional rectangle. Let $\mathscr{R}_{\text{par}}$ be the set of all parity rectangles $S_I \cap R$ and $T_I \cap R$ restricted to R. For counting reasons, we shall understand $\mathscr{R}_{\text{par}}$ as a multi-set, elements of $\mathscr{R}_{\text{par}}$ corresponding to different parity rectangles are considered different. Under this provision, $\mathscr{R}_{\text{par}}$ has size 2^{n+1} and by Claim 2.17 every edge in S is contained in exactly 2^{n-1} elements of $\mathscr{R}_{\text{par}}$.

It can be shown that, for every $I \subseteq [n]$, each of two I-parity rectangles can be covered by at most $4|I|^2$ disjoint monochromatic subrectangles (see, for example, Proposition 8.2 in Section 8). Let $\mathscr{M}_I$ be the set of all these (at most $8|I|^2$) monochromatic rectangles, and $\mathscr{M}$ be the union of all $\mathscr{M}_I$'s. Since μ is a rectangle *measure*, we have that $\mu(M) \le 1$ for all $M \in \mathscr{M}$.

Since the rectangles in each $\mathscr{M}_I$ are *disjoint*, we have that each $e \in R$ belongs to precisely 2^{n-1} rectangles in $\mathscr{M}$. Hence, we can obtain a fractional partition of R by setting $r_M = 2^{-(n-1)}$ for all rectangles $M \in \mathscr{M}$, and $r_M = 0$ for all other rectangles. Since $|\mathscr{M}_I| \le 8|I|^2 \le 8n^2$ for each I, the convexity of our measure μ implies that

$$\begin{aligned}\mu(R) &\le \sum_{M\in\mathscr{M}} r_M \cdot \mu(M) \le \sum_{M\in\mathscr{M}} r_M = \sum_I \sum_{M\in\mathscr{M}_I} 2^{-(n-1)} \\ &\le 2^{-(n-1)} \sum_{i=0}^{n} \binom{n}{i} 8n^2 = 8n^2 2^{-(n-1)} 2^n = 16n^2 . \qquad \square\end{aligned}$$

Call a rectangle function μ *additive* if $\mu(R) = \sum_{e\in R} \mu(e)$. It can be shown (Exercise 2.3) that for such functions we have equality in (2.5). A rectangle function μ is *positive* if $\mu(R) > 0$ for every non-empty rectangle R.

Consider now Khrapchenko-type rectangle functions, that is, functions μ of the form

$$\mu(R) = s(R) \cdot \varphi\left(\frac{w(R)}{s(R)}\right), \tag{2.6}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is convex function, $w(R)$ is some "weight" function of rectangles, and $s(R)$ is some additive and positive rectangle function, the "size" of rectangles.

THEOREM 2.18. *A rectangle function μ defined by (2.6) is convex if either $w(R)$ is additive, or $w(R)$ is convex and φ is nondecreasing.*

PROOF. To prove the first claim, assume that both $w(R)$ and $s(R)$ are additive, and let $R_1, \ldots, R_m, r_1 \ldots, r_m$ be a fractional partition of R. Set $s_i = s(R_i)$ and $w_i = w(R_i)$. By Exercise 2.3, we have that $w(R) = \sum_i r_i \cdot w_i$ and $s(R) = \sum_i r_i \cdot s_i$.

For a real convex function φ, numbers x_i in its domain, and positive weights a_i, Jensen's inequality states that

$$\varphi\left(\frac{\sum a_i x_i}{\sum a_i}\right) \le \frac{\sum a_i \varphi(x_i)}{\sum a_i} . \tag{2.7}$$

Applying this we obtain (where the sums are over all i with $r_i > 0$):

$$\begin{aligned}\mu(R) &= s(R) \cdot \varphi\left(\frac{w(R)}{s(R)}\right) \\ &= \left(\sum_i r_i s_i\right) \cdot \varphi\left(\frac{\sum_i r_i w_i}{\sum_i r_i s_i}\right) && \text{Exercise 2.3} \\ &\le \sum_i r_i s_i \cdot \varphi\left(\frac{w_i}{s_i}\right) && \text{(2.7) with } a_i = r_i s_i \text{ and } x_i = w_i/s_i \\ &= \sum_i r_i \mu(R_i) .\end{aligned}$$

If $w(R)$ is convex and φ is nondecreasing, then we can replace the second equality by inequality, and the desired inequality $\mu(R) \le \sum_i r_i \cdot \mu(R_i)$ still holds. $\square$

Note that Khrapchenko's measure (2.4) has the form (2.6), where $\varphi(x) = x^2$ is a nondecreasing convex function, and both $s(R) = |R|$ and $w(R) = |Y \cap R|$ are positive additive functions. Hence, this measure is convex.

By Theorem 2.16, neither taking another sets Y of "special" edges nor taking another convex function φ can lead to a better rectangle measure than Khrapchenko's one: none of them can yield super-quadratic lower bounds on the formula size.

Call rectangle measures of the form (2.6) *polynomial measures of degree k* if $\varphi(x) = x^k$, $k \geq 1$. That is, each such measure has the form

$$\mu(R) = \frac{w(R)^k}{s(R)^{k-1}},$$

where $w(R)$ is some additive rectangle function, and $s(R)$ is some additive and positive rectangle function.

We have seen that no polynomial measure of degree $k \geq 2$ can yield a super-quadratic lower bound. But what about measures of smaller degree—can we then obtain larger lower bounds?

If the weight function $w(R)$ is additive positive, then the answer is negative.

PROPOSITION 2.19. *Let R be an n-dimensional rectangle. If a polynomial measure μ of degree k uses positive weight function $w(R)$, then*

$$\mu(R) \leq (2n)^k.$$

Note that for $k < 2$ this is $o(n^2)$.

PROOF. The normalization condition $\mu(M) \leq 1$, for a monochromatic rectangle M implies that

$$w(M) \leq s(M)^{1-1/k}.$$

Since every n-dimensional rectangle can be (non-disjointly) covered by at most $2n$ monochromatic rectangles $M_{i,\varepsilon}$, we have

$$w(R) \leq \sum_{i,\varepsilon} w(M_{i,\varepsilon}) \leq \sum_{i,\varepsilon} s(M_{i,\varepsilon})^{1-1/k} \leq 2n \cdot s(R)^{1-1/k}.$$

Dividing by $s(R)^{1-1/k}$ and raising to the power k we get the inequality. □

In Proposition 2.19 we have two requirements on the weight function $w(R)$: it must be additive and positive. Let us look at what happens, if we relax any of these two conditions.

First, let us require that $w(R)$ is positive but not necessarily additive. Namely, say that a rectangle function μ is *subadditive* if $\mu(R) \leq \mu(R_1) + \mu(R_2)$, as long as R is a union of two disjoint rectangles R_1 and R_2.

If we define $w(R) = L(R)$ to be the smallest size of a formula separating R, then $w(R)$ is subadditive (show this!) and positive. Take $s(R) := |R|$. The resulting rectangle function $\mu(R) = w(R)^k/|R|^{k-1}$ is normalized since $L(R)$ is normalized. Most boolean functions in n variables, and hence, most n-dimensional rectangles R require $L(R) \geq 2^{n(1-o(1))}$. For such rectangles R, measure $\mu(R)$ gets asymptotically close to the values

$$\frac{2^{kn}}{2^{2n(k-1)}} = 2^{n(2-k)}.$$

Hence, polynomial measured of degree $k < 2$, based on positive and subadditive weight measures $w(R)$ *can* yield even exponential lower bounds!

But what is we would require the weight function $w(R)$ be additive and allow to take also negative values? That such measures, even for $k = 1$, *can* yield quadratic lower bounds, was show in the proof of Theorem 2.15. The measure $w(R) = \sum_{e \in R} w(e)$ constructed there is additive, but takes positive as well as negative values.

RESEARCH PROBLEM 2.20. *Can rectangle measures $\mu(R) = w(R)^k / |R|^{k-1}$ with $1 \leq k < 2$ yield super-quadratic lower bounds when the weight function $w(R)$ is an additive, but* not necessarily nonnegative *rectangle function?*

Most of the known rectangle measures are defined by associating with a boolean function f in question, a matrix $A : S_f \to \mathbb{F}$ over some field $\mathbb{F}$. Given a matrix parameter $p(A)$ (rank, norm, etc.) one obtains a rectangle function $R \mapsto p(A_R)$, where A_R denotes the restriction of A to the rectangle R obtained by setting to 0 all entries outside R. To have a rectangle *measure* we need to normalize this function. This is usually made by taking

$$\mu_A(R) = \frac{p(A_R)}{C}, \quad \text{where} \quad C = \max_M p(A_M)$$

and the maximum is over all monochromatic subrectangles M of the "ambient" rectangle S_f; C is the *normalization constant*.

In matrix terms, the convexity condition turns to: for every sequence $r_1, \ldots, r_t$ of real numbers in $[0, 1]$,

$$A_R = \sum_{i=1}^{t} r_i \cdot A_{M_i} \quad \text{implies} \quad p(A_R) \leq \sum_{i=1}^{t} r_i \cdot p(A_{M_i}). \tag{2.8}$$

Interesting measures can be obtained from matrix norms. A mapping $A \mapsto \|A\|$ is a matrix norm if it satisfies all the properties of vector norms: (i) $\|A\| \geq 0$ with equality if and only if $A = 0$; (ii)$\|rA\| = |r| \cdot \|A\|$ for all numbers r and all matrices A, and (iii) $\|A + B\| \leq \|A\| + \|B\|$ for all matrices A and B.

In particular, any rectangle function of the form $\mu(R) := \|A_R\|$ is convex. Moreover, by Theorem 2.18, if φ is a non-decreasing convex real function and s is an additive rectangle function, then the rectangle function

$$\mu(R) = s(R) \cdot \varphi\left(\frac{\|A_R\|}{s(R)}\right), \tag{2.9}$$

is also convex, and hence cannot give better than $O(n^2)$ lower bounds.

The *spectral norm* of A is defined by

$$\|A\|_2 = \max_{x,y \neq 0} \frac{|x^\top A y|}{\|x\|_2 \|y\|_2},$$

where $\|x\|_2 = (\sum_i x_i^2)^{1/2}$ is the Euclidean norm of x. Associate with every matrix A the following rectangle measure

$$\mu_A(R) = \frac{\|A_R\|_2^2}{\max_M \|A_M\|_2^2}, \tag{2.10}$$

where the maximum is over all monochromatic subrectangles M of R. It can be shown that this measure is convex (Exercise 2.7).

Another important parameter of matrices is their rank. Given an $n \times n$ matrix A (over some field), we can associate with it the following measure for n-dimensional

rectangles:

$$\varrho_A(R) = \frac{\mathrm{rk}(A_R)}{\max_M \mathrm{rk}(A_M)}, \tag{2.11}$$

where the maximum is over all monochromatic subrectangles of R. If $\mathrm{rk}(A_R) = 0$ then we set $\varrho_A(R) = 0$.

Subadditivity of rank implies that these measures are subadditive. But it turns out that rank-based measures are not convex.

PROPOSITION 2.21. *For any even integer n there is an $n \times n$ $(0,1)$ matrix A such that the measure ϱ_A is not convex.*

Note that the rank parameter *itself* is not convex by a simple reason: if $A = r \cdot B$ for some $0 < r < 1$, then $\mathrm{rk}(A) = \mathrm{rk}(B)$, not $\mathrm{rk}(A) \le r \cdot \mathrm{rk}(B)$. For rank based *rectangle measures* this is no more so obvious: the matrix A on both sides of the first equality in (2.8) is the *same*.

PROOF. Let n be even. Take a rectangle $R = R^0 \times R^1$ with $R^0 = \{x_1, \ldots, x_n\}$ and $R^1 = \{y_1, \ldots, y_n\}$ where $x_i = e_i$, $y_i = e_i + e_{i+1}$ and $e_i \in \{0,1\}^{n+1}$ is the ith unit vector. Let A be the complement of the $n \times n$ unit matrix. We define the fractional partition of the rectangle R into monochromatic subrectangles as follows.

For every $i \in [n]$ we take the size-1 rectangle $M_i = \{(x_i, y_i)\}$ and give it weight $r_i = 1$. To cover the rest of the rectangle R, we use rectangles

$$M_I = \{(x_i, y_j) \mid i \in I, j \notin I\}$$

for all $I \subseteq [n]$ of size $|I| = n/2$, and give them weight

$$r_I = \left(4 - \frac{4}{n}\right)\binom{n}{n/2}^{-1}.$$

With such a choice of the numbers r_i and r_I, the left-hand side equality in (2.8) holds, because rectangle M_I contains $n^2/4$ of the $n^2 - n$ ones in A and there are $\binom{n}{n/2}$ such rectangles.

For every $i \in [n]$ we have that $\mu_A(M_i) = 0$ since we have only 0's on the diagonal of A. For every subset I of $[n]$ we have that $\mu_A(M_I) = 1$ since there are no 0's outside the diagonal, implying that A_{M_I} is an all-1 matrix. Hence, on the right hand side of the corresponding inequality in (2.8) for convexity we have the sum of n zeros (the ranks of the size one matrices on the diagonal) and $\binom{n}{n/2}$ terms each being at most $4\binom{n}{n/2}^{-1}$, implying that the right hand sums to at most 4. On the other hand, since $\mathrm{rk}(A)$ is n or $n-1$ (which depends on n and the field), on the left hand side we have $\varrho_A(R) \ge (n-1)/2$: by the construction of R, no monochromatic subrectangle M of R can hit the diagonal in more than one entry, implying that $\mathrm{rk}(A_M) \le 2$. □

We have shown that, for some measures μ_A, the convexity inequality (2.8) fails badly: the right hand side is constant whereas the left had side is $\Omega(n)$. Since the measures μ_A based on the rank are not convex, Theorem 2.16 does not apply for them. Still, Razborov (1992b) proved that these measures belong to the class of so-called submodular measures, and none of them can yield larger than $O(n)$ lower bound.

2.7. Complexity is not submodular

In order to prove that some boolean function f requires large formulas, one tries to find some clever "combinatorial" measure μ on the set of all boolean functions satisfying two conditions: $\mu(f)$ is a lower bound on the size of any circuit computing f, and $\mu(f)$ can be non-trivially bounded from below at some *explicit* boolean functions f. One class of such measures, proposed by Mike Paterson, is the following.

Let $\mathscr{B}_n$ be the set of all boolean function in n variables. A *formal complexity measure* of boolean functions is a mapping $\mu : \mathscr{B}_n \to \mathbb{R}$ which assigns positive values to each boolean function. The requirements are that μ is *normalized*, that is, assigns each literal a value ≤ 1, and satisfies the following two simple rules for all $f, g \in \mathscr{B}_n$:

$$\mu(f \vee g) \leq \mu(f) + \mu(g); \tag{2.12}$$

$$\mu(f \wedge g) \leq \mu(f) + \mu(g). \tag{2.13}$$

Note that the minimal formula size $L(f)$ itself is a formal measure with both inequalities being equalities.

In order to understand what measures are "good" (can lead to large lower bounds) it is important to understand what measures are "bad". We have already seen that convex measures are bad. There is another class of bad measures—submodular ones.

A formal complexity measure $\mu : \mathscr{B}_n \to \mathbb{R}$ is *submodular* if it is normalized and for all $f, g \in \mathscr{B}_n$,

$$\mu(f \wedge g) + \mu(f \vee g) \leq \mu(f) + \mu(g). \tag{2.14}$$

Note that this condition is stronger than both (2.12) and (2.12).

THEOREM 2.22. *If μ is a submodular measure on $\mathscr{B}_n$, then $\mu(f) \leq O(n)$ for each $f \in \mathscr{B}_n$.*

PROOF. Let $\boldsymbol{g}_d$ be a random boolean function in d variables $x_1, \ldots, x_d$. That is, we choose $\boldsymbol{g}_d$ randomly and uniformly from $\mathscr{B}_d$. We are going to prove by induction on d that

$$\mathrm{E}\left[\mu(\boldsymbol{g}_d)\right] \leq d + 1. \tag{2.15}$$

Given a variable x_i, set $x_i^1 := x_i$ and $x_i^0 := \overline{x}_i$.

Base. $d = 1$. Here we have $\mu(g(x_1)) \leq 2$ for any $g(x_1)$. This follows from the normalization condition if g is a variable x_1 or its negation $\overline{x}_1$. By the subadditivity we also have

$$\mu(0) + \mu(1) = \mu(x_1 \wedge \overline{x}_1) + \mu(x_1 \vee \overline{x}_1) \leq \mu(x_1) + \mu(\overline{x}_1) \leq 2$$

which proves $\mu(g(x_1)) \leq 2$ in the remaining case when g is a constant.

Inductive step. Assume that (2.15) is already proved for d. Let the symbol $\approx$ mean that two random functions have the same distribution. Note that

$$\boldsymbol{g}_{d+1} \approx \left(\boldsymbol{g}_d^0 \wedge x_{d+1}^0\right) \vee \left(\boldsymbol{g}_d^1 \wedge x_{d+1}^1\right), \tag{2.16}$$

where $\boldsymbol{g}_d^0$ and $\boldsymbol{g}_d^1$ are two independent copies of $\boldsymbol{g}_d$. By duality,

$$\boldsymbol{g}_{d+1} \approx \left(\boldsymbol{g}_d^0 \vee x_{d+1}^0\right) \wedge \left(\boldsymbol{g}_d^1 \vee x_{d+1}^1\right). \tag{2.17}$$

By the linearity of expectation, we obtain from (2.16) and (2.12) (remember that the latter is a consequence of the submodularity condition) that

$$\mathrm{E}\left[\mu(\boldsymbol{g}_{d+1})\right] \leq \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^0 \wedge x_{d+1}^0\right)\right] + \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^1 \wedge x_{d+1}^1\right)\right] \tag{2.18}$$

and similarly from (2.17) and (2.13),

$$\mathrm{E}\left[\mu(\boldsymbol{g}_{d+1})\right] \le \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^0 \vee x_{d+1}^0\right)\right] + \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^1 \vee x_{d+1}^1\right)\right]. \tag{2.19}$$

Summing (2.18), (2.19) and applying consecutively (2.14), normalization of μ and the inductive assumption (2.15), we obtain

$$\begin{aligned} 2\cdot\mathrm{E}\left[\mu(\boldsymbol{g}_{d+1})\right] &\le \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^0 \wedge x_{d+1}^0\right)\right] + \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^0 \vee x_{d+1}^0\right)\right] + \\ &\quad \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^1 \wedge x_{d+1}^1\right)\right] + \mathrm{E}\left[\mu\left(\boldsymbol{g}_d^1 \vee x_{d+1}^1\right)\right] \\ &\le \mathrm{E}\left[\mu(\boldsymbol{g}_d^0)\right] + \mu(x_{d+1}^0) + \mathrm{E}\left[\mu(\boldsymbol{g}_d^1)\right] + \mu(x_{d+1}^1) \\ &\le 2\cdot\mathrm{E}\left[\mu(\boldsymbol{g}_d)\right] + 2 \\ &\le 2d+4. \end{aligned}$$

This completes the proof of (2.15). But this inequality only says that the expected value of $\mu(\boldsymbol{g}_n)$ does not exceed $n+1$ for a *random* function $\boldsymbol{g}_n$, whereas our goal is to give an upper bound on $\mu(f_n)$ for *each* function f_n. So, we must somehow "de-randomize" this result. To achieve this goal, observe that every function $f_n \in F_n$ can be expressed in the form

$$f_n = (\boldsymbol{g}_n \wedge (\boldsymbol{g}_n \oplus f_n \oplus 1)) \vee ((\boldsymbol{g}_n \oplus 1) \wedge (\boldsymbol{g}_n \oplus f_n). \tag{2.20}$$

But $\boldsymbol{g}_n \approx \boldsymbol{g}_n \oplus f_n \oplus 1 \approx \boldsymbol{g}_n \oplus 1 \approx \boldsymbol{g}_n \oplus f_n$. So, applying to (2.20) the inequalities (2.12) and (2.13), averaging the result over $\boldsymbol{g}_n$ and applying (2.15) with $d = n$, we obtain $\mu(f_n) = \mathrm{E}\left[\mu(f_n)\right] \le 4\cdot\mathrm{E}\left[\mu(\boldsymbol{g}_n)\right] \le 4n+4$, as desired. □

2.8. The drag-along principle

Suppose we want to prove that a boolean function f has high complexity, say, requires large DeMorgan formulas over $\wedge, \vee, \neg$. If the function is indeed hard, then it should have some *specific* properties forcing its formulas be large, that is, fooling every small formula to make an error.

It turns out that formal complexity measures cannot capture any specific properties of boolean functions. When using such measures, every lower bound for a given function f must also prove that many other unrelated functions have large complexity. Thus, we cannot use any special properties of our function!

THEOREM 2.23 (The Drag-Along Principle). *Suppose μ is a formal complexity measure and there exists a function $f \in \mathscr{B}_n$ such that $\mu(f) > s$. Then, for at least $1/4$ of all g in $\mathscr{B}_n$, $\mu(g) > s/4$.*

PROOF. Let g be any function in $\mathscr{B}_n$. Define $f = h \oplus g$ where $h = f \oplus g$. Then,

$$\mu(f) \le \mu(g) + \mu(\neg g) + \mu(h) + \mu(\neg h). \tag{2.21}$$

This follows from (2.12) and (2.13) and the definition of parity,

$$f = (f \oplus g) \oplus g = h \oplus g = (h \wedge g) \vee (\neg h \wedge \neg g).$$

By way of contradiction assume that the set $\mathscr{G} = \{g \in \mathscr{B}_n \mid \mu(g) < s/4\}$ contains *more* than $3/4$ of all function in $\mathscr{B}_n$. If we pick the above function g randomly in $\mathscr{B}_n$ with probability $|\mathscr{B}_n|^{-1}$, then $\neg g, h, \neg h$ are also random elements of $\mathscr{B}_n$ (though not independent) each with the same probability. Using the trivial union bound we have

$$\Pr[\text{some of } h, \neg h, g, \neg g \text{ is not in } \mathscr{G}] < 4\cdot\frac{1}{4} = 1.$$

Thus, there must be at least one choice for g such that all four functions $h, \neg h, g, \neg g$ belong to $\mathscr{G}$, that is, have measure $< s/4$. By (2.21), this implies that $\mu(f) < s$, which is a contradiction. □

Thus, if one uses a formal complex measure to prove that f is complex, then one proves much more! That is, *any* lower bounds proof for formulas, based on some formal complexity measure $\mu(f)$, automatically fulfills the "largeness" condition of so-called "natural proofs" (we sketch this important concept in Appendix): if $\mu(f)$ is large for some *specific* function f, then $\mu(\mathbf{f})$ must be also large for a *random* function $\mathbf{f}$. So, for such a proof to be "unnatural," the predicate "$\mu(f) \geq t$" must be not constructive, that is, must be not computable in exponential(!) time $2^{O(n)}$.

Exercises

Ex. 2.1. Let $n = 2m+1$, and consider the majority function MAJ_n, which outputs 1 iff $x_1+\ldots+x_n \geq m+1$. Use Khrapchenko's theorem to show that this function requires DeMorgan formulas of size $\Omega(n^2)$. *Hint*: Consider the subrectangle $A \times B \subseteq S_f$ of $f = \mathrm{MAJ}_n$ with $A = \{a : |a| = m+1\}$ and $B = \{b : |b| = m\}$.

Ex. 2.2. Show that Khrapchenko's theorem cannot yield larger than quadratic lower bounds. *Hint*: Each vector in $\{0,1\}^n$ has only n neighbors, that is, vectors y with $\mathrm{dist}(x,y) = 1$.

Ex. 2.3. Show that, if μ is an additive rectangle function then, for every fractional partition $R = \sum_i r_i \cdot R_i$, we have that $\mu(R) = \sum_{i=1}^{t} r_i \cdot \mu(R_i)$.

Ex. 2.4. Show that any linear combination of convex rectangle functions is a convex rectangle function.

Ex. 2.5. Let $a(R)$ and $b(R)$ be arbitrary additive nonnegative rectangle functions, and consider the rectangle function

$$\mu(R) = \frac{f(a(R))}{g(b(R))},$$

where $f, g : \mathbb{R} \to \mathbb{R}$ are non-decreasing, and f is *sub-multiplicative* in that $f(x \cdot y) \leq f(x) \cdot f(y)$.

Show that, if μ is normalized then, for every n-dimensional rectangle R, we have that $\mu(R) \leq \varphi(2n)$.

Hint: Consider a covering of R by $2n$ (overlapping) monochromatic rectangles.

Ex. 2.6. Consider rectangle measures of the form $\mu(R) = w(R)^k/|R|^{k-1}$, where $w(R)$ is an arbitrary subadditive rectangle function: if $R = R_1 \cup \cdots \cup R_t$ is a partition of R, then $w(R) \leq w(R_1) + \cdots + w(R_t)$. Recall that Khrapchenko's measure has this form with $k = 2$ and $w(R)$ being the number of pairs $(x,y) \in R$ with $\mathrm{dist}(x,y) = 1$. The goal of this exercise is to show that, for $k > 2$, such measures fail badly: they cannot yield even non-constant lower bounds!

Namey, let S_n be the rectangle of the parity function in n variables. Show that, for every constant $k > 2$ there is a constant $c = c_k$ (depending only on k, not on n) such that $\mu(S_n) \leq c$.

Hint: Consider the following "first difference" decomposition of S_n. For $1 \leq i < n$, $\varepsilon \in \{0,1\}$ and a string $u \in \{0,1\}^i$, let $R^i_{u,\varepsilon}$ be the rectangle consisting of all pairs (x,y) such that $x_{i+1} = \varepsilon$, $y_{i+1} = 1-\varepsilon$ and $x_j = y_j = u_j$ for all $j = 1,\ldots,i$. Use the normalization condition $\mu(R^i_{u,\varepsilon}) \leq 1$ to show that the sum of μ-measures of these rectangles is constant.

Ex. 2.7. Prove that the spectral norm measure $\mu_A(R)$, defined by (2.10), is convex.
Hint: Show that the rectangle function $s(R) = \|\boldsymbol{x}_R\|_2^2 \cdot \|\boldsymbol{y}_R\|_2^2$ is additive, and use Theorem 2.18.

Ex. 2.8. Let A and B be two disjoint subsets of $\{0,1\}^n$. Define the set $A \otimes B$ to contain all pairs (a,b) of vectors $a \in A$ and $b \in B$ such that a and b differ in exactly one bit. Define now the measure

$$\mu(f) = \frac{|A \otimes B|^2}{|A| \cdot |B|},$$

where $A = f^{-1}(0)$ and $B = f^{-1}(1)$. Observe that this is the measure used in Khrapchenko's theorem (Theorem 2.12). Prove that $\mu(f)$ is a formal complexity measure.

Hint: Argue by induction as in the proof of Rychkov's lemma (Lemma 1.8). In the induction step use the inequality

$$\frac{c_1^2}{a_1 \cdot b} + \frac{c_2^2}{a_2 \cdot b} \geq \frac{(c_1 + c_2)^2}{(a_1 + a_2) \cdot b}$$

which can be checked by a cross-multiplication.

Bibliographic Notes

Theorem 2.8 is due to Andreev (1987). The version of Nechiporuk's theorem (Theorem 2.10) we presented is due to Mike Paterson (unpublished) and Zwick (1991a). We followed the exposition in Boppana and Sipser (1990). Contents of Section 2.6 are from Hrubes *et al.* (2009). Theorem 2.16 is due to Karchmer, Kushilevitz and Nisan (1995). That Khrapchenko's measure is a formal complexity measure (Example 2.8) was observed by Mike Paterson. Theorem 2.22 is due to Razborov (1992b). Theorem 2.23 is due to Razborov and Rudich (1997). Its name "drag-along principle" is due to Richard Lipton.

CHAPTER 3

Monotone Formulas

We now consider monotone formulas, that is, formulas with fanin-2 AND and OR gates. Such formulas can only compute monotone boolean functions, that is, functions f such that $f(x) \leq f(y)$ as long as $x_i \leq y_i$ for all positions i. Let $L_+(f)$ denote the smallest leafsize of (=the smallest number of leaves in) a monotone formula computing f. Just like for DeMorgan formulas (with AND, OR and NOT gates) it is possible to lower bound $L_+(f)$ by a monotone decomposition number $\mathsf{D}_+(S_f)$ of the rectangle $S_f = f^{-1}(0) \times f^{-1}(1)$.

Recall that an n-dimensional *rectangle* is just a Cartesian product $R = S \times T$ of subsets $S, T \subseteq \{0,1\}^n$, $S \cap T = \emptyset$. A rectangle R is *monochromatic* if all its edges $(a,b) \in R$ are separated by some literal[1] in that $x_i^{\sigma}(a) = 0$ and $x_i^{\sigma}(b) = 1$. If this separation is done by a monotone literal, that is, by a variable x_i (and not by its negation $\neg x_i$) then we call R *monotone*. *rectangle!monotone* That is, a monochromatic rectangle is monotone if there is a position i such that $a_i = 0$ and $b_i = 1$ for all $(a,b) \in M$.

The *monotone partition number* $\mathsf{D}_+(S)$ of a rectangle S is the smallest number t such that S can be decomposed into t disjoint monotone monochromatic rectangles. Note that this measure is defined not for all rectangles. A simplest counterexample is a rectangle $S = \{(\mathbf{1}, \mathbf{0})\}$, consisting of just one pair of vectors. If, however, $S \subseteq S_f := f^{-1}(0) \times f^{-1}(1)$ for a *monotone* boolean function f then, for every $(a,b) \in S$, there must be a position i for which $a_i = 0$ and $b_i = 1$. Hence, in this case $\mathsf{D}_+(S)$ is well defined.

LEMMA 3.1. *For every monotone boolean function f and for every rectangle $S \subseteq S_f$ we have that*

$$L_+(f) \geq \mathsf{D}_+(S).$$

PROOF. The proof is the same as that of Rychkov's lemma (Lemma 1.8). The only difference is the basis case $L_+(f) = 1$. Since in this case we have no negated variables at all, the function f must be just a single variable x_i, implying that S_f itself is a monotone monochromatic rectangle. The induction step is the same. □

3.1. The rank lower bound

To bound $\mathsf{D}_+(S_f)$ from bellow, Razborov (1990) suggested to use rank arguments, where the rank is over some (fixed in advance) field $\mathbb{F}$. Given a rectangle R, we denote by A_R the matrix which is obtained from the matrix A by changing to 0 all its entries $(u,v) \notin R$. Let also $\mathrm{rk}(A)$ denote the rank of A over $\mathbb{F}$.

[1] As before, $x_i^1 = x_i$ and $x_i^0 = \neg x_i$.

LEMMA 3.2. *Let f be a monotone boolean function, $I \subseteq f^{-1}(0)$ and $J \subseteq f^{-1}(1)$. Then for every $|I| \times |J|$ matrix $A \neq 0$,*

$$L_+(f) \geq \frac{\mathrm{rk}(A)}{\max_R \mathrm{rk}(A_R)}, \tag{3.1}$$

here the maximum is over all monotone monochromatic subrectangles of $I \times J$.

PROOF. Let $t = L_+(f)$. By Lemma 3.1 we know that there must exists a set $\mathscr{R}$ of $|\mathscr{R}| \leq t$ monotone monochromatic rectangles such that all the rectangles in $\mathscr{R}$ are pairwise disjoint, and their union covers the whole rectangle $I \times J$. So $A = \sum_{R \in \mathscr{R}} A_R$, and hence, by the subadditivity of rank,

$$\mathrm{rk}(A) \leq \sum_{R \in \mathscr{R}} \mathrm{rk}(A_R) \leq |\mathscr{R}| \cdot \max_{R \in \mathscr{R}} \mathrm{rk}(A_R)$$

implying the desired lower bound on $|\mathscr{R}|$ and hence, on $t = L(f)$. □

It is clear that the same lower bound (3.1) also holds for *non-monotone* formulas, if we do not require monochromatic rectangles be monotone. However, Razborov (1992b) has proved that in this (non-monotone) case the result is useless: for any boolean function f in n variables, the fraction on the right-hand side of (3.1) is then a submodular measure, and hence, cannot exceed $O(n)$ (see Theorem 2.22). Fortunately, in the monotone case, Lemma 3.2 *can* give large lower bounds, and we are going to show this in the next two sections. But before, let us make a note on notation.

Boolean functions $f : \{0,1\}^n \to \{0,1\}$ are predicates on the n-cube. It is however often more convenient to identify each vector $a \in \{0,1\}^n$ with the set $\{i \mid a_i = 1\}$ of its 1-positions, and look at boolean functions as predicates on the family of all subsets of $[n] = \{1, \ldots, n\}$. In these terms, (a, b) is separated by a variable x_i iff $i \notin a$ and $i \in b$.

3.2. Lower bounds for quadratic functions

A monotone quadratic function of a graph $G = ([n], E)$ is a monotone boolean function

$$f_G(x_1, \ldots, x_n) = \bigvee_{\{i,j\} \in E} x_i \wedge x_j \,.$$

Note that $f_G(a) = 0$ iff $I = \{i \mid a_i = 1\}$ is an independent set in G. It is clear that $L_+(f_G) \leq |E|$ for any graph G, but for some graphs this trivial upper bound is very far from the truth.

EXAMPLE 3.3. Let $G = ([n], E)$ be a complete bipartite graph with $E = S \times T$, $S \cap T = \emptyset$ and $|S| = |T| = n/2$. Then $|E| = n^2/4$, but f_G can be computed by a monotone formula

$$F(x_1, \ldots, x_n) = \left(\bigvee_{i \in S} x_i \right) \wedge \left(\bigvee_{j \in T} x_j \right)$$

of leafsize $|S| + |T| = n$.

So, a natural question is: what quadratic functions require monotone formulas of super-linear size? We will use the rank argument to show that such are boolean functions defined by dense graphs without 4-cycles. A 4-cycle in G is a set v_1, v_2, v_3, v_4 of four distinct vertices such that v_1v_2, v_2v_3, v_3v_4 and v_4v_1 are edges of G.

THEOREM 3.4. *If $G = (V, E)$ is a triangle-free graph without 4-cycles, then*

$$L_+(f_G) \geq |E|/2 \,.$$

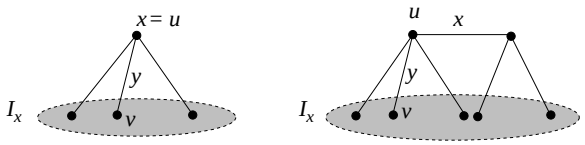


FIGURE 1. The cases when $x \in V$ and when $x \in E$.

PROOF. We look at vertices as one-element and edges as two-element sets. For a vertex $x \in V$, let I_x be the set of its neighbors. For an edge $x \in E$, let I_x be the set of all its *proper* neighbors; that is, $v \in I_x$ precisely when $v \notin x$ and v is adjacent with an endpoint of x. Since G has no triangles and no 4-cycles, the sets I_x are independent sets, and must be rejected by f. We will concentrate on only these independent sets.

Let A be a $(0,1)$ matrix whose rows correspond to independent sets I_x with $x \in V \cup E$, and columns to edges $y \in E$. The entries are defined by

$$A[x,y] = \begin{cases} 1 & \text{if } x \cap y \neq \emptyset, \\ 0 & \text{if } x \cap y = \emptyset. \end{cases}$$

CLAIM 3.5. If M is a monotone monochromatic rectangle, then $\mathrm{rk}(A_M) \leq 2$.

PROOF. Since $M = M^0 \times M^1$ is monotone and monochromatic, there must be a vertex $v \in V$ such that

$$v \notin I_x \text{ and } v \in y \text{ for all } x \in M^0 \text{ and } y \in M^1.$$

Hence, for each $x \in M^0$, we have two possible cases.

Case 1: $v \in x$. Since $v \in y$ for all $y \in M^1$, in this case we have that $x \cap y \supseteq \{v\} \neq \emptyset$, implying that $A_M[x,y] = 1$ for all $y \in M^1$. That is, in this case the x-th row of A_M is the all-1 row.

Case 2: $v \notin x$. We claim that in this case the x-th row of A_M must be the all-0 row. To show this, assume that $A_M[x,y] = 1$ for some $y \in M^1$. Then $x \cap y \neq \emptyset$, implying that x and y must share a common vertex $u \in x \cap y$ (see Fig. 1). Moreover, $u \neq v$ since $v \notin x$. Together with $v \in y$, this implies that $y = \{u, v\}$. But then $v \in I_x$, a contradiction. □

By Lemma 3.2, it remains to show that the entire matrix A has full column-rank $|E|$ over $GF(2)$.

Take an arbitrary subset $\emptyset \neq F \subseteq E$ of edges. We have to show that the columns of the submatrix A' of A corresponding to the edges in F cannot sum up to the all-0 column over $GF(2)$. If F is not an even factor, that is, if the number of edges in F containing some vertex v is odd, then the row of v in A' has an odd number of 1's, and we are done.

Hence, we may assume that F *is* an even factor. Take an arbitrary edge $x = uv \in F$, and let $H \subseteq F$ be the set of edges in F incident to at least one endpoint of x. Since both vertices u and v have even degree (in F), the edge x has a nonempty intersection with an *odd* number of edges in F: one intersection with itself and an even number of intersections with the edges in $H - \{x\}$. Hence, the row of x in A' contains an odd number of 1's, as desired. □

Explicit constructions of dense triangle-free graphs without 4-cycles are known. Such is, for example, the point-line incidence $n \times n$ graph H of a projective plane

$PG(2,q)$ for a prime power q. Such a plane has $n = q^2+q+1$ points and n subsets of points (called lines). Every point lies in $q+1$ lines, every line has $q+1$ points, any two points lie on a unique line, and any two lines meet is a unique point. Now, if we put points on the left side and lines on the right, and joint a point x with a line L by an edge iff $x \in L$, then the resulting bipartite $n \times n$ graph will have $(q+1)n = \Theta(n^{3/2})$ edges and contain no 4-cycles. For this graph Theorem 3.4 yields

COROLLARY 3.6. $L_+(f_H) = \Theta(n^{3/2})$.

3.3. A super-polynomial lower bound

Let $G = (U, V, E)$ be a bipartite graph with $V = \{1, \ldots, n\}$ and $U = \{n+1, \ldots, 2n\}$. For a subset $S \subseteq U$ of vertices on the left part, let

$$\Gamma_1(S) := \{j \in V \mid (i,j) \in E \text{ for all } i \in S\}$$

denote the set of its *common neighbors* of S on the right part. Associate with G a monotone boolean function f_G in $2n$ variables defined by:

$$f_G(x_1, \ldots, x_{2n}) = \bigvee_{S \subseteq U, |S| \le k} \bigwedge_{i \in S \cup \Gamma_1(S)} x_i \,.$$

That is, $f_G(x) = 1$ iff there is a subset $S \subseteq U$ of size $|S| \le k$ such that $x_i = 1$ for all $i \in S \cup \Gamma_1(S)$.

By its definition, the function f_G can be computed by a trivial monotone formula of leafsize at most $\sum_{i=1}^{k} i\binom{n}{i} \le n^{O(k)}$. We will show that, for some explicit graphs G, this trivial upper bound is almost optimal. The proof will be based on the (quite often used in circuit complexity) fact that so-called "disjointness matrices" have large rank.

3.3.1. Disjointness matrices. The k-disjointness matrix $D_{n,k}$ is a $(0,1)$ matrix whose rows as well as columns are labeled by all $\sum_{i=0}^{k}\binom{n}{i}$ subsets a of $[n]$ of size at most k; the entry in the a-th row and b-th column is defined by:

$$D_{n,k}[a,b] = \begin{cases} 0 & \text{if } a \cap b \neq \emptyset, \\ 1 & \text{if } a \cap b = \emptyset. \end{cases}$$

This matrix plays an important role in computational complexity.

LEMMA 3.7. *The k-disjointness matrix $D = D(n,k)$ has full rank over $GF(2)$, that is,*

$$\mathrm{rk}(D_{n,k}) = \sum_{i=0}^{k} \binom{n}{i}.$$

PROOF. Let $N = \sum_{i=0}^{k}\binom{n}{i}$. We must show that the rows of D are linearly independent over $GF(2)$, i.e., that for any non-zero vector $\lambda = (\lambda_{I_1}, \lambda_{I_2}, \ldots, \lambda_{I_N})$ in $GF(2)^N$ we have $\lambda \cdot D \neq 0$. For this, consider the following polynomial:

$$f(x_1, \ldots, x_n) := \sum_{|I| \le k} \lambda_I \prod_{i \in I} x_i.$$

Since $\lambda \neq 0$, at least one of the coefficients λ_I is nonzero, and we can find some I_0 such that $\lambda_{I_0} \neq 0$ and I_0 is *maximal* in that $\lambda_I = 0$ for all $I \supset I_0$. Assume w.l.o.g. that $I_0 = \{1, \ldots, t\}$, and make in the polynomial f the substitution $x_i = 1$ for all $i \notin I_0$. After this substitution has been made, a *non-zero* polynomial over the first t variables $x_1, \ldots, x_t$ remains such that the term $x_1 x_2 \cdots x_t$ is left untouched (here we use the

maximality of I_0). Hence, after the substitution we obtain a polynomial which is 1 for some assignment $(a_1,\dots,a_t)$ to its variables. But this means that the polynomial f itself takes the value 1 on the assignment $b=(a_1,\dots,a_t,1,\dots,1)$. Hence,

$$1=f(b)=\sum_{|I|\le k}\lambda_I\prod_{i\in I}b_i.$$

Let $J_0:=\{i\,:\,a_i=0\}$. Then $|J_0|\le k$ and, moreover, $\prod_{i\in I}b_i=1$ if and only if $I\cap J_0=\emptyset$, which is equivalent to $D_{I,J_0}=1$. Thus,

$$\sum_{|I|\le k}\lambda_I D_{I,J_0}=1,$$

meaning that the J_0-th coordinate of the vector $\lambda\cdot D$ is non-zero. □

3.3.2. A lower bound for Paley graphs. A bipartite graph $G=(U,V,E)$ is *k-separated* if, for every two nonempty subsets $S,T\subseteq U$ of size at most k, we have that

$$S\cap T=\emptyset \quad\text{iff}\quad \Gamma_1(S)\cap\Gamma_0(T)\neq\emptyset, \tag{3.2}$$

where $\Gamma_0(S):=\{j\in V\mid (i,j)\in E \text{ for no } i\in S\}$ is the set of all *common non-neighbors* of S. That is, for every two disjoint subsets S and T of size at most k on the left part there is a vertex $v\in V$ on the right part such that v is joined by an edge to all vertices in S and to none of the vertices in T. Explicit bipartite $n\times n$ graphs, which are k-separated for $k=\Theta(\log n)$, are known. Such are, for example, Paley graphs.

THEOREM 3.8. *If G is k-separated then $L_+(f_G)\ge n^{\Omega(k)}$.*

PROOF. For a constant $\sigma\in\{0,1\}$, define X_σ to be the set of all vectors $x\in\{0,1\}^{2n}$ such that, for some subset $S\subseteq U$ of size $|S|\le k$,

$$x_i=\sigma \quad\text{iff}\quad i\in S\cup\Gamma_\sigma(S).$$

That is, $x\in X_0$ iff there is a subset S of at most k vertices on the left side of G such that x has 0's exactly in positions $i\in S\cup\Gamma_0(S)$, and $x\in X_1$ iff there is a subset S of at most k vertices on the left side of G such that x has 1's exactly in positions $i\in S\cup\Gamma_1(S)$. Since no vector can have 0 and 1 in the same position, (3.2) implies that $X_0\cap X_1=\emptyset$. Hence, $X_0\times X_1$ is a rectangle.

By the definition of $f=f_G$, we have that $f(y)=1$ for all $y\in X_1$ and, by (3.2), we also have that $f(x)=0$ for all $x\in X_0$. Hence, $X_0\times X_1$ is a subrectangle of $f^{-1}(0)\times f^{-1}(1)$.

Let $\mathscr{R}$ be a decomposition of $X_0\times X_1$ into monotone monochromatic subrectangles. For each $R\in\mathscr{R}$ there is a separating position $i=i_R\in U\cup V=\{1,\dots,2n\}$ such that $x_i=0$ and $y_i=1$ for all pairs $(x,y)\in R$. Hence, we can assign to each pair (x,y) in $X_0\times X_1$ its position $i_{(x,y)}$: this is the separating position i_R of the unique rectangle $R\in\mathscr{R}$ such that $(x,y)\in R$. Define now the $|X_0|\times|X_1|$ matrix A by

$$A[x,y]=\begin{cases}0 & \text{if } i_{(x,y)}\in U,\\ 1 & \text{if } i_{(x,y)}\in V.\end{cases}$$

That is, to determine the value of the (x,y)-entry of A we take the unique rectangle R containing (x,y) and set the value to 1 iff the separating position i_R of this rectangle belongs to the right part V of the bipartition.

Associate with each vector $x\in X_0$ the set $S_x=\{i\in U\mid x_i=0\}$, and with each vector $y\in X_1$ the set $T_y=\{i\in U\mid y_i=1\}$. By (3.2), we have that then

$$A[x,y]=1 \quad\text{iff}\quad i_{(x,y)}\in V \quad\text{iff}\quad S_x\cap T_y=\emptyset.$$

Hence, A is the disjointness matrix $D_{n,k}$, and Lemma 3.7 implies that A has full rank over $GF(2)$, that is, $\mathrm{rk}(A) = \sum_{i=0}^{k} \binom{n}{i}$.

On the other hand, since for each rectangle $R \in \mathscr{R}$, its separating position i_R belongs either to U or to V (but not to both), we have that $\mathrm{rk}(A_R) \leq 1$ for all $R \in \mathscr{R}$, and Lemma 3.2 yields the desired lower bound on $L_+(f)$. □

As mentioned above, explicit k-separated bipartite graphs with $k = \Omega(\log n)$ are known. Such are, for example, Paley graphs.

Let n be a n odd prime congruent to 1 modulo 4. A *Paley graph* is a bipartite graph $G = (V_1, V_2, E)$ with parts $V_1 = V_2 = GF(n)$ where two nodes, $x \in V_1$ and $y \in V_2$, are joined by an edge if and only if $x - y$ is a non-zero square in $GF(n)$, that is, if $x - y = z^2 \bmod n$ for some $z \in GF(n)$, $z \neq 0$. The condition $n \equiv 1 \bmod 4$ is only to ensure that -1 is a square in the field, so that the resulting graph is undirected.

It is known that Paley $n \times n$ graphs over $GF(n)$ are k-separated as long as $k2^k < \sqrt{n}$, and in particular, are k-separated for $k = \Omega(\log n)$. This is a well known result, and is proved using some deep results regarding sums of quadratic characters $\chi(x) = x^{(n-1)/2}$ over $GF(n)$.

COROLLARY 3.9. *If G is a bipartite $n \times n$ Paley graph, then*

$$L_+(f_G) \geq n^{\Omega(\log n)}.$$

Bibliographic Notes

Lemma 3.2 was proved in Razborov (1990). Theorem 3.4 was proved in [**81**]. The presented proof of Lemma 3.7 was given by Razborov (1987). Theorem 3.8 was proved by Gál (1998).

CHAPTER 4

Monotone Circuits

We now consider monotone circuits, that is, circuits with fanin-2 AND and OR gates. Like monotone formulas, such circuits can only compute monotone boolean functions. The difference from monotone formulas is that now the fan-outs of gates may be arbitrary, not just 1 as in the case of formulas. That is, a result computed at some gate can be used many times with no need to recompute it again and again, as in the case of formulas. This additional future makes the lower bounds problem more difficult.

In this chapter, all considered boolean functions are assumed to be monotone.

4.1. Switching lemma for monotone forms

Recall that a monotone k-CNF (conjunctive normal form) is an And of an arbitrary number of monotone clauses, each being an Or of at most k variables. Dually, a monotone k-DNF is an Or of an arbitrary number of monomials, each being an And of at most k variables. Note that in k-CNFs we allow clauses shorter than k.

In an *exact k-CNF* we require that all clauses have *exactly k* distinct variables; *exact k-DNF* is defined similarly. For two boolean functions f and g in n variables, we write $f \le g$ if $f(x) \le g(x)$ for all input vectors x. For a CNF/DNF C we will denote by $|C|$ the number of clauses/monomials in it.

Our goal is to show that complex monotone function, that is, monotone functions requiring large monotone circuits cannot be "simple" in a sense that they cannot be approximated by small CNFs and DNFs. The proof of this will be based on the following "switching lemma" allowing us to switch between CNFs and DNFs, and vice versa.

LEMMA 4.1 (Switching Lemma). *For every s-CNF f_0 there is an r-DNF f_1 and an exact $(r+1)$-DNF D such that*

$$f_1 \le f_0 \le f_1 \vee D \quad \textit{and} \quad |D| \le s^{r+1}. \tag{4.1}$$

Dually, for every r-DNF f_1 there is an s-CNF f_0 and an exact $(s+1)$-CNF C such that

$$f_0 \wedge C \le f_1 \le f_0 \quad \textit{and} \quad |C| \le r^{s+1}. \tag{4.2}$$

PROOF. We prove the first claim (the second is dual). Let $f_0 = C_1 \wedge \cdots \wedge C_\ell$ be an s-CNF; hence, each clause C_i has $|C_i| \le s$ variables. It will be convenient to identify clauses and monomials with the *sets* of indices of their variables. We say that a monomial M *pierces* a clause C_i if $M \cap C_i \neq \emptyset$.

We associate with the CNF f_0 the following tree T of fan-out at most s. This is a DNF-tree for f_0 we already defined in Section 1.5; the only difference is that now we have no negated variables.

The first node of T corresponds to the first clause C_1, and the outgoing $|C_1|$ edges are labeled by the variables from C_1. Suppose we have reached a node v, and let M be the monomial consisting of the labels of edges from the root to v. If M pierces all the

clauses of f_0, then v is a leaf. Otherwise, let C_i be the *first* clause such that $M \cap C_i = \emptyset$. Then the node v has $|C_i|$ outgoing edges labeled by the variables in C_i.

Note that each path from the root to a leaf of T corresponds to a monomial of f_0 (since each such path intersects all clauses). More important is that also the converse holds: each monomial of f_0 must appear as a path from the root to a leaf. Thus, we have just represented the DNF of f_0 as a tree, implying that $T(x) = f_0(x)$ for all input vectors $x \in \{0,1\}^n$. But some paths (monomials) may be longer than $r+1$. So, we now cut-off these long paths.

Namely, let f_1 be the OR of all paths of length at most r ending in leafs, and D be the set of all paths of length exactly $r+1$. Observe that:

(i) every monomial of f_1 is also a monomial of f_0, and
(ii) every monomial of f_0, which is not a monomial of f_1, must contain (is an extension of) at least one monomial of D.

For every input $x \in \{0,1\}^n$, we have $f_1(x) \le f_0(x)$ by (i), and $f_0(x) \le f_1(x) \vee D(x)$ by (ii). Finally, we also have that $|D| \le s^{r+1}$, because every node of T has fan-out at most s. □

Most important in the Switching Lemma is that the DNF D, correcting possible errors, contains only s^{r+1} monomials instead of all $\binom{n}{r+1}$ possible monomials.

4.2. Lower bounds criterion

We now give a general lower bounds criterium for monotone circuits.

DEFINITION 4.2. Let $f(x_1,\dots,x_n)$ be a monotone boolean function. We say that f is *t-simple* if for every pair of integers $1 \le r,s \le n-1$ there exists an exact $(s+1)$-CNF C, an exact $(r+1)$-DNF D, and a subset $I \subseteq \{1,\dots,n\}$ of $|I| \le s$ such that

(a) $|C| \le t \cdot r^{s+1}$ and $|D| \le t \cdot s^{r+1}$, and
(b) either $C \le f$ or $f \le D \vee \bigvee_{i\in I} x_i$ (or both) hold.

THEOREM 4.3 (Criterion). *If a monotone boolean function can be computed by a monotone circuit of size t, then it is t-simple.*

PROOF. Given a monotone circuit, the idea is to approximate every intermediate gate (more exactly – the function computed at the gate) by an s-CNF and an r-DNF, and to show that when doing so we do not introduce too many errors. If the function computed by the whole circuit is not t-simple, then it cannot be approximated well by such a CNF/DNF pair meaning that every such pair must make many errors. Since the number of errors introduced at each separate gate is small, the total number of gates must be large. To make as few errors at each gate as possible we will use the Switching Lemma (Lemma 4.1) which allows us to approximate an s-CNF by small r-DNFs and vice versa.

Let $F(x_1,\dots,x_n)$ be a monotone boolean function, and suppose that F can be computed by a monotone circuit of size t. Our goal is to show that then the function F is t-simple. To do this, fix an arbitrary pair of integers $1 \le s,r \le n-1$.

Let $f = g * h$ be a gate in our circuit. By an *approximator* of this gate we will mean a pair f_0, f_1, where f_0 is an s-CNF (a *left* approximator of f) and f_1 is an r-DNF (a *right* approximator of f) such that $f_1 \le f_0$.

We say that such an approximator f_0, f_1 of f introduces a new error on input $x \in \{0,1\}^n$ if the approximators of g and of h did not make an error on x, but the

approximator of f does. That is, $g_0(x) = g_1(x) = g(x)$ and $h_0(x) = h_1(x) = h(x)$, but either $f_0(x) \neq f(x)$ or $f_1(x) \neq f(x)$.

We define approximators inductively as follows.

Case 1: f is an input variable, say, $f = x_i$. In this case we take $f_0 = f_1 := x_i$. It is clear that this approximator introduces no errors.

Case 2: f is an And gate, $f = g \wedge h$. In this case we take $f_0 := g_0 \wedge h_0$ as the left approximator of f; hence, f_0 introduces no new errors. To define the right approximator of f we use Lemma 4.1 to convert f_0 into an r-DNF f_1; hence, $f_1 \leq f_0$. Let E_f be the set of inputs on which f_1 introduces a new error, i.e.,

$$E_f := \{x \mid f(x) = f_0(x) = 1 \text{ but } f_1(x) = 0\}.$$

By Lemma 4.1, all these errors can be "corrected" by adding a relatively small exact $(r+1)$-DNF: there is an exact $(r+1)$-DNF D such that $|D| \leq s^{r+1}$ and $D(x) = 1$ for all $x \in E_f$.

Case 3: f is an Or gate, $f = g \vee h$. This case is dual to Case 2. We take $f_1 := g_1 \vee h_1$ as the right approximator of f; hence, f_1 introduces no new errors. To define the left approximator of f we use Lemma 4.1 to convert f_1 into an s-CNF f_0; hence, $f_1 \leq f_0$. Let E_f be the set of inputs on which f_0 introduces a new error, i.e.,

$$E_f := \{x \mid f(x) = f_1(x) = 0 \text{ but } f_0(x) = 1\}.$$

By Lemma 4.1, all these errors can be "corrected" by adding a relatively small exact $(s+1)$-CNF: there is an exact $(s+1)$-CNF C such that $|C| \leq r^{s+1}$ and $C(x) = 0$ for all $x \in E_f$.

Proceeding in this way we will reach the last gate of our circuit computing the given function F. Let F_0, F_1 be its approximator, and let E be the set of all inputs $x \in \{0,1\}^n$ on which F differs from at least of one of the functions F_0 or F_1. Since at input gates (= variables) no error was made, for every such input $x \in E$, the corresponding error should be introduced at some intermediate gate. That is, for every $x \in E$ there is a gate f such that $x \in E_f$ (approximator of f introduces an error on x for the first time). But we have shown that, for each gate, all these errors can be corrected by adding an exact $(s+1)$-CNF of size at most r^{s+1} or an exact $(r+1)$-DNF of size at most s^{r+1}. Since we have only t gates, all such errors $x \in E$ can be corrected by adding an exact $(s+1)$-CNF C of size at most $t \cdot r^{s+1}$ and an exact $(r+1)$-DNF D of size at most $t \cdot s^{r+1}$, that is, for all inputs $x \in \{0,1\}^n$, we have

$$C(x) \wedge F_0(x) \leq F(x) \leq F_1(x) \vee D(x).$$

This already implies that the function F is t-simple. Indeed, if the CNF F_0 is empty (i.e., if $F_0 \equiv 1$) then $C \leq F$, and we are done. Otherwise, F_0 must contain some clause S of length at most s, say, $S = \bigvee_{i \in I} x_i$ for some I of size $|I| \leq s$. Since $F_0 \leq S$, the condition $F_1 \leq F_0$ implies $F \leq F_1 \vee D \leq F_0 \vee D \leq S \vee D$, as desired. This completes the proof of Theorem 4.3. □

4.3. Explicit lower bounds

In order to show that a given boolean function cannot be computed by a monotone circuit of size at most t, it is enough, by Theorem 4.3, to show that the function is not t-simple for at least one(!) choice of parameters s and r. In this section we demonstrate how this can be used to derive exponential lower bounds for concrete boolean functions.

In applications, boolean functions f are usually defined as set-theoretic predicates. In this case we say that f accepts a set $S \subseteq \{1,\dots,n\}$ if and only if f accepts its incidence vector.

A set S is a *positive input* for f if $f(S) = 1$, and a *negative input* if $f(\overline{S}) = 0$, where $\overline{S}$ is the complement of S. Put otherwise, a positive (negative) input is a set of variables which, if assigned the value 1 (0), forces the function to take the value 1 (0) regardless of the values assigned to the remaining variables. Note that one set S can be both positive and negative input! For example, if $f(x_1,x_2,x_3)$ outputs 1 iff $x_1+x_2+x_3 \geq 2$, then $S=\{1,2\}$ is both positive and negative input for f, because $f(1,1,x_3)=1$ and $f(0,0,x_3)=0$.

To translate the definition of t-simplicity of f (Definition 4.2) in terms of positive/negative inputs, note that if C is a CNF, then $C \leq f$ means that every negative input of f must contain at least one clause of C (looked at as set of indices of its variables). Similarly, $f \leq D \vee \bigvee_{i\in I} x_i$ means that every positive input must either intersect the set I or contain at least one monomial of D.

4.3.1. Detecting triangles. We begin with the simplest example. We will also present a more respectable applications—a $2^{\Omega(n^{1/4})}$ lower bound—but this special case already demonstrates the common way of reasoning pretty well.

Let us consider a monotone boolean function Δ_m, whose input is an undirected graph on m vertices, represented by $n=\binom{m}{2}$ variables, one for each possible edge. The value of the function is 1 if and only if the graph contains a triangle (three incident vertices). Clearly, there is a monotone circuit of size $O(m^3)$ computing this function: just test whether any of $\binom{m}{3}$ triangles is present in the graph. Thus, the following theorem is tight, up to a poly-logarithmic factor.

THEOREM 4.4. *Any monotone circuit, detecting whether a given m-vertex graph is triangle-free, must have $\Omega\left(m^3/\log^4 m\right)$ gates.*

PROOF. Let t be the minimal number for which Δ_m is t-simple. By Theorem 4.3, it is enough to show that $t \geq \Omega\left(m^3/\log^4 m\right)$. For this proof we take

$$s := \lfloor 5\log^2 m \rfloor \quad \text{and} \quad r := 1\,.$$

According to the definition of t-simplicity, we have only two possibilities.

Case 1: Every positive input for Δ_m either intersects a fixed set I of s edges, or contains at least one of $L \leq ts^{r+1} = ts^2$ 2-element sets of edges $R_1,\dots,R_L$.

As positive inputs for Δ_m we take all triangles, i.e., graphs on m vertices with exactly one triangle; we have $\binom{m}{3}$ such graphs. At most $s(m-2)$ of them will have an edge in I. Each of the remaining triangles must contain one of ts^2 given pairs of edges R_i. Since two edges can lie in at most one triangle, we conclude that, in this case,

$$t \geq \frac{\binom{m}{3} - s(m-2)}{s^2} = \Omega\left(m^3/\log^4 m\right)\,.$$

Case 2: Every negative input for Δ_m contains at least one of $tr^{s+1} = t$ sets of edges $S_1,\dots,S_t$, each of size $|S_i| = s+1$.

In this case we consider the graphs $E = E_1 \cup E_2$ consisting of two disjoint non-empty cliques E_1 and E_2 (we look at graphs as sets of their edges). Each such graph E is a negative input for Δ_m, because its complement is a bipartite graph, and hence, has no triangles. The number of such graphs is a half of the number 2^m of all binary strings of length m excluding $\mathbf{0}$ and $\mathbf{1}$. Hence, We have $2^{m-1}-1$ such graphs, and each

of them must contain at least one of the sets $S_1, \ldots, S_t$. Every of these sets of edges S_i is incident to at least $\sqrt{2s}$ vertices, and if $E \supseteq S_i$ then all these vertices must belong to one of the cliques E_1 or E_2. Thus, at most $2^{m-\sqrt{2s}} - 1$ of our negative inputs E can contain one fixed set S_i, implying that, in this case,

$$t \geq \frac{2^{m-1}-1}{2^{m-\sqrt{2s}}-1} \geq 2^{\sqrt{2s}-1} \geq 2^{3\log m} \geq m^3 .$$

Thus, in both cases, $t \geq \Omega\left(m^3/\log^4 m\right)$, and we are done. □

4.3.2. Graphs of polynomials. Our next example is the following monotone boolean function introduced by Andreev (1985). Let $q \geq 2$ be a prime power, and set $d := \lfloor (q/\ln q)^{1/2}/2 \rfloor$. Consider $q \times q$ $(0,1)$ matrices $A = (a_{i,j})$. Given such a matrix A, we are interested in whether it contains a graph of a polynomial $h : GF(q) \to GF(q)$, that is, whether $a_{i,h(i)} = 1$ for all rows $i \in GF(q)$.

Let f_n be a monotone boolean function in $n = q^2$ variables such that $f_n(A) = 1$ iff A contains a graph of at least one polynomial over $GF(q)$ of degree at most $d-1$. That is,

$$f_n(X) = \bigvee_h \bigwedge_{i \in GF(q)} x_{i,h(i)} ,$$

where h ranges over all polynomials over $GF(q)$ of degree at most $d-1$. Since we have at most q^d such polynomials, the function f_n can be computed by a monotone boolean circuit of size at most q^{d+1}, which is at most $n^{O(d)} = 2^{O(n^{1/4}\sqrt{\ln n})}$. We will now show that this trivial upper bound is almost optimal.

THEOREM 4.5. *Any monotone circuit computing the function f_n has size at least $2^{\Omega(n^{1/4}\sqrt{\ln n})}$.*

PROOF. Take a minimal t for which the function f_n is t-simple. Since $n = q^2$ and (by our choice) $d = \Theta(n^{1/4}\sqrt{\ln n})$, it is enough by Theorem 4.3 to show that $t \geq q^{\Omega(d)}$. For this proof we take

$$s := \lceil d \ln q \rceil \quad \text{and} \quad r := d ,$$

and look at input matrices as bipartite $q \times q$ graphs. In the proof we will essentially use the well-known fact that no two distinct polynomials of degree at most $d-1$ can coincide on d points. According to the definition of t-simplicity, we have only two possibilities.

Case 1: Every positive input for f_n either intersects a fixed set I of at most s edges, or contains at least one of $L \leq ts^{r+1}$ $(r+1)$-element sets of edges $R_1, \ldots, R_L$.

Graphs of polynomials of degree at most $d-1$ are positive inputs for f_n. Each set of l $(1 \leq l \leq d)$ edges is contained in either 0 or precisely q^{d-l} of such graphs. Hence, at most sq^{d-1} of these graphs can contain an edge in I, and at most $q^{d-(r+1)}$ of them can contain any of the given graphs R_i. Therefore, in this case we again have

$$t \geq \left(1 - \frac{s}{q}\right) \frac{q^d}{s^{r+1} \cdot q^{d-(r+1)}} \geq \left(\frac{q}{s}\right)^{\Omega(r)} \geq q^{\Omega(d)} .$$

Case 2: Every negative input for f_n contains at least one of $K \leq tr^{s+1}$ $(s+1)$-element sets of edges $S_1, \ldots, S_K$.

Let $\boldsymbol{E}$ be a random bipartite graph, with each edge appearing in $\boldsymbol{E}$ independently with probability $\gamma := (2d \ln q)/q$. Since there are only q^d polynomials of degree at most $d-1$, the probability that the complement of $\boldsymbol{E}$ will contain the graph of at least one

of them does not exceed $q^d(1-\gamma)^q \le q^{-d}$, by our choice of γ. Hence, with probability at least $1-q^{-d}$, the graph $\boldsymbol{E}$ is a negative input for f. On the other hand, each of the sets S_i is contained in $\boldsymbol{E}$ with probability $\gamma^{|S_i|} = \gamma^{s+1}$. Thus, in this case,

$$t \ge \frac{1-q^{-d}}{r^{s+1}\gamma^{s+1}} \ge \left(\frac{q}{2d^2\ln q}\right)^{\Omega(s)} \ge 2^{\Omega(s)} \ge q^{\Omega(d)},$$

where the third inequality holds for all $d \le (q/\ln q)^{1/2}/2$.

We have proved that the function f can be t-simple only if $t \ge q^{\Omega(d)}$. By Theorem 4.3, this function cannot be computed by monotone circuits of size smaller than $q^{\Omega(d)}$. □

4.4. Extension to circuits with real-valued gates

We now consider monotone circuits where, besides boolean AND and OR gates, one may use arbitrary monotone *real-valued* functions $\varphi : \mathbb{R}^2 \to \mathbb{R}$ as gates. Such a function φ is *monotone* if $\varphi(x_1,x_2) \le \varphi(y_1,y_2)$ whenever $x_1 \le y_1$ and $x_2 \le y_2$. The corresponding circuits are called *monotone real circuit*.

As in boolean circuits, inputs for such circuits also are binary strings $x \in \{0,1\}^n$; the output must be also a binary bit 0 or 1. But at each intermediate gate any monotone function $g : \{0,1\}^n \to \mathbb{R}$ may be computed. Hence, unlike in boolean case, here we have uncountable number of possible gates $\varphi : \mathbb{R}^2 \to \mathbb{R}$, and one may expect that at least some monotone boolean functions can be computed much more efficiently by such circuits. Exercise 4.6 at the end of this chapter shows that this intuition is correct: so-called "slice functions" can be computed by a very small monotone circuit with real-valued gates, but easy counting shows most of slice functions cannot be computed by boolean circuits of polynomial size, even if NOT gates are allowed!

It is therefore somewhat surprising that the criterion for boolean circuits (Theorem 4.3) remains true also for circuits with real-valued gates. The only difference from the boolean case is that now in the definition of t-simplicity we take slightly larger CNFs and DNFs, which does not greatly change the asymptotic values of resulting lower bounds.

We say that a monotone boolean function f is *weakly t-simple* if the conditions in Definition 4.2 hold with (a) replaced by

(a') $|C| \le t\cdot(2r)^{s+1}$ and $|D| \le t\cdot(2s)^{r+1}$

That is, the only difference from the definition of t-simplicity are a slightly larger upper bounds on the number of clauses in C and monomials in D.

THEOREM 4.6 (The Criterion for Real Circuits). *Let f be a monotone boolean function. If f can be computed by a monotone real circuit of size t then f is weakly t-simple.*

PROOF. The proof is similar to the boolean case (Theorem 4.6). We only have to show how to construct the approximators for real-valued gates. The idea is to consider thresholds of real gates and approximate the thresholded values. For a real-valued function $f : \{0,1\}^n \to \mathbb{R}$ and a real number a, let $f^{(a)}$ denote the boolean function that outputs 1 if $f(x) \ge a$, and outputs 0 otherwise.

Let now $\varphi : \mathbb{R}^2 \to \mathbb{R}$ be a gate at which the function $f(x)$ is computed, and let $g(x)$ and $h(x)$ are functions $g,h : \{0,1\}^n \to \mathbb{R}$ computed at the inputs of this gate. A simple (but crucial) observation is that then

$$\varphi\big(g(x),h(x)\big) \ge a \iff \exists b,c:\ g(x)\ge b,\ \ h(x)\ge c \ \text{ and } \ \varphi(b,c)\ge a\,.$$

The ($\Rightarrow$) direction is trivial: just take $b = g(x)$ and $c = h(x)$. The other direction ($\Leftarrow$) follows from the monotonicity of φ: $\varphi(g(x),h(x)) \geq \varphi(b,c) \geq a$.

Together with $f^{(a)}(x) = 1$ iff $\varphi(g(x),h(x)) \geq a$, this allows us to determine each threshold function $f^{(a)}$ of a gate $f = \varphi(g,h)$ from the thresholds of its input gates as:

$$f^{(a)} = \bigvee_{\varphi(b,c)\geq a} (g^{(b)} \wedge h^{(c)}) \tag{4.3}$$

as well as

$$f^{(a)} = \bigwedge_{\varphi(b,c)<a} (g^{(b)} \vee h^{(c)}). \tag{4.4}$$

It is convenient to think these expressions as an infinite AND and an infinite OR, respectively. However, since the number of settings $x \in \{0,1\}^n$ for input variables is finite, the real gates take only finite number of possible values, and therefore, we only need finite expressions.

As before, every threshold $f^{(a)}$ is approximated by two functions: an s-CNF $f_0^{(a)}$ and an r-DNF $f_1^{(a)}$. The approximators for the thresholds of the input variables are 0, 1, or the variable itself, depending on the value of the threshold; they can always represented by at most one literal and thus newer fail.

Let now $f = \varphi(g,h)$ be an intermediate gate with two input gates g and h, and suppose that, for all (finitely many!) reals b,c, the left and right approximators for threshold functions $g^{(b)}$ and $h^{(c)}$ of its input gates are already constructed.

To construct the *left* approximator from the approximators of its two input gates g and h, we first consider the representation

$$f^{(a)} = \bigvee_{\varphi(b,c)\geq a} (g_1^{(b)} \wedge h_1^{(c)}).$$

Since the monomials in the r-DNFs $g_1^{(b)}$ and $h_1^{(c)}$ have length at most r, all the subexpressions $g_1^{(b)} \wedge h_1^{(c)}$ can be turned into a single $2r$-DNF D_a such that

$$D_a(x) = 1 \quad \text{iff} \quad f^{(a)}(x) = 1 \quad \text{iff} \quad f(x) \geq a\,. \tag{4.5}$$

After that we use the same procedure as before (that is, Lemma 4.1) to convert this DNF into an s-CNF $f_0^{(a)}$. This can be done for each (of the finitely many) threshold values a, and we only need to ensure that the number of errors introduced when approximating the whole gate f does not depend on this number of thresholds.

When forming the s-CNF $f_0^{(a)}$, we introduce errors as we throw away clauses that become longer than s. We want to count the number of inputs $x \in \{0,1\}^n$ such that $f^{(a)}(x) = 0$ while $f_0^{(a)}(x) = 1$ for some a, i.e., the union over a of the errors introduced in a gate by $f_0^{(a)}$. To do this, let us list in the increasing order $a_1 < a_2 < \ldots < a_N$ all the $N \leq 2^n$ possible values $f(x)$ the gate f can output when the input vector x ranges over $\{0,1\}^n$. Hence,

$$D := D_{a_1} \vee D_{a_2} \vee \cdots \vee D_{a_N}$$

is a $2r$-DNF, and this DNF makes no error on x, i.e., $D(x) = f(x)$. By (4.5), we have that

$$D_{a_1} \geq D_{a_2} \geq \cdots \geq D_{a_N}\,.$$

That is, every monomial of $D_{a_{i+1}}$ contains at least one monomial of D_{a_i}. Hence, if $t(D)$ denotes the family of all transversals of D, that is, the family of all subsets of variables,

each of which intersects all the monomials of D, then

$$t(D_{a_1}) \subseteq t(D_{a_2}) \subseteq \cdots \subseteq t(D_{a_N}),$$

implying that $t(D) = t(D_{a_N})$. This means that all the clauses (=transversals), which we throw away (because they are longer than s) when forming an s-CNF f_0 from the DNF D, are precisely those clauses, which we would throw away when converting the $2r$-DNF D_{a_N} into an s-CNF. Thus, by Lemma 4.1, all the errors that may appear during the construction of the left approximator f_0, can be corrected by an exact $(s+1)$-CNF C of size $|C| \le (2r)^{s+1}$. That is, for every input x such that $f(x) = 0$ but $f_0(x) = 1$, we have that $C(x) = 0$.

A dual argument can be used to bound the number of errors introduced when constructing the right approximator f_1. Note that we cannot use the DNF (4.5) for this purpose since D is a $2r$-DNF, not an r-DNF. But we can argue as above by using the expression (4.4) instead of (4.3). Then all the introduced errors can be corrected by an exact $(r+1)$-DNF D of size $|D| \le (2s)^{r+1}$. The rest of the proof is the same as that of Theorem 4.3. □

Since the definitions of t-simple functions and of weakly t-simple function are almost the same, Theorem 4.6 allows to extend lower bounds for the monotone *boolean* circuits (we proved above) to the monotone *real* circuits. For example, the same argument as in the proof of Theorem 4.5 yields

THEOREM 4.7. *Any monotone real circuit computing the polynomial function f_n has size at least $2^{\Omega(n^{1/4}\sqrt{\ln n})}$.*

REMARK 4.8. Lower bounds for monotone real circuits have found intriguing applications in proof complexity. In particular, Pudlák (1997) used such bounds to prove the first exponential lower bound on the length of so-called "cutting plane proofs," a proof system for solving integer programming problems. We will describe this result in Section 18.5.

The extension of the lower bounds criterium from monotone boolean circuits to monotone real circuits shows the power of the criterion. On the other hand, it gives us an explanation of why the method requires monotonicity.

PROPOSITION 4.9. *Any boolean function in n variables can be computed using $n-1$ real monotone fanin-2 gates and one non-monotone unary gate.*

PROOF. For an input vector $x \in \{0,1\}^n$, let $\mathrm{bin}(x) = \sum_{i=1}^{k} x_i 2^{i-1}$ be the number whose binary code is x. It is easy to see that $\mathrm{bin}(x)$ can be computed by a circuit $C(x)$ using $n-1$ real fanin-2 gates of the form $g(u,v) = u + 2v$. This can be done via the recurrence:

$$\mathrm{bin}(x) = x_1 + 2 \cdot \mathrm{bin}(x') = g(x_1, \mathrm{bin}(x')),$$

where $x' = (x_2, \ldots, x_n)$. These gates are monotone.

Now, every boolean function f defines a unique set of numbers

$$L_f = \{\mathrm{bin}(x) \mid f(x) = 1\}.$$

Hence, in order to compute f, it is enough to attach the (non-monotone) output gate testing whether $C(x) \in L_f$ or not. □

4.5. Criterion for graph properties

When dealing with decision problems defined on graphs, variables x_e usually correspond to edges, not to vertices. Each set S of edges defines a graph, and graphs (unlike just unstructured sets) have many interesting parameters. It is therefore many possibilities to define the "length" $\mu(C)$ of a clause $C = \bigvee_{e\in S} x_e$:

- as the number $|S|$ of edges in S;
- as the number $|V(S)|$ of vertices touched by the edges in S;
- as the number $\kappa(S)$ of connected components in the graph S,

and so on. Depending on what concrete graph property we are dealing with, one measure may be more suitable than another. It makes therefore sense to extend the lower bounds criterion so that it allows different measures of "length."

To do this we only need one additional concept. Namely, say that an input vector x *respects* a length measure μ if we cannot add an edge from outside the set $S = \{e \mid x_e = 1\}$ to any of its subsets without increasing the weight of the μ-length of S, that is, if

$$\mu(S' \cup \{e\}) \geq \mu(S) + 1 \text{ for any subset } S' \subseteq S \text{ and any } e \notin S.$$

We additionally require that $\mu(S)/2 \leq |S| \leq \mu(S)^2$. In particular, if $\mu(S) := |V(S)|$ then these inequalities hold, but in this case only cliques (complete graphs) will respect this measure.

Let ν and μ be any length measures. We only require that both they are at most $\nu(S)$, the number of vertices touched by the edges in S. The following lemma and its proof look more "complicated" than our initial Switching Lemma (Lemma 4.1), but this is only "notational complexity" (we work in more general measure spaces)—the proof idea is the same as in the case of trivial measures $\mu(S) = \nu(S) := |S|$.

LEMMA 4.10. *Let f_0 be a CNF whose clauses have ν-length at most s. Then there exist DNFs f_1 and D such that: D contains at most s^{4r} monomials, all monomials in f_1 have μ-length $< r$, all monomials in D have μ-length $\geq r$, and for all inputs $x \in \{0,1\}^n$ respecting the norm μ, we have that*

$$f_1(x) \leq f_0(x) \leq f_1(x) \vee D(x). \tag{4.6}$$

PROOF. Let $f_0 = C_1 \wedge \cdots \wedge C_m$. We identify each clause $C = \bigvee_{e\in S} x_e$ with the set S of edges indexing its variables. Hence, each clause C_i has $|C_i| \leq s^2$ variables. Arguing as in the proof of Lemma 4.1 we can associate with f_0 a tree T of fan-out at most s^2 by slightly modifying the condition of when we declare a node to be a leaf. Namely, we now say that a monomial M *pierces* a clause C_i if $\mu(M \cup \{e\}) > \mu(M)$ for all edges $e \in C_i$. (This in particular implies that $M \cap C_i = \emptyset$, but the converse needs not hold if μ is a non-trivial length measure, not $\mu(S) = |S|$.)

As in the proof of Lemma 4.1, the first node of T corresponds to the first clause C_1, and the outgoing $|C_1|$ edges are labeled by the variables from C_1. Suppose we have reached a node v, and let M be the monomial consisting of the labels of edges from the root to v. If M pierces all the clauses of f_0, then v is a leaf. Otherwise, let C_i be the *first* clause not pierced by M. Then the node v has $|C_i|$ outgoing edges labeled by the variables in C_i.

We now let f_1 consist of all paths (monomials) ending in a leaf of T and whose μ-length is smaller than r. Let also D contain all paths M to nodes v such that the path to its father has μ-length $< r$, but the path M to the node v itself has μ-length

$\geq r$. That is, D consists of all paths whose μ-length reached the threshold r for the first time. In particular, the μ-length of each monomial in D is at least r.

First observe that each monomial M of D has length (not just μ-length!) $|M| \leq 2r$. Indeed, when going from the father to its child in T, the length of a path increases by exactly 1, whereas the μ-length increases by at least 1 and by at most 2 (only one edge is added). Hence, monomials in D correspond to paths in T of length at most $2r$. Since every node of T has fan-out at most s^2, this gives the desired upper bound $|D| \leq (s^2)^{2r} = s^{4r}$ on the total number of monomials in D.

It remains to prove (4.6). Since, by the construction, each path of T is either a monomial of f_1 or is an extension of at least one monomial in D, we immediately have that $f_0(x) \leq f_1(x) \vee D(x)$ holds for all inputs $x \in \{0,1\}^n$.

However, this time the inequality $f_1(x) \leq f_0(x)$ needs not to hold for all inputs x. The reason, why this could happen, is that the fact that a monomial M pierces all clauses of f_0 alone does not imply that M must also *intersect* all the clauses of f_0: in general, $\mu(M \cup \{e\}) = \mu(M)$ does not necessarily imply that $e \in M$. Note however, that this cannot happen if the set $S_x = \{e \mid x_e = 1\}$ respects the norm μ: this would mean that we can add to M an edge e lying outside S_x (and hence, outside M) without increasing the μ-length of M. □

This lemma motivates the following modification of the notion of t-simplicity for graph properties. Let ν be some length measure for negative inputs, and μ be some length measure for positive inputs.

DEFINITION 4.11. A graph property f is *weakly t-simple* with respect to (ν, μ), if for all integers $1 \leq r, s \leq n-1$ there exists a set I of edges of ν-length at most s, a system of sets $S_1, \ldots, S_K$ of edges each of ν-length at least s, and a system of sets $R_1, \ldots, R_L$ of edges each of μ-length at least r such that[1] $K \leq t(2r)^{4s}$, $L \leq t(2s)^{4r}$ and at least one of the following two conditions hold:

a. Every positive input of μ-length at least r, which respects the norm μ, either intersects the set I or contains at least one of the sets $R_1, \ldots, R_L$.
b. Every negative input of ν-length at least s, which respects the norm ν, contains at least one of the sets $S_1, \ldots, S_K$.

We have the following extension of Theorem 4.6 to more general length measures for inputs.

THEOREM 4.12. *If a monotone boolean function can be computed by a monotone real circuit of size t, then it is t-simple with respect to any pair ν, μ of length measures.*

The proof of this theorem is the same as that of Theorem 4.6: just use Lemma 4.10 (and its dual version) instead of Lemma 4.1. We leave a detailed proof as an exercise.

4.6. Clique-like problems

We consider graphs on a fixed set of m vertices. We have $n = \binom{m}{2}$ boolean variables, one for each potential edge. Then each boolean function $f : \{0,1\}^n \to \{0,1\}$ described some graph property. A prominent **NP**-complete graph property if a monotone boolean function $CLIQUE(m,k)$ which accepts a given graph iff it contains a k-clique, that is, a a subgraph on k vertices whose all vertices are pairwise adjacent.

[1] We take $(2r)^{4r}$ instead of just r^{4s}, as suggested by Lemma 4.10, in order to cover also the real-valued case.

Instead of proving a lower bound on this function we will do this for a much larger class of "clique-like" functions.

A complete k-partite graph is a disjoint union of k cliques, some of which may consist of just one isolated vertex. Each such graph is defined by a coloring of vertices in k colors, and two vertices are adjacent iff they recieve different colors. Note that none of such graphs can contain a k-clique, but adding any one edge already yields a k-clique.

Let $2 \le a \le b \le m$ be integers. An (a,b)-*clique function* is a monotone boolean function f such that, for every graph G on m vertices,

$$f(G) = \begin{cases} 0 & \text{if } G \text{ is complete } (a-1)\text{-partite graph;} \\ 1 & \text{if } G \text{ is a } b\text{-clique;} \\ \text{any value} & \text{otherwise.} \end{cases}$$

THEOREM 4.13. *Let $32 \le a \le b \le m/32$, and let f be an (a,b)-clique function. Then the minimal number of gates in a monotone real circuit computing f is exponential in* $\min\{a, m/b\}^{1/4}$.

In particular, for $k = \lfloor \sqrt{m} \rfloor$, $CLIQUE(m,k)$ requires $2^{\Omega(m^{1/8})}$ gates.

PROOF. Let f be an (a,b)-clique function. We are going to apply the refined version of the lower bounds criterion (Theorem 4.6). To do this, we must first choose appropriate length measure μ for positive inputs an a length measure ν for negative inputs.

What to take as positive and how to measure their length is clear. All b-cliques are positive inputs for f. A natural measure for a clique S is to take

$$\mu(S) := \text{ the number of vertices touched by the edges in } S\,.$$

It is clear that every clique S respects this measure: we cannot add a new edge e to S without increasing the number of vertices.

What to take as negative inputs is also clear: such are all *complements* of complete $(a-1)$-partite graphs. Each such complement G_h is defined by a coloring h of vertices in $a-1$ colors: two vertices u and v are adjacent in G_h iff $h(u) = h(v)$. But what should we take as a length $\nu(G_h)$ of such graphs? We cannot take the same length measure $\mu(G_h)$ (as for positive inputs) because the graphs G_h do not respect this measure: if $S \subseteq G_h$ and if the ends of the edge e belong to different parts of G_h, then $\mu(S \cup \{e\}) = \mu(S)$. Observe however that, in this case, the graph $S \cup \{e\}$ has one connected component fewer! This suggests the following length measure for negative inputs: take

$$\nu(S) := \mu(S) - \kappa(S)\,,$$

where $\kappa(S)$ is the number of connected components in S. We claim that this measure is already respected by all graphs $G_h = (V,E)$. To show this, let $S \subseteq E$ and $e \notin E$. Let also $V(S)$ be the set of all vertices touched by the edges of S; hence, $|V(S)| = \mu(S)$. Since each connected component of E is either an isolated vertex or a clique, the edge e must lie between two distinct components of E. Since each connected component of S lies entirely in some of the components of E, we have that

a. either $e \subseteq V(S)$, and hence, $\mu(S \cup \{e\}) = \mu(S)$ and $\kappa(S \cup \{e\}) = \kappa(S) - 1$;
b. or $|e \cap V(S)| = 1$, and hence, $\mu(S \cup \{e\}) = \mu(S) + 1$ and $\kappa(S \cup \{e\}) = \kappa(S)$;
c. or $e \cap V(S) = \emptyset$, and hence, $\mu(S \cup \{e\}) = \mu(S) + 2$ and $\kappa(S \cup \{e\}) = \kappa(S) + 1$.

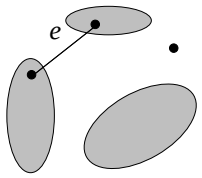


FIGURE 1. A complement of a complete 4-partite graph. Any new edge e can only join its distinct connected components.

In all three case we have that

$$\nu(S\cup\{e\}) = \mu(S\cup\{e\}) - \kappa(S\cup\{e\}) = \mu(S) - \kappa(S) + 1 = \nu(S) + 1\,,$$

as desired.

Suppose now that the function f is t-simple. By Theorem 4.12, it is enough to show that then $t \geq 2^{\Omega(N^{1/4})}$. Set

$$r := \left\lfloor \left(\frac{a-1}{32}\right)^{1/4} \right\rfloor \quad \text{and} \quad s := \left\lfloor \left(\frac{m}{32b}\right)^{1/4} \right\rfloor .$$

According to Definition 4.11 we have only two possibilities, depending on what of the two of its items holds.

Case 1: Every positive input of μ-length at least r, which respects the norm μ, either intersects the set I or contains at least one of the sets $R_1,\ldots,R_L$, each of μ-length at least r.

Positive inputs are b-cliques. At least $\binom{m}{b} - s^2\binom{m-2}{b-2} \geq \frac{1}{2}\binom{m}{b}$ of such cliques must avoid a fixed set I of $|I| \leq s^2$ edges. Each of these b-cliques must contain at least one of $L \leq t\cdot(2s)^{4r}$ r-cliques $R_1,\ldots,R_L$. Since each R_i is contained in $\binom{m-r}{b-r}$ of k-cliques, we conclude that in this case

$$t \geq \frac{\frac{1}{2}\binom{m}{b}}{(2s)^{4r}\binom{m-r}{b-r}} \geq \left(\frac{m}{16s^4 b}\right)^{\Omega(r)} = 2^{\Omega(a^{1/4})}\,.$$

Case 2: Every negative input of ν-length at least s, which respects the norm ν, contains at least one of the sets $S_1,\ldots,S_K$, each of ν-length at least s.

Each negative input G_h consists of m_1 isolated vertices and m_2 mutually disjoint cliques, where $1 \leq m_1 + m_2 \leq a-1$. Thus,

$$\nu(G_h) = \mu(G_h) - \kappa(G_h) = (m - m_1) - m_2 \geq m - a + 1 > s\,.$$

Since the graphs G_h respect the norm ν, we have that each of these graphs must contain at least one of the sets of edges $S_1,\ldots,S_K$, where $\nu(S_i)\geq s$ and $K \leq t\cdot(2r)^{4s}$. We have $(a-1)^m$ colorings h, and it remains to estimate for how many of them, the induced graph G_h can contain a fixed set of edges S, with $\nu(S)\geq s$.

If $V_1,\ldots,V_d$ are the sets of vertices of the connected components of S, then by the definition of the norm ν, $|V_1|+\ldots+|V_d| \geq s+d$. If $G_h \supseteq S$, then all the vertices in each of the classes V_i must get the same color. Hence, the number of colorings h, for which $G_h \supseteq S$, does not exceed

$$(a-1)^d (a-1)^{m-(s+d)} = (a-1)^{m-s}\,.$$

Thus, in this case,

$$t \geq \frac{(a-1)^m}{(2r)^{4s}(a-1)^{m-s}} = \left(\frac{a-1}{16r^4}\right)^s = 2^{(m/b)^{1/4}}. \qquad \square$$

As mentioned above, the class of clique-like functions includes some **NP**-complete problems, like $CLIQUE(m,k)$. But the class of (a,b)-clique functions is much larger. So large that it also includes some graph properties computable by non-monotone circuits of polynomial size!

A *graph function* is a function φ assigning each graph G a real number $\varphi(G)$. Such a function φ is *clique-like* if

$$\omega(G) \leq \varphi(G) \leq \chi(G),$$

where $\omega(G)$ is the clique number, i.e. the maximum number of vertices in a complete subgraph of G, and $\chi(G)$ is the chromatic number, i.e. the smallest number of colors which is enough to color the vertices of G so that no adjacent vertices receive the same color.

Fix k to be the square root of the number m of vertices, and let f_φ denote the monotone boolean function of $n = \binom{m}{2}$ boolean variables encoding the edges of a graph on m vertices, whose values are defined by

$$f_\varphi(G) = 1 \quad \text{iff} \quad \varphi(G) \geq k.$$

Note that

$$f_\varphi(G) = \begin{cases} 1 & \text{if } \omega(G) \geq k, \\ 0 & \text{if } \chi(G) \leq k-1. \end{cases}$$

OBSERVATION 4.14. For every clique-like graph function φ, the boolean function f_φ is a (k,k)-clique function.

PROOF. If G contains a k-clique, then $\varphi(G) \geq \omega(G) = k$, and hence, the function f_φ accepts G. On the other hand, if G is a complete $(k-1)$-partite graph, then $\varphi(G) \leq \chi(G) \leq k-1$, and f_φ rejects G. $\square$

Although we always have that $\omega(G) \leq \chi(G)$, the gap between these two quantities can be quite large: results of Erdös (1967) imply that the maximum of $\chi(G)/\omega(G)$ over all m-vertex graphs G has the order $\Theta\left(m/\log^2 m\right)$. So, at least potentially, the class of clique-like functions is large enough. And indeed, Tardos (1987) observed that this class includes not only **NP**-complete problems (like the clique function) but also some problems from **P**.

LEMMA 4.15. *There exists an explicit monotone clique-like graph function φ which is computable in polynomial time.*

PROOF. In his seminal paper on Shannon-capacity of graphs Lovász (1979a) introduced the capacity $\vartheta(G)$. The function $\varphi'(G) := \vartheta(\overline{G})$, where $\overline{G}$ denotes the complement of G, is a monotone clique-like function. Grötschel, Lovász and Schrijver (1981) gave a polynomial time approximation algorithm for ϑ. That is, given a graph G and a rational number $\varepsilon > 0$ the algorithm computes, in polynomial time, a function $g(G,\varepsilon)$ such that

$$\vartheta(G) \leq g(G,\varepsilon) \leq \vartheta(G) + \varepsilon.$$

Now, for any $0 < \varepsilon < 1/2$ the function $\lfloor g(\overline{G}, \varepsilon) \rfloor$ is a polynomial time computable clique-like function. This function might not be monotone. Let us therefore consider the monotone function

$$\varphi(G) = \lfloor g(\overline{G}, n^{-2}) + e(G) \cdot n^{-2} \rfloor,$$

where n is the number of vertices and $e(G)$ the number of edges in G. This is the desired monotone clique-like function computable in polynomial time. □

Together with Theorem 4.13, Observation 4.14 and Lemma 4.15 immediately yield an exponential trade-off between monotone and non-monotone circuits.

THEOREM 4.16. *For every clique-like graph function φ, the boolean function f_φ can be computed by a non-monotone boolean circuit of polynomial size, but any monotone real circuit requires $2^{\Omega(n^{1/16})}$ gates.*

We will use this theorem later in Section 18.5 to prove exponential lower bounds for widely used proof systems—resolution refutation and cutting plane proofs.

4.7. What about circuits with NOT gates?

As we mentioned at the very beginning, no non-linear lower bounds are known for circuits using NOT gates. So, what is "bad" with the arguments we described in this and the previous chapters? Why they do not work for non-monotone circuits?

A possible answer is that the arguments are just too general! In order to show that no circuit with t gates can compute a given boolean function f, we have to show that no such circuit C can separate the set $f^{-1}(0)$ from $f^{-1}(1)$, that is, reject *all* vectors in $f^{-1}(0)$ and accept *all* vectors in $f^{-1}(1)$. Current arguments for monotone circuits (and formulas) do much more: there are relatively small subsets $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$ (sets of particular negative and positive inputs) such that every monotone circuit separating A from B must be large.

To be more specific, let A be the set of all complete $(k-1)$-partite graphs on m vertices, and B be the set of all k-cliques. Hence, for *any* k-clique function f, members of A are negative inputs and members of B are positive inputs for f. We have shown that any monotone circuit separating A from B must have exponential size.

On the other hand, A *can* be separated from B by a small circuit if we allow just one NOT gate be used at the top of the circuit! Indeed, each graph in A has at least $K = \binom{m/k}{2}$ edges, whereas each graph in B (a k-clique) has only $\binom{k}{2}$ edges, which is smaller than K for for every $k < \sqrt{m}$. Hence, if $g = \neg Th_K$ is the negation of the threshold-K function, then $g(a) = 0$ for all $a \in A$, and $g(b) = 1$ for all $b \in B$. Since threshold functions have small monotone circuits (at most n^2 in the number n of input variables), the resulting circuit is also small, separates A from B, and has only one NOT gate.

That is, it is not hard to separate the pair A, B by a monotone circuit – it is only hard to do this separation in a "right" direction: reject all vectors $a \in A$, and accept all vectors $b \in B$. This motivates the following definition.

Let f be a monotone boolean function. Say that a pair A, B with $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$ is *r-hard* if every monotone circuit separating A and B (either in a "right" or in a "wrong" direction) must have super-polynomial size.

Exercise 4.7 shows that any r-hard pair A, B requires a super-polynomial number of gates in any circuit that separates A from B and uses up to r NOT gates. In the next chapter we will show that $r = \lceil \log(n+1) \rceil$ is a critical number of allowed NOT gates:

having an r-hard pair for such an r we would have proved a super-polynomial lower bound for non-monotone circuits! What we know so far is that the clique function produces an r-hard pair for r about $\log\log n$; this was shown by Amano and Maruoka (2005).

RESEARCH PROBLEM 4.17. *Exhibit an explicit r-hard pair A, B for $r \gg \log\log n$.*

Exercises

EX. 4.1. A *partial b–(n,k,λ) design* is a family $\mathscr{F}$ of k-element subsets of $\{1,\ldots,n\}$ such that any b-element set is contained in at most λ of its members. We can associate with each such design $\mathscr{F}$ a monotone boolean function $f_{\mathscr{F}}$ such that $f_{\mathscr{F}}(S)=1$ if and only if $S \supseteq F$ for at least one $F \in \mathscr{F}$. Assume that $\ln|\mathscr{F}| < k-1$ and that each element belongs to at most N members of $\mathscr{F}$. Use Theorem 4.3 to show that for every integer $a \geq 2$, every monotone circuit computing $f_{\mathscr{F}}$ has size at least

$$\ell := \min\left\{\frac{1}{2}\left(\frac{k}{2b\ln|\mathscr{F}|}\right)^a,\ \ \frac{|\mathscr{F}|-a\cdot N}{\lambda\cdot a^b}\right\}.$$

Hint: Take $r=a-1$, $s=b-1$ and show that under this choice of parameters, the function $f_{\mathscr{F}}$ can be t-simple only if $t \geq \ell$. When doing this, note that the members of $\mathscr{F}$ are positive inputs for $f_{\mathscr{F}}$. To handle the case of negative inputs, take a random subset T in which each element appears independently with probability $p=(1+\ln|\mathscr{F}|)/k$, and show that T is not a negative input for $f_{\mathscr{F}}$ with probability at most $|\mathscr{F}|(1-p)^k \leq e^{-1}$.

EX. 4.2. Derive Theorem 4.5 from the previous exercise.

Hint: Observe that the family of all q^d graphs of polynomials of degree at most $d-1$ over $GF(q)$ forms a partial b–(n,k,λ) design with parameters $n=q^2$, $k=q$ and $\lambda=q^{d-b}$.

EX. 4.3. Andreev (1987) has shown how, for any prime power $q \geq 2$ and $d \leq q$, to construct an explicit family $\mathscr{F}$ of subsets of $\{1,\ldots,n\}$ which, for every $b \leq d+1$, forms a partial b–(n,k,λ) design with parameters $n=q^3$, $k=q^2$, $\lambda=q^{2d+1-b}$ and $|\mathscr{F}|=q^{2d+1}$. Use Exercise 4.1 to show that the corresponding boolean function $f_{\mathscr{D}}$ requires monotone circuits of size exponential in $\Omega\left(n^{1/3-o(1)}\right)$.

EX. 4.4. A boolean function $f(x_1,\ldots,x_n)$ is a *k-slice function* if $f(x)=0$ for all x with $|x|<k$, and $f(x)=1$ for all x with $|x|>k$. Show that some slice functions require DeMorgan circuits of size $2^{\Omega(n)}$.

Hint: Take $k=n/2$ and argue as in the proof of Theorem 1.2.

EX. 4.5. Given a vector $x=(x_1,\ldots,x_n)$ in $\{0,1\}^n$, associate with it the following two integers $h_+(x) := |x|\cdot 2^n + b(x)$ and $h_-(x) := |x|\cdot 2^n - b(x)$, where $|x| = x_1+\cdots+x_n$ and $b(x)=\sum_{i=1}^n x_i 2^{i-1}$. Prove that for any two vectors $x \neq y$,
a. if $|x|<|y|$, then $h_+(x)<h_+(y)$ and $h_-(x)<h_-(y)$;
b. if $|x|=|y|$, then $h_+(x)\leq h_+(y)$ if and only if $h_-(x)\geq h_-(y)$.

EX. 4.6. Let $f(x_1,\ldots,x_n)$ be a k-slice function, $0\leq k\leq n$. Use the previous exercise to show that f can be computed by a circuit with $O(n)$ monotone real-valued functions as gates. *Hint*: As the last gate take a monotone function $\varphi:\mathbb{R}^2\to\{0,1\}$ such that $\varphi(h_+(x),h_-(x))=f(x)$ for all inputs x of weight $|x|=k$.

EX. 4.7. Let f be a boolean function and suppose that it can be computed by a circuit of size t with at most r negations. Show that then, for any $A\subseteq f^{-1}(0)$ and $B\subseteq f^{-1}(1)$, there is a monotone boolean function g such that g can be computed by a

monotone circuit of size at most t and either g or its negation $\neg g$ rejects a 2^{-r} fraction of inputs from A and accepts a 2^{-r} fraction of inputs from B.

Hint: Argue by induction on r. If $r \geq 1$, then consider the first negation gate and the function g that is computed at the gate immediately before this negation gate. Let $\varepsilon \in \{0,1\}$ be such that $g(a) = \varepsilon$ for at least one half of the inputs $a \in A$. If also one half of the inputs $b \in B$ have $g(b) = \varepsilon \oplus 1$, then either g or $\neg g$ has the property stated in the lemma. If this is not the case, try to apply the induction hypothesis.

Bibliographic Notes

Until 1985, the largest known lower bound on the size of monotone circuits for an explicit boolean function of n variables was only $4n$. The breakthrough was made by Razborov (1985a,1985b) who proved a super-polynomial lower bound of $n^{\Omega(\log n)}$ for $CLIQUE_{n,k}$ with $k = \Theta(\log n)$. Shortly afterwards, Andreev (1985) and Alon and Boppana (1987) used similar methods to obtain exponential lower bounds. A simpler and more symmetric lower bounds argument, presented above, emerged implicitly in works of Haken (1995), and Haken and Cook (1999), and was made explicit in [**79**]. The Switching Lemma (Lemma 4.1) was first proved in terms of finite limits in [**79**]; the decision-tree version presented above is due to Berg and Ulfberg (1999). Harnik and Raz (2000) used this lemma to prove the numerically highest known lower bound $2^{\Omega(n^{1/3})}$ for a monotone boolean function n variables. The proof idea of Theorem 4.6 is due to Avi Wigderson. That slice functions have monotone real circuits of linear size (Exercise 4.6) was first observed by Rosenbloom (1997). Lemma 4.15 is due to Éva Tardos (1987).

CHAPTER 5

Mystery of Negations

The main difficulty in proving non-trivial lower bounds on the size of circuits over $\{\wedge, \vee, \neg\}$ is the presence of NOT gates – we already know how to prove even exponential lower bounds if no NOT gates are allowed. The effect of such gates on circuit size remains to a large extent a mystery. It is, therefore, worth to recall what we actually know about this "ghost." Among the basic questions concerning the role of NOT gates are the following:

1. For what monotone boolean functions NOT gates are useless, that is, cannot lead to much more efficient circuits?

2. Given a function f, what is the minimum number $M(f)$ of NOT gates in a circuit computing f?

3. Given a circuit, to what extend can we decrease the number of NOT gates in it without a substantial increase in circuits size? In particular, how much can the size of a circuit increase when trying to compute f using the smallest number $M(f)$ of NOT gates?

4. Suppose that a function f in n variables *can* be computed by a circuit of size polynomial in n, but every circuit with $M(f)$ negations computing f requires superpolynomial size. What is then the minimal number of negations sufficient to compute f in polynomial size? In other words, how many NOT gates do we need in oder to achieve superpolynomial savings in circuit size?

In this chapter we answer these questions.

5.1. When NOT gates are useless?

Let us consider circuits with gates $\wedge, \vee, \neg$. Recall that such a circuit is a DeMorgan circuit if its inputs are variables and their negations, and gates are fanin-2 AND and OR gates. A circuit is monotone if it has no negated inputs.

For a boolean function f, let $C(f)$ denote the smallest number of gates in a circuit over $\{\wedge, \vee, \neg\}$ computing f. Let also $C'(f)$ denote the number of $\wedge$ and $\vee$ gates in a DeMorgan circuit computing f. It can be shown (do this!) that $C'(f) \leq 2 \cdot C(f)$. For a monotone boolean function f, let $C_+(f)$ denote the smallest number of gates in a monotone DeMorgan circuit computing f.

As we already mentioned above, current methods are not able to prove larger than $C(f) \geq 5n$ lower bounds, where n is the number of variables. On the other hand, we already know how to prove even exponential lower bounds for monotone circuits where we have no NOT gates at all. Even better, there is a large class of monotone boolean functions f for which NOT gates are almost useless, that is, $C_+(f)$ is not much larger than $C(f)$. These are so-called "slice functions". Unfortunately, known lower bounds arguments for monotone circuits do not work for these functions.

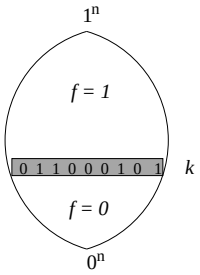


FIGURE 1. A k-slice function.

5.1.1. Slice functions. A boolean function $f(\boldsymbol{x})$ is a *k-slice function* if $f(\boldsymbol{x}) = 0$ when $|\boldsymbol{x}| < k$, and $f(\boldsymbol{x}) = 1$ when $|\boldsymbol{x}| > k$; here $|\boldsymbol{x}| = x_1 + \ldots + x_n$ is the number of 1's in $\boldsymbol{x}$. Note that slice functions are monotone! They are, however, nontrivial only on the k-th slice of the binary n-cube $\{0,1\}^n$. Note also that, for *every* boolean function f, the function $f^{(k)}$ defined by

$$f^{(k)} = f \wedge Th_k^n \wedge \neg Th_{k+1}^n \vee Th_{k+1}^n$$

is a k-slice function. Here, as before, Th_k^n is the threshold-k function in n variables which accepts a given vector iff it has at least k 1's.

Important property of slice functions is that NOT gates are almost useless when computing them. This is because we can replace each negated input in a circuit for a k-slice function f by a small monotone circuit computing a threshold function. The idea, due to Berkowitz is to consider threshold functions

$$T_{k,i}^n(x_1,\ldots,x_n) := Th_k^{n-1}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)\,.$$

A simple (but crucial) observation is that, for all input vectors $\boldsymbol{x} \in \{0,1\}^n$ with exactly k ones, $T_{k,i}(\boldsymbol{x})$ is the negation of the ith bit x_i:

$$T_{k,i}(\boldsymbol{x}) = \neg x_i\,. \tag{5.1}$$

It is known that all these n threshold functions ($i = 1,\ldots,n$) can be computed by a monotone circuit of size $O(n\log^2 n)$. Hence, if we replace all n negated inputs in a (non-monotone) circuit

$$f(x_1,\ldots,x_n) = F(x_1,\ldots,x_n,\neg x_1,\ldots,\neg x_n)$$

for a k-slice function f by outputs of this circuit, we obtain a *monotone* circuit

$$F_+(x_1,\ldots,x_n) = F(x_1,\ldots,x_n,T_{k,1}(\boldsymbol{x}),\ldots,T_{k,n}(\boldsymbol{x}))\,.$$

It is not difficult to verify that F_+ also computes f. That $F_+(\boldsymbol{x}) = F(\boldsymbol{x})$ for all inputs $\boldsymbol{x}$ with $|\boldsymbol{x}| = k$ follows from (5.1). To show that the same holds for all remaining input vectors, observe that

$$F(x_1,\ldots,x_n,0,\ldots,0) \le F(x_1,\ldots,x_n,\neg x_1,\ldots,\neg x_n) \le F(x_1,\ldots,x_n,1,\ldots,1)\,.$$

This holds because the circuit F itself is monotone, i.e., has only AND and OR gates (negations are only on inputs). Thus, if $|\boldsymbol{x}| < k$, then $f(\boldsymbol{x}) = 0$ *independent* of whether $x_i = 0$ or $x_i = 1$. Hence, on such input vectors,

$$F_+(x_1,\ldots,x_n) = F(x_1,\ldots,x_n,0,\ldots,0) \le f(x_1,\ldots,x_n) = 0\,.$$

The case of input vectors with more than k ones is dual.

What we have just proved is the following

THEOREM 5.1. *For any slice function f in n variables,*

$$C'(f) \geq C_+(f) - O(n \log^2 n).$$

Thus, any lower bound $C_+(f) \gg n \log^2 n$ on the monotone(!) complexity of a slice function would yield superlinear lower bound on their non-monotone complexity. Unfortunately, existing methods for monotone circuits (and formulas) do not work for slice functions.

The obstacle for this, roughly, is that either the set of positive or the set of negative inputs of a slice function are not "scattered" enough. For the lower bounds criterium (Theorem 4.3) to work, we need that the number of positive (as well as negative) inputs of f containing a fixed r-element set is relatively small. Now, if f is a k-slice function with, say, $k \leq n/2$, then the only interesting negative inputs are $(n-k)$-element sets, corresponding to the vectors on the k-th slice of the n-cube on which the function takes value 0. But then up to $2^{(n-k)-r} \geq 2^{n/2-r}$ such inputs may share r common elements.

When trying to understand the monotone complexity of k-slice functions, it is important to first understand the case $k = 2$. This leads to so-called "graph complexity", a notion we already described in the first chapter.

5.1.2. Negated inputs as new variables. There is also another bridge between monotone and non-monotone complexities. Namely, with *any* boolean function f in n variables, it is possible to associate a monotone boolean function g_f if $2n$ variables so that

$$C'(f) \geq C_+(g_f) - 4n.$$

Let $f(x)$ be any boolean function in n variables. Take a set y of new n variables and define a boolean function $g_f(x,y)$ by

$$g_f(x,y) = \alpha(x,y) \wedge f(x) \vee \beta(x,y),$$

where

$$\alpha(x,y) = \bigwedge_{i=1}^{n} (x_i \vee y_i) \quad \text{and} \quad \beta(x,y) = \bigvee_{i=1}^{n} (x_i \wedge y_i).$$

That is, $\alpha(x,y) = 1$ iff $x \vee y = \mathbf{1}$ and $\beta(x,y) = 1$ iff $x \wedge y \neq \mathbf{0}$ (a component-wise OR and AND).

LEMMA 5.2. *For any boolean function f, g_f is a monotone function.*

PROOF. If $g(x,y) = g_f(x,y)$ is not monotone, there must be vectors a, b so that $g(a,b) = 1$ and changing some bit from 0 to 1 makes $g = 0$. Clearly, $\beta(a,b) = 0$; otherwise, after the change β would still output 1. Since $g(a,b) = 1$ it must be the case that $\alpha(a,b) = 1$. But then after the change β must be equal to 1, a contradiction. □

LEMMA 5.3. *For any boolean function f,*

$$g_f(x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n) = f(x_1, \ldots, x_n).$$

PROOF. Let y be the vector $y = (\neg x_1, \ldots, \neg x_n)$. Then, by definition, $\alpha(x,y) = 1$ and $\beta(x,y) = 0$. □

THEOREM 5.4. *For any boolean function f in n variables,*

$$C'(f) \leq C_+(g_f) \leq C'(f) + 4n.$$

PROOF. The first inequality $C'(f) \le C_+(g_f)$ follows from Lemma 5.3. Now suppose that f has a circuit $F(x_1,\dots,x_n,\neg x_1,\dots,\neg x_n)$ of size ℓ. This is a monotone circuit with fanin-2 AND and OR gates; inputs are variables and their negations. Replace now the negated inputs $\neg x_1,\dots,\neg x_n$ by new variables $y=(y_1,\dots,y_n)$, extend the circuit by adding an AND with a circuit monotone computing $\alpha(x,y)$ and adding an OR with a circuit monotone computing $\beta(x,y)$. Let $F'(x,y)$ the resulting monotone circuit. It is clear that F' has size at most $\ell+4n$. We claim that F' is the desired monotone circuit:

$$g_f(\boldsymbol{x},\boldsymbol{y}) = F'(\boldsymbol{x},\boldsymbol{y}).$$

Suppose that F' is different from g_f for some values of the inputs x and y. Then, clearly, $\beta(x,y)=0$; otherwise, they would agree. Also $\alpha(x,y)$ must equal 1; again, if not, the two values could not disagree. We now claim that for each k, $x_k=\neg y_k$. Suppose that this was false. Then, let $x_k=y_k$ for some k. Clearly, the common value cannot be 1 since $\beta=0$. Also the common value cannot be 0 since $\alpha=1$. This proves that for each k, $x_k=\neg y_k$. But then

$$f(x) = F(x_1,\dots,x_n,\neg x_1,\dots,\neg x_n) = F'(x,y).$$

Since, by Lemma 5.3,

$$f(x) = g_f(x_1,\dots,x_n,\neg x_1,\dots,\neg x_n) = g_f(x,y),$$

we have that $g_f(x,y)=F'(x,y)$. This is a contradiction with our assumption that g_f and F' differ on input (x,y). □

5.2. The Markov theorem

More than 50 years ago, Markov (1957) has made an intriguing observation that *every* boolean (and even multi-output) function on n variables can be computed by a circuit with only about $\log n$ negations.[1] To state and prove his result, we need a concept of a "decrease" of functions.

For two binary vectors $x=(x_1,\dots,x_n)$ and $y=(y_1,\dots,y_n)$ we write, as before, $x\le y$ if $x_i\le y_i$ for all i. We also write $x<y$ if $x\le y$ and $x_i<y_i$ for at least one i. A boolean function $f:\{0,1\}^n\to\{0,1\}$ is *monotone* if $x\le y$ implies $f(x)\le f(y)$. A *chain* in the binary n-cube is an increasing sequence $Y=\{y^1<y^2<\ldots<y^k\}$ of vectors in $\{0,1\}^n$.

Given such a chain, we look at how many times our boolean function f changes its value from 1 to 0 along this chain, and call this number the decrease of f on this chain. Namely, say that i is a *jump position* (or a *jump down* position) of f along Y, if

$$f(y^i)=1 \quad\text{and}\quad f(y^{i+1})=0.$$

The number of all jump positions is the *decrease* $d_Y(f)$ of f on the chain Y. The *decrease* $d(f)$ of f is the maximum of $d_Y(f)$ over all chains Y.

Note that we only count the "jumps" from 1 to 0: positions j for which $f(y^j)=0$ and $f(y^{j+1})=1$ do not contribute to $d_Y(f)$. In particular, we have that $d(f)\le n/2$ for every boolean function f in n variables, and $d(f)=0$ for all monotone functions.

The *inversion complexity*, $I(f)$, of a boolean function f is the minimum number of NOT gates contained in a circuit over $\{\wedge,\vee,\neg\}$ computing f.

We have the following surprisingly tight result.

[1] Here and in what follows, all logarithms are base two; hence, $\lceil\log(n+1)\rceil$ is the number of bits in the binary representation of n.

THEOREM 5.5 (Markov 1957). *For every boolean function f,*

$$I(f) = \lceil \log(d(f)+1) \rceil .$$

The same result holds also for multi-output functions $f : \{0,1\}^n \to \{0,1\}^m$. In this case, the decrease of f along a chain is the number of times at least one of m components of f changes its value from 1 to 0.

We prove the lower and upper bounds on $I(f)$ separately.

LEMMA 5.6 (Lower bound).

$$I(f) \geq \lceil \log(d(f)+1) \rceil . \tag{5.2}$$

PROOF. We can assume that $I(f) > 0$, for otherwise the function f would be monotone, and in this case $d(f) = 0$.

Fix a chain $Y = \{y^1 < y^2 < \ldots < y^k\}$ for which $d_Y(f) = d(f)$. Take an arbitrary circuit C computing f, and let g be the function computed on the output of the *first* NOT gate of C. Hence, the function computed at the input of g is *monotone*.

Our goal is to prove the following claim.

CLAIM 5.7. It is possible to replace g by a constant 0 or 1 so that the function f' computed by the resulting circuit satisfies

$$d_Y(f') \geq \frac{1}{2}\, d_Y(f) .$$

Having this, the desired lower bound (5.2) can be shown as follows. If the original circuit C would have $r < \lceil \log(d(f)+1) \rceil = \lceil \log(d_Y(f)+1) \rceil$ NOT gates, then repeating Claim 5.7 r times we would obtain a circuit without any negations computing a function f_r for which $d_Y(f_r) \geq 2^{-r} \cdot d_Y(f) \geq 1$ But the function f_r is monotone (no NOT gates are used to compute it), a contradiction.

So, it remains to prove Claim 5.7.

Let g be the function computed on the output of the *first* NOT gate of C. Since the function computed at the input of this gate is monotone, we have that $d_Y(g) \leq 1$, that is, g can make at most one jump down on Y.

Case 1: $d_Y(g) = 0$ (no jumps at all). In this case, we have that $g(Y) \equiv 0$ or $g(Y) \equiv 1$, and we can replace this negation gate by the corresponding constant 0 or 1. For the function f' computed by the resulting circuit we then have that $f'(y) = f(y)$ for all $y \in Y$, implying that $d_Y(f') = d_Y(f)$ in this case.

Case 2: $d_Y(g) = 1$. In this case there is a $1 \leq t < k$ such that $g(y) = 1$ for all $y \in Y_1 := \{y^1, \ldots, y^t\}$, and $g(y) = 0$ for all $y \in Y_0 = \{y^{t+1}, \ldots, y^k\}$. Let

$$Y(f) = \{\boldsymbol{y}_i \mid f(y^i) > f(y^{i+1})\}$$

be the set of jumps made by function f on the entire chain Y; hence, $|Y(f)| = d_Y(f)$. Depending on whether $|Y_1 \cap Y(f)| \geq |Y(f)|/2$ or not, replace the gate g by constant 1 or by constant 0. In both cases the resulting circuit has one negation gate fewer, and computes a function f' for which $d_Y(f') \geq |Y(f)|/2 = d_Y(f)/2$.

This completes the proof of Claim 5.7, and hence, the proof of the lower bound (5.2). □

LEMMA 5.8 (Upper bound).

$$I(f) \leq \lceil \log(d(f)+1) \rceil . \tag{5.3}$$

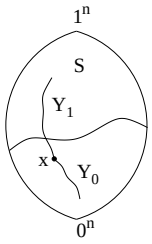


FIGURE 2. Chain Y_0 ends in x, and chain Y_1 starts with x.

PROOF. We will prove the lemma by induction on $M(f) := \lceil \log(d(f)+1) \rceil$.

Base: $M(f) = 0$. Then $d(f) = 0$, so f is monotone and $I(f) = 0$.

Induction Step: Suppose $I(f') \leq M(f')$ for all boolean functions f' such that $M(f') \leq M(f) - 1$. Let S be the set of all vectors $x \in \{0,1\}^n$ such that $d_Y(f) < 2^{M(f)-1}$ for every chain Y *starting* with x:

$$S = \{x \mid d_Y(f) < 2^{M(f)-1} \text{ for any chain } Y \text{ starting in } x\}.$$

Note that the set S is upwards closed: if $x \in S$ and $x \leq y$, then $y \in S$. This holds because each chain starting in y can be extended to a chain starting in x.

CLAIM 5.9. for every chain Y *ending* in a vector outside the set S we also have $d_Y(f) < 2^{M(f)-1}$.

PROOF. Assume that there is a chain Y_0 ending in a vector $x \notin S$ and such that $d_{Y_0}(f) \geq 2^{M(f)-1}$ (see Fig. 2). The fact that x does not belong to S means that there must be a chain Y_1 starting in x for which $d_{Y_1}(f) \geq 2^{M(f)-1}$. But then the decrease $d_{Y_0 \cup Y_1}(f)$ of f on the combined chain $Y_0 \cup Y_1$ is

$$d_{Y_0 \cup Y_1}(f) = d_{Y_0}(f) + d_{Y_1}(f) \geq 2^{M(f)} = 2^{\lceil \log(d(f)+1) \rceil} > d(f),$$

a contradiction with the definition of $d(f)$. □

Consider now two functions f_0 and f_1 defined as follows:

$$f_0(x) = \begin{cases} f(x) & \text{if } x \in S, \\ 0 & \text{if } x \notin S \end{cases}$$

and

$$f_1(x) = \begin{cases} 1 & \text{if } x \in S, \\ f(x) & \text{if } x \notin S. \end{cases}$$

CLAIM 5.10. Both $d(f_0)$ and $d(f_1)$ are strictly smaller than $2^{M(f)-1}$.

PROOF. We show this for f_0 (the argument for f_1 is similar). Let Y be a chain for which $d_Y(f_0) = d(f_0)$. Let x be a vector which Y starts in and y be a vector which Y ends in. If $x \in S$ or $y \notin S$, then $d(f_0) \leq 2^{M(f)-1} - 1$ by Claim 5.9 and definition of S. So, assume that $x \notin S$ and $y \in S$. Since the set S is upwards closed, some initial part Y_0 of the chain Y lies outside S and the remaining part Y_1 lies in S. By the definition of the function f_0, it is constant 0 on Y_0, and coincides with f on Y_1. By the definition of the set S, we have that the decrease of f_0 on Y_1 is smaller than $2^{M(f)-1} - 1$. Since $f_0(z) = 0$ for all $z \in Y_0$, there cannot be any additional jump down of f_0 along the entire chain $Y = Y_0 \cup Y_1$. □

Hence,

$$M(f_i) = \lceil \log(d(f_i)+1) \rceil \le \lceil \log 2^{M(f)-1} \rceil = M(f) - 1.$$

By the induction hypothesis, $I(f_i) \le M(f_i) \le M(f) - 1$ for both $i = 0, 1$. It therefore remains to show that

$$I(f) \le 1 + \max\{ I(f_0),\ I(f_1) \}. \tag{5.4}$$

For this we need one auxiliary result. A *connector* of two boolean functions $f_0(x)$ and $f_1(x)$ in n variables is a boolean function g in $n+2$ variables such that $g(0,1,x) = f_0(x)$ and $g(1,0,x) = f_1(x)$.

CLAIM 5.11. Every pair of functions $f_0(x)$, $f_1(x)$ has a connector g such that

$$I(g) \le \max\{I(f_0), I(f_1)\}.$$

Let us first complete the proof of Lemma 5.8 using this claim. Let $\chi_S(x)$ be the characteristic function of S, and let g be a connector of f_0 and f_1 guaranteed by Claim 5.11. By the definition of the functions f_0 and f_1, we then have that our original function $f(x)$ can be computed as

$$f(x) = g\big(\neg\chi_S(x), \chi_S(x), x\big).$$

Indeed, if $x \in S$ then $f(x) = f_0(x) = g(0,1,x) = g\big(\neg\chi_S(x), \chi_S(x), x\big)$, and similarly for all vectors $x \notin S$. Since the set S is upwards closed, its characteristic function $\chi_S(x)$ is monotone, and hence, requires no NOT gates. Thus, Claim 5.11 implies

$$I(f) \le 1 + I(g) \le 1 + \max\{I(f_0), I(f_1)\}.$$

This completes the proof of (5.3), and thus, the proof of Lemma 5.8.

It remains therefore to prove Claim 5.11.

We argue by induction on $r := \max\{I(f_0), I(f_1)\}$. If $r = 0$ then both functions f_i are monotone, and we can take $g(u,v,x) = (u \wedge f_1) \vee (v \wedge f_0)$.

For the induction step, let $C_i(x)$ be a circuit with $I(f_i)$ negations computing $f_i(x)$. Replacing the first NOT gate in C_i by a new variable ξ we obtain a circuit $C_i'(\xi, x)$ on $n+1$ variables which contains one NOT gate fewer. Let $f_i'(\xi, x)$ be the function computed by this circuit; hence, $I(f_i') \le r - 1$. Moreover, if $h_i(x)$ is the monotone function computed immediately before the first NOT gate in C_i, then

$$f_0(x) = f_0'(\neg h_0(x), x) \quad \text{and} \quad f_1(x) = f_1'(\neg h_1(x), x). \tag{5.5}$$

By the induction hypothesis, there is a boolean function $g'(u,v,\xi,x)$ (the connector of the pair f_0', f_1') such that $I(g') \le \max\{I(f_0'), I(f_1')\} \le r - 1$,

$$g'(0,1,\xi,x) = f_0'(\xi,x) \quad \text{and} \quad g'(1,0,\xi,x) = f_1'(\xi,x).$$

Replace now the variable ξ in $g'(u,v,\xi,x)$ by the function

$$Z(u,v,x) := \neg\big((u \wedge h_1(x)) \vee (v \wedge h_0(x))\big).$$

Since $Z(0,1,x) = \neg h_0(x)$ and $Z(1,0,x) = \neg h_1(x)$, (5.5) implies that the obtained function $g(u,v,x)$ is a connector of f_0 and f_1. Since the functions h_0 and h_1 are monotone, we have $I(g) \le 1 + I(g') \le r$, as desired. □

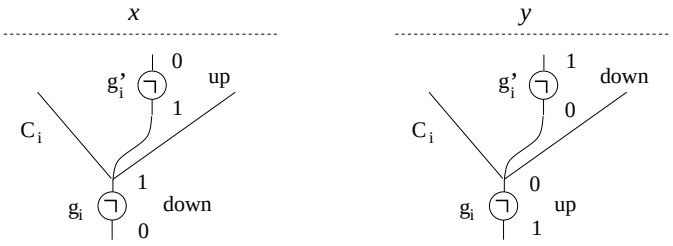


FIGURE 3. If, when going from input x to input $y > x$, g_i changes from "down" state to "up" state, then the subformula F_i must contain at least one NOT gate changing its state form "up" to "down" state.

5.3. Formulas require exponentially more NOT gates

We now consider *formulas*, that is, circuits with AND, OR and NOT gates whose fanout in a circuit is 1. The only difference from the general circuits (over the same basis) considered in the previous section is that now the underlying graph of a circuit is a tree, not an arbitrary directed acyclic graph. It is "clear" that this (requiring fanin 1) should restrict the power of circuits. And indeed, we will now show that the minimal number of NOT gates in formulas must be exponentially larger than in circuits.

Define the *inversion complexity*, $I_F(f)$, of a boolean function f in the class of formulas as the minimum number of NOT gates contained in a formula computing f.

By Markov's theorem, the minimum number of NOT gates in a *circuit* for f is about $\log_2 d(f)$, where $d(f)$ is the decrease of f. In the case of *formulas* we have:

THEOREM 5.12. *For every boolean function f, $I_F(f) = d(f)$.*

We again prove the lower and upper bounds on $I_F(f)$ separately.

LEMMA 5.13 (Lower bound). $I_F(f) \geq d(f)$.

PROOF. Let C be a formula computing f. Our goal is to show that then C must have at least $d(f)$ NOT gates.

If the input to a NOT gate g is 0 and the output is 1, then we call the state of g *up*. The state of g is *down* if the input of g is 1 and the output is 0. We denote by $\text{down}(x)$ the number of NOT gates in the formula C whose states are down when the input for C is vector x.

CLAIM 5.14. If $x < y$, then $\text{down}(y) - \text{down}(x) \geq 0$. If moreover, $f(x) = 1$ and $f(y) = 0$, then $\text{down}(y) - \text{down}(x) \geq 1$.

PROOF. We change the input of C from x to y. Let $g_1, \ldots, g_m$ be all NOT gates in C that change from down state to up state, when going from input x to input y. If a subformula C_i entering a NOT gate g_i, changing from down state to up state when going from input x to input $y > x$, would have no NOT gates, then C_i would be monotone, implying that $C_i(x) \leq C_i(y)$ (see Fig. 3). But since g_i changes from down state to up state, this means that C_i changes from up state to down state, that is, $C_i(x) = 1$ and $C_i(y) = 0$, a contradiction. Hence, C_i must contain at least one NOT gate changing its state from up to down state. Let g_i' be any of these NOT gates such that there are no other NOT gates between it and g_i. Now, none of the gates g_i' can be among $g_1, \ldots, g_m$ because their behavior when going from x to y is different. Also,

since C is a formula, we have that all gates $g'_1, \ldots, g'_m$ must be distinct, for otherwise same gate in C would be forced to have fanout at least 2. This shows down(y) is at least down(x).

Now, if $f(x) = 1$ and $f(y) = 0$, then the output of C must be connected, by a path without NOT gates, with a NOT gate g'_0 which changes from up state to down state, when going from x to y. This gate is not among the gates $g'_1, \ldots, g'_m$, because the (unique) path from each g'_i to the output contains a NOT gate g_i of a different type, namely, changing its state form down to up state. Thus, in this case, down(y) − down$(x) \geq 1$. □

Take now a chain $Y = \{y^1 < y^2 < \ldots < y^k\}$ for which $d_Y(f) = d(f)$. Hence, the number of indices i such that $f(y^i) = 1$ and $f(y^{i+1}) = 0$ is $d(f)$, and for each such index i we have that down(y^i) − down$(y^{i+1}) \geq 1$. Since for the remaining indices j we still have down(y^j) − down$(y^{j+1}) \geq 0$, this implies that

$$\text{down}(y^k) - \text{down}(y^1) \geq d(f).$$

Thus, the number of NOT gates in C must be at least down$(y^k) \geq d(f)$. □

LEMMA 5.15 (Upper bound). $I_F(f) \leq d(f)$.

PROOF. Induction on $d(f)$. The base case $d(f) = 0$ is trivial, since then f is monotone and $I_F(f) = 0$.

For the induction step, suppose that $d(f) \geq 1$, and $I_F(f') \leq d(f')$ for all boolean functions f' such that $d(f') \leq d(f) - 1$. Let S be the set of all vectors $x \in \{0,1\}^n$ such that $d_Y(f) = 0$ for every chain Y *starting* with x:

$$S = \{x \mid d_Y(f) = 0 \text{ for any chain } Y \text{ starting in } x\}.$$

Note that the set S is upwards closed: if $x \in S$ and $x \leq y$ then $y \in S$. This holds because each chain starting in y can be extended to a chain starting in x.

As in the proof of Markov's theorem, consider two functions f_0 and f_1 defined by:

$$f_0(x) = \begin{cases} f(x) & \text{if } x \in S, \\ 0 & \text{if } x \notin S \end{cases}$$

and

$$f_1(x) = \begin{cases} 1 & \text{if } x \in S, \\ f(x) & \text{if } x \notin S. \end{cases}$$

Let also χ_S be the characteristic function of the set S itself, that is,

$$\chi_S(x) = \begin{cases} 1 & \text{if } x \in S, \\ 0 & \text{if } x \notin S. \end{cases}$$

It is easy to see that

$$f = f_0 \vee (f_1 \wedge \neg\chi_S).$$

Indeed, if $x \in S$, then $f_0(x) = f(x)$ and $\neg\chi_S(x) = 0$, and if $x \notin S$, then $f_0(x) = 0$, $f_1(x) = f(x)$ and $\neg\chi_S(x) = 1$.

CLAIM 5.16. $d(f_0) = d(\chi_S) = 0$ and $d(f_1) \leq d(f) - 1$.

PROOF. Since the set S is upwards closed, its characteristic function χ_S is monotone, implying that $d(\chi_S) = 0$. That $d(f_0) = 0$ follows from the fact f_0 cannot take value 1 on a chain Y until Y enters the set S.

To show that $d(f_1) \leq d(f) - 1$, assume that $d(f_1) \geq d(f)$. Since we only count the number of changes of values of f on a chain from 1 to 0 (not from 0 to 1), the

maximum $d(f_1) = \max_X d_X(f_1)$ is achieved on a chain X *ending* in a vector y such that $f_1(y) = 0$. Since $d_Y(f) = 0$ for all chains Y *starting* with some vector in S, there must be a chain X which *ends* in some vector $y \notin S$ and for which $d_X(f_1) \geq d(f)$ holds. On the other hand, the fact that y is not in S implies that there must be a chain Y *starting* in y such that $d_Y(f) \geq 1$. But then for the combined chain $X \cup Y$ we have that $d_{X\cup Y}(f) = d_X(f) + d_Y(f) \geq d(f) + 1$, a contradiction with the definition of $d(f)$. □

By Claim 5.16 and the induction hypothesis, we have that $I_F(f_0) = 0$, $I_F(\chi_S) = 0$ and $I_F(f_1) \leq d(f) - 1$. Hence, the desired upper bound follows:

$$I_F(f) \leq I_F(f_0) + I_F(f_1) + I_F(\chi_S) + 1 \leq d(f).$$ □

5.4. The Fischer theorem

According to Markov's theorem, every boolean function in n variables *can* be computed by a circuit with at most

$$M(n) := \lceil \log(n+1) \rceil$$

NOT gates. The next important step was made by Fischer (1974): restricting the number of negations to $M(n)$ entails only a polynomial blowup in circuit size!

THEOREM 5.17 (Fischer 1974). *If a function on n variables can be computed by a circuit over $\{\wedge, \vee, \neg\}$ of size t, then it can be computed by a circuit of size at most $2t + O(n^2 \log^2 n)$ using at most $M(n)$ NOT gates.*

PROOF. It is easy to show (do this!) that every circuit of size t can be transformed to a circuit of size at most $2t$ such that all negations are placed only on the input variables. Hence, it is enough to show how to compute the (multi-output) function

$$NEG(x_1, \ldots, x_n) = (\neg x_1, \ldots, \neg x_n)$$

by a circuit of size $O(n^2 \log^2 n)$ using $M(n)$ negations; the function NEG is also known as an *invertor*.

We already know (see Eq. (5.1)) that, on inputs $x \in \{0,1\}^n$ with exactly k 1's, the negation $\neg x_i$ of its ith bit can be computed as $\neg x_i = T_{k,i}(\boldsymbol{x})$, where

$$T^n_{k,i}(x_1, \ldots, x_n) := Th^{n-1}_k(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

Using this observation, we can also simulate the behavior of $\neg x_i$ on *all* inputs.

CLAIM 5.18. For any $x \in \{0,1\}^n$ and any $1 \leq i \leq n$, we have that $\neg x_i = f_i(x)$, where

$$f_i(x) := \bigwedge_{k=1}^{n} \left(\neg Th^n_k(x) \vee T^n_{k,i}(x) \right).$$

PROOF. Take an arbitrary vector $a \in \{0,1\}^n$. If $\neg x_i(a) = 1$ then $a_i = 0$, implying that in this case $Th^n_k(a) = T^n_{k,i}(a)$ for all $k = 1, \ldots, n$, and hence, $f_i(a) = 1$. If $\neg x_i(a) = 0$ then $a_i = 1$. So, for $k = |a|$, we then have $Th^n_k(a) = 1$ and $T^n_{k,i}(a) = 0$, implying that $f_i(a) = 0$. □

It can be shown (we will not do this) that all the functions Th^n_k and $T^n_{k,i}$ ($0 \leq k \leq n, 1 \leq i \leq n$) can be computed by a *monotone* circuit of size $O(n^2 \log^2 n)$. Hence, it remains to compute the function

$$\neg T(x) := \left(\neg Th^n_1(x), \neg Th^n_2(x), \ldots, \neg Th^n_n(x) \right)$$

using at most $M(n) = \lceil \log(n+1) \rceil$ negations. To do this, we first take a monotone circuit $C_1(x)$ computing the function

$$T(x) := \left(Th_1^n(x), Th_2^n(x), \ldots, Th_n^n(x) \right).$$

Observe that the outputs of this circuit belong to the set A_{sort} of all inputs $y \in \{0,1\}^n$ whose bits are *sorted* in decreasing oder $y_1 \geq y_2 \geq \ldots \geq y_n$. Using only $M(n)$ negations it is possible to construct a circuit $C_2(y)$ of size $O(n)$ which computes $NEG(y)$ correctly on all inputs in A_{sort} (Exercise 5.19). The resulting circuit $C(x) = C_2(C_1(x))$ computes $\neg T(x)$. □

EXERCISE 5.19. Let $n = 2^r - 1$, and consider the set A_{sort} of all vectors $x \in \{0,1\}^n$ whose bits are *sorted* in decreasing oder $x_1 \geq x_2 \geq \ldots \geq x_n$. Construct a circuit C_n of size $O(n)$ which has at most r NOT gates and computes

$$NEG(x_1, \ldots, x_n) = (\neg x_1, \ldots, \neg x_n) \quad \text{for all inputs} \quad x \in A_{sort}.$$

Hint: Let $x = (x_1, \ldots, x_n) \in A_{sort}$. Take the middle bit x_m $(m = n/2)$ and show that the output of C_n can be obtained from the output of $C_{n/2}$ and the output of $\neg x_m$. For this observe that $\neg x_m = 1$ implies $\neg x_j = \neg x_m$ for all $j > m$, whereas $\neg x_m = 0$ implies $\neg x_j = \neg x_m$ for all $j < m$.

5.5. How many negations are enough to prove $\mathbf{P} \neq \mathbf{NP}$?

In order to prove the well known conjecture that $\mathbf{P} \neq \mathbf{NP}$, it would be enough to prove that some functions $f : \{0,1\}^n \to \{0,1\}^n$ in **NP** cannot be computed by circuits of polynomial (in n) size. By the results of Markov and Fischer, it would be enough to prove a "weaker" result. Namely, let

$\mathbf{P}^{(r)}$ = class of all functions computable by polynomial-size circuits with at most r NOT gates.

Then, by Markov–Fischer results, we have that:

$$\text{If } \text{CLIQUE} \notin \mathbf{P}^{(r)} \text{ for } r = \lceil \log_2(n+1) \rceil, \text{ then } \mathbf{P} \neq \mathbf{NP}.$$

The breakthrough result of Razborov (1985a) states that

$$\text{CLIQUE} \notin \mathbf{P}^{(r)} \text{ for } r = 0.$$

Amano and Maruoka (2005) have shown that essentially the same argument yields a stronger result:

$$\text{CLIQUE} \notin \mathbf{P}^{(r)} \text{ even for } r = (1/6) \log \log n.$$

At the first glance, this development looks like a promising way to prove that $\mathbf{P} \neq \mathbf{NP}$: just extend the bound to circuits with a larger and larger number r of allowed NOT gates. But how large this number r of allowed NOT gates must be in order to have the conclusion $\mathbf{P} \neq \mathbf{NP}$? This question motivates the following parameter for functions f:

$$R(f) = \min\{r \mid f \notin \mathbf{P}^{(r)} \text{ implies } f \notin \mathbf{P}\}.$$

By the results of Markov and Fischer, for any f, we have that

$$0 \leq R(f) \leq \lceil \log_2(n+1) \rceil$$

holds for every function f in n variables. This parameter is most interesting for *monotone* functions since they need no NOT gates at all, if we don't care about the circuit size. We already know that $R(f) = 0$ for a large class of monotone boolean functions f, namely—for slice functions. But no specific slice function f with $f \notin \mathbf{P}^{(0)}$ is known.

On the other hand, would $R(f) \le (1/6)\log\log n$ hold for every monotone function f, then we would already have that CLIQUE $\notin \mathbf{P}$, and hence, that $\mathbf{P} \neq \mathbf{NP}$. Unfortunately, there *are* monotone functions f for which $R(f)$ is near to Markov's $\log n$-border.

THEOREM 5.20. *There are explicit monotone functions* $f : \{0,1\}^n \to \{0,1\}^n$ *such that* $f \in \mathbf{P}$ *but*

$$f \notin \mathbf{P}^{(r)} \quad \textit{unless} \quad r \ge \log n - O(\log\log n)\,.$$

PROOF. The proof idea is to take a feasible monotone boolean function $g : \{0,1\}^n \to \{0,1\}$, and consider a monotone multi-output function $f : \{0,1\}^{kn} \to \{0,1\}^k$ computing $k = 2^r$ copies of g on *disjoint* sets of variables. We call such a function f a *k-fold extension* of g. We then show that, if g requires *monotone* circuits of exponential size, then f requires circuits of super-polynomial size, even if up to r NOT gates are allowed.

CLAIM 5.21. Let f be a monotone boolean function, and k be a power of 2. If the k-fold extension of f can be computed by a circuit with $\log_2 k$ NOT gates, then f can be computed by a *monotone* circuit of the same size.

PROOF. It is enough to prove the lemma for $k = 2$ (we can then iterate the argument). Thus, take a circuit with one NOT gate computing two copies $f_0 = f(Y_0)$ and $f_1 = f(Y_1)$ of the monotone function $f(X)$ on disjoint sets of variables. Let g be the monotone(!) boolean function computed at the input to the (unique) NOT gate.

We have only two possibilities: either some minterm of g lies entirely in Y_1, or not. In the first case we assign constant 1 to all the variables in Y_1, whereas in the second case we assign constant 0 to all the variables in Y_0. As the function g is monotone, in both cases it turns into a constant function (constant 1 in the first case, and constant 0 in the second), and the subsequent NOT gate can be eliminated. But since $Y_0 \cap Y_1 = \emptyset$, the setting $Y_\varepsilon \mapsto \varepsilon$ does not affect the function $f_{1-\varepsilon}$. Hence, we obtain a circuit which contains no NOT gates and computes either f_0 or f_1, and hence, also $f(X)$ after renaming the input variables. □

To finish the proof of Theorem 5.20, we will make use of an explicit monotone boolean clique-like function T_m in m variables considered by Tardos (1987). In Section 4.6 we have shown (see Theorem 4.16) that this function is feasible—can be computed by a non-monotone circuit of size $m^{O(1)}$—but every monotone circuit computing it requires size is exponential[2] in $\Omega(m^{1/16})$.

Let $n = km$ where $k = 2^r$ and $r = \lfloor \log_2 n - 32\log_2\log_2 n \rfloor$; hence, k is about $n/(\log_2 n)^{32}$. Consider the k-fold extension f_n of T_m. Then f_n can be computed by a (non-monotone) circuit of size at most $k \cdot m^{O(1)}$, which is, of course, polynomial in n. Hence, $f_n \in \mathbf{P}$. On the other hand, by Claim 5.21, every circuit with at most r NOT gates computing f_n must have size exponential in

$$m^{1/16} \approx \left(\frac{n}{k}\right)^{1/16} = (\log n)^{32/16} = (\log_2 n)^2\,.$$

Thus, $f_n \notin \mathbf{P}^{(r)}$ unless $r \ge \log_2 n - 32\log_2\log_2 n$. □

The message of Theorem 5.20 is that, in the context of the **P** vs. **NP** problem, it is important to understand the role of NOT gates when their number r is *indeed* very close to the Markov–Fisher upper bound of $\log n$.

[2]Another explicit feasible monotone boolean function—logical permanent—requiring monotone circuits of size $n^{\Omega(\log n)}$ was earlier given by Razborov (1985b). For the k-fold extensions f_n of this function the same argument yields $R(f_n) = \Omega(\log n)$.

The function f_n in Theorem 5.20 has many output bits. It would be interesting to prove a similar result for a *boolean* (i.e., single output) function.

RESEARCH PROBLEM 5.22. *Find an explicit monotone* boolean *function* f_n *for which* $R(f_n) = \Omega(\log n)$.

Bibliographic Notes

That n threshold functions $Th_1^n, \ldots, Th_n^n$ can be simultaneously computed by a monotone circuit of size $O(n \log^2 n)$ was proved by Mike Paterson (unpublished), Wegener (1985) and Valiant (1986). Theorem 5.4 is due to Richard Lipton. A version of Markov's theorem to formulas (Theorem 5.12) was proved by Morizumi (2008). Tanaka and Nishino (1994), and Beals et al. (1998) have shown that the term $O(n^2 \log^2 n)$ in Theorem 5.17 can be replaced by $O(n \log n)$. Theorem 5.20 was proved in [**80**].

CHAPTER 6

Span Programs

In 1993 Karchmer and Wigderson introduced an interesting linear algebraic model for computing boolean functions – the *span program*. A span program for a function $f(x_1,\ldots,x_n)$ is presented as a matrix over some field[1], with rows labeled by literals, that is, variables x_i or their negations $\neg x_i$ (one literal can label many rows). The span program accepts an input assignment if and only if the all-1 vector can be obtained as a linear combination of the rows whose labels are satisfied by the input. The size of the span program is the number of rows in the matrix. A span program is *monotone* if only positive literals are used as labels of the rows, i.e. negated variables are not allowed.

The model turns out to be quite strong: classical models for computing boolean functions – like switching networks or DeMorgan formulas – can be simulated by span programs without any increase in size. Moreover, and this was one of the motivations to introduce this model, the size of span programs lower bounds the size of parity branching programs—a model where no larger than n^2 lower bounds are known even in the simplest, read-once case (along each s-t path, each variable can be tested at most once). It is therefore not surprising that proving lower bounds on the size of span programs is a hard task, even in monotone case.

In this chapter we will show how this task can be solved using linear algebra arguments.

6.1. The model

We first describe the model more precisely.

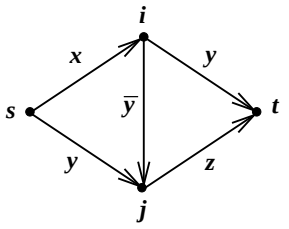
Let $\mathbb{F}$ be a field. A *span program* over $\mathbb{F}$ is given by a matrix M over $\mathbb{F}$ with its rows labeled by literals $x_1,\ldots,x_n,\neg x_1,\ldots,\neg x_n$; one literal may label several rows. If only positive literals $x_1,\ldots,x_n$ are used, then the program is called *monotone*. The *size* of a span program M is the number of rows in it. For an input $a=(a_1,\ldots,a_n)\in\{0,1\}^n$, let M_a denote the submatrix of M obtained by keeping those rows whose labels are satisfied by a. That is, M_a contains rows labeled by those x_i for which $a_i=1$ and by those $\neg x_i$ for which $a_i=0$. The program M *accepts* the input a if the all-1 vector $\mathbf{1}$ (or any other, fixed in advance, vector) belongs to the span of the rows of M_a. A span program computes a boolean function f if it accepts exactly those inputs a where $f(a)=1$. That is,

$$f(a)=1 \quad \text{iff} \quad \mathbf{1}\in \mathrm{Span}(M_a). \tag{6.1}$$

It is useful to keep in mind the following equivalent definition of the acceptance condition (6.1):

$$f(a)=0 \quad \text{iff there exists a vector } \boldsymbol{r} \text{ such that } \langle \boldsymbol{r},\mathbf{1}\rangle = 1 \text{ and } M_a\cdot \boldsymbol{r}=\mathbf{0}. \tag{6.2}$$

[1] In this chapter we will only work over the field $GF(2)$, but the results hold for any field.



labels	matrix M				edges
x	1	1	0	0	(s,i)
y	0	1	0	1	(i,t)
y	1	0	1	0	(s,j)
z	0	0	1	1	(j,t)
$\overline{y}$	0	1	1	0	(i,j)
v_0	1	0	0	1	

FIGURE 1. A switching network for the threshold-2 function $Th_2^3(x,y,z)$ in three variables (which outputs 1 iff $x+y+z \geq 2$) and the corresponding span program.

That is, a vector a is rejected iff some odd vector $\boldsymbol{r}$ (vector with an odd number of 1's) is orthogonal to all *rows* of M. This follows from a simple observation that $\mathbf{1} \in \mathrm{Span}(M_a)$ iff all vectors in $\mathrm{Span}(M_a)^\perp$ are even; here, as customary, $V^\perp$ is the orthogonal complement of V, and is defined as the set of vectors orthogonal to every vector in V.

Finally, note that the number of columns is not counted as a part of the size. It is always possible to restrict the matrix of a span program to a set of linearly independent columns without changing the function computed by the program, therefore it is not necessary to use more columns than rows. However, it is usually easier to design a span program with a large number of columns, many of which may be linearly dependent.

6.2. Power of span programs

One of the oldest models for computing boolean functions is that of switching networks or nondeterministic branching programs. This model includes that of DeMorgan formulas and was intensively studied after C. E. Shannon introduced this model about 60 years ago.

THEOREM 6.1. *If a boolean function can be computed by a switching network of size s then it can also be computed by a span program of size at most s. The same holds for their monotone versions.*

PROOF. Let $G=(V,E)$ be a switching network for a function f, with $s,t \in V$ its special vertices. Take the standard basis $\{\boldsymbol{e}_i \mid i \in V\}$ of the $|V|$-dimensional space over $GF(2)$, i.e., $\boldsymbol{e}_i$ is a binary vector of length $|V|$ with exactly one 1 in the ith coordinate.

The span program M is constructed as follows. For every edge $\{i,j\}$ in E add the row $\boldsymbol{e}_i \oplus \boldsymbol{e}_j = \boldsymbol{e}_i + \boldsymbol{e}_j$ to M and label this row by the label of this edge (see Fig. 1). It is easy to see that there is an s-t path in G, all whose labeled edges are switched on by an input vector a, if and only if the rows of M_a span the vector $\boldsymbol{v}_0 = \boldsymbol{e}_s \oplus \boldsymbol{e}_t$. Therefore, M computes f, and its size is $|E|$. □

The program, we just constructed, is not quite that what we called a span program: in the acceptance condition we use not $\mathbf{1}$ but some other vector $\boldsymbol{v}_0$. This, however, is only a technical matter: just add one more row $\boldsymbol{v}_0 \oplus 1$ labeled by constant 1.

Theorem 6.1 shows that span programs are not weaker than switching networks, and hence, than DeMorgan formulas and deterministic branching programs. What span programs capture is the size of parity branching programs. These are switching networks with the "parity-mode": an input a is accepted iff the number of s-t paths

consistent with a is odd (see Section 16.1.2). Namely, if $\mathrm{SP}(f)$ denotes the complexity of a boolean function in the class of span programs, and $\oplus\mathrm{BP}(f)$ in the class of parity branching programs, then $\mathrm{SP}(f) \le 2 \cdot \oplus\mathrm{BP}(f)$ and $\oplus\mathrm{BP}(f) \le \mathrm{SP}(f)^{O(1)}$; see Karchmer–Wigderson (1993) for details.

6.3. Power of monotone span programs

We will exhibit a monotone boolean function f in n variables such that $\mathrm{SP}(f) \le n$ but any monotone circuit for f requires $n^{\Omega(\log n)}$ gates.

A *spanning subgraph* of a graph $G = (V, E)$ is a graph $G' = (V, F)$ where $F \subseteq E$; the set of vertices remains the same. A (connected) *component* of a graph is a maximal set of its vertices such that there is a path between any two of them. A graph is *connected* if it consists of just one component. The *degree* $d_F(i)$ of a vertex i is the number of edges of F which are incident to i.

An *odd factor* in a graph is a spanning subgraph with all degrees odd.

LEMMA 6.2. *If a graph is connected then it has an odd factor if and only if the number of its vertices is even.*

PROOF. Suppose that G has an odd factor $G' = (V, F)$. Hence, all degrees $d_F(i)$ are odd. By Euler's theorem , the sum $\sum_{i \in V} d_F(i)$ equals $2|F|$, and hence, is even. Thus, the number $|V|$ of summands must be even, as claimed.

For the other direction, suppose that the graph $G = (V, E)$ is connected and has an even number of vertices, say $V = \{x_1, \ldots, x_{2m}\}$. For every $i = 1, \ldots, m$, fix any one path $P_i = (V_i, E_i)$ connecting x_i to x_{i+m}. Let F be the set of those edges from E which appear in an odd number of the sets $E_1, \ldots, E_m$.

We claim that the subgraph (V, F) is the desired odd factor. Indeed, observe that if a vertex x appears in a path P_i then either $\deg_{E_i}(x)$ is even or $\deg_{E_i}(x) = 1$, and this last event happens iff x is a leaf of this path, i.e., if $x = x_i$ or $x = x_{i+m}$. Since each vertex $x \in V$ is a leaf of exactly one of the paths $P_1, \ldots, P_m$, we have that the sum of degrees $D(x) := \sum_{i=1}^{m} \deg_{E_i}(x)$ is odd. It remains to observe that, by the definition of F, this sum $D(x)$ is congruent modulo 2 to the degree $\deg_F(x)$ of x in the graph (V, F). □

We now consider the following function $\mathrm{ODDFACTOR}_n$ on $n = m^2$ variables: the input is an $m \times m$ $(0,1)$ matrix representing a bipartite graph with m vertices in each part; the graph is accepted if it has an odd factor.

LEMMA 6.3. *Every monotone circuit computing* $\mathrm{ODDFACTOR}_n$ *requires* $n^{\Omega(\log n)}$ *gates.*

PROOF. One of celebrated results of Razborov (1985b) is an $n^{\Omega(\log n)}$ lower bound on the size of any monotone circuit for the perfect matching problem. In fact, he proved that such number of gates is necessary in any monotone circuit which: (i) accepts every perfect matching, and (ii) rejects a constant fraction of all unbalanced 2-colorings of vertices; each 2-coloring is identified with the graph of all monochromatic edges.

Every perfect matching is an odd factor, and should be accepted. On the other hand, an odd 2-coloring (in which each color occupies an odd number of vertices) has two odd components, and thus must be rejected: by Lemma 6.2, none of them can have an odd factor. As odd 2-colorings constitute half of all 2-colorings, Razborov's argument yields the result. □

It is therefore somewhat surprising that ODDFACTOR$_n$ can be computed by a very small monotone span program.

THEOREM 6.4. *ODDFACTOR$_n$ can be computed by a monotone span program of size n.*

PROOF. We construct the desired span program for ODDFACTOR$_n$ as follows. Let $V = V_1 \cup V_2$ be the vertex set ($|V_1| = |V_2| = m$), and let $X = \{x_{i,j}\}$ with $i \in V_1$ and $j \in V_2$ be the corresponding set of boolean variables (one for each potential edge). Take the standard basis $\{\boldsymbol{e}_i \mid i \in V\}$ of the $2m$-dimensional space over $GF(2)$, i.e., $\boldsymbol{e}_i$ is a binary vector of length $2m$ with exactly one 1 in the ith coordinate. Let M be the m^2 by $2m$ matrix whose rows are vectors $\boldsymbol{e}_i + \boldsymbol{e}_j$ labeled by the corresponding variables $x_{i,j}$. We claim that this span program computes ODDFACTOR$_n$. To verify this we have to show that the all-1 vector $\mathbf{1} = (1, \ldots, 1)$ is a sum (over $GF(2)$) of vectors of the form $\boldsymbol{e}_i + \boldsymbol{e}_j$ precisely when the corresponding edges (i, j) form an odd factor.

Take an arbitrary graph $E \subseteq V_1 \times V_2$. Suppose that E has an odd factor $F \subseteq E$. Since the degree $d_F(i)$ of each vertex $i \in V$ in the subgraph F is odd, we have

$$\sum_{(i,j)\in F} (\boldsymbol{e}_i + \boldsymbol{e}_j) = \mathbf{1}$$

because for each $i \in V$, the vector $\boldsymbol{e}_i$ occurs exactly $d_F(i)$ times in this sum. Thus, our span program M accepts the graph E, as desired.

Suppose now that E has no odd factors. By Lemma 6.2, the graph E must have a connected component with an odd number of vertices. Take such a component $G' = (A, B, F)$ where $A \subseteq V_1$ and $B \subseteq V_2$; hence, $|A \cup B|$ is odd. The program M must reject the graph E. Assume the opposite, i.e., that some subset of rows, labeled by edges in E, sum up to the all-1 vector $\mathbf{1}$. Since our subgraph G' is a connected component, no vertex from the set $A \cup B$ is incident to a vertex from outside. This means that the 1's in the positions, corresponding to vertices in $A \cup B$, can be obtained only by summing along the edges in that component. Hence, there must be a subset of edges $H \subseteq E \cap (A \times B)$ such that the vector

$$\boldsymbol{w} = \sum_{(i,j)\in H} (\boldsymbol{e}_i + \boldsymbol{e}_j)$$

has 1's in all the coordinates from $A \cup B$. For each $i \in A \cup B$, the number of terms $\boldsymbol{e}_i + \boldsymbol{e}_j$ in this sum is exactly the degree $d_H(i)$ of i in the subgraph H. By Euler's theorem, the sum $\sum_{i\in A\cup B} d_H(i)$ equals $2|H|$, and hence, is even. Since $|A \cup B|$ is odd, there must be at least one $i_0 \in A \cup B$ for which $d_H(i_0) \equiv 0 \bmod 2$ (the sum of an odd number of odd numbers would be odd). But this means that the vector $\boldsymbol{w}$ has a 0 in the i_0-th coordinate, a contradiction.

Thus, the designed span program M correctly computes ODDFACTOR$_n$. Since it has only $m^2 = n$ rows, we are done. □

Thus, for some monotone boolean functions their monotone span program size is exponentially smaller than their monotone circuits size. The converse direction remains open.

RESEARCH PROBLEM 6.5. *Do there exist functions admitting polynomial size monotone circuits which require superpolynomial size monotone span programs?*

6.4. Weakness of monotone span programs

We will now present a rank argument to show that some monotone boolean functions require large monotone span programs.

Let $f(x_1,\dots,x_n)$ be a monotone boolean function. It will be convenient to look at f as accepting/rejecting subsets $a \subseteq [n] = \{1,\dots,n\}$: $f(a) = 1$ iff $f(\chi_a) = 1$ for the characteristic vector χ_a of a. Let also $\overline{a} = [n] - a$ denote the complement of a set a.

Let A, B be some pair of families of subsets of $[n]$. A boolean function f in n variables *separates* this pair if $f(a) = 1$ for all $a \in A$, and $f(\overline{b}) = 0$ for all $b \in B$. A pair A, B of sets is *cross intersecting* if $a \cap b \neq \emptyset$ for all $a \in A$ and $b \in B$.

Every family A of subsets of $[n]$ defines a monotone boolean function in a natural way:

$$f_A(x) = \bigvee_{a \in A} \bigwedge_{i \in a} x_i\,.$$

That is, we just take a DNF whose monomials correspond to the members of A. This shows that every cross-intersecting pair A, B can be separated by at least one monotone boolean function, namely—by f_A. Indeed, this function must accept all members of A, by its definition. Take now a set $b \in B$. In the vector $\chi_{\overline{b}}$, all positions i with $i \in b$ are set to 0. Since b intersects all $a \in A$, all monomials (and hence, the function f_A itself) will be evaluated to 0 on input vector $\chi_{\overline{b}}$, implying that $f(\overline{b}) = 0$.

DEFINITION 6.6. Let A, B be some pair of families of subsets of $[n]$. The pair (A, B) is *locally intersecting* if every set $b \in B$ can be divided in to two parts $b = b_0 \cup b_1$ so that every $a \in A$ has a nonempty intersection with exactly one of these parts.

Given a locally intersecting pair (A, B), define its *disjointness matrix* $D_{A,B}$ to be an $|A|$ by $|B|$ matrix, with its rows indexed by sets $a \in A$ and its columns indexed by sets $b \in B$, such that the entries of $D = D_{A,B}$ are defined by

$$D[a,b] = \begin{cases} 0 & \text{if } a \cap b_0 \neq \emptyset, \\ 1 & \text{if } a \cap b_1 = \emptyset. \end{cases}$$

THEOREM 6.7. *If a pair A, B is locally intersecting, then any monotone span program over $GF(2)$ separating this pair must have size at least* $\mathrm{rk}(D_{A,B})$.

PROOF. Let M be a monotone span program separating (A, B). Let r be the number of rows and c the number of columns in M. The idea is to show that the disjointness matrix $D = D_{A,B}$ of A, B is a matrix of scalar products of vectors of dimension at most r; this yields $\mathrm{rk}(D) \leq r$.

For every $a \in A$, let $\boldsymbol{v}_a \in GF(2)^r$ be a vector witnessing the fact that a must be accepted, which means that

$$\boldsymbol{v}_a^\top \cdot M = \mathbf{1}\,,$$

where $\mathbf{1}$ is the all-1 vector and $\boldsymbol{v}_a$ is a vector which is nonzero only in coordinates corresponding to elements of a.

Let $b = b_0 \cup b_1 \in B$. Since the complement $\overline{b}$ of b cannot be accepted, no linear combination of the rows of $M_{\overline{b}}$ can give $\mathbf{1}$. Hence, by the dual acceptance condition (6.2), for each $b \in B$ there is a vector $\boldsymbol{u}_b$ in $GF(2)^c$ such that

$$\langle \mathbf{1}, u_b \rangle = 1 \text{ and } M_{\overline{b}} \cdot \boldsymbol{u}_b = \mathbf{0}.$$

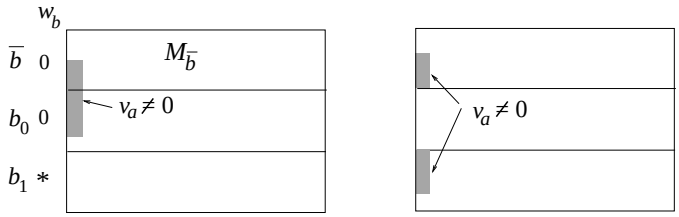


FIGURE 2. The two cases $a \cap b_0 \neq \emptyset$ and $a \cap b_1 \neq \emptyset$.

Let $\boldsymbol{w}_b$ be the vector in $GF(2)^r$ obtained from the vector $M \cdot \boldsymbol{u}_b$ by replacing to 0 all its elements, corresponding to the rows labeled by elements of b_0; note that

$$\boldsymbol{w}_b(i) \neq 0 \text{ only if } i \in b_1.$$

Indeed, the elements $\boldsymbol{w}_b(i)$ with $i \in \overline{b}$ are zero because $M_{\overline{b}} \cdot \boldsymbol{u}_b = 0$, and the elements $\boldsymbol{w}_b(i)$ with $i \in b_0$ are zero by the definition of $\boldsymbol{w}_b$. We claim that

$$D[a,b] = \langle \boldsymbol{v}_a, \boldsymbol{w}_b \rangle .$$

To show this, recall that the set b is splitted into two disjoint parts $b = b_0 \cup b_1$ such that each member of A has a nonempty intersection with exactly one of these parts. If $a \cap b_0 \neq \emptyset$ then $a \cap b_1 = \emptyset$, and hence, the vectors $\boldsymbol{v}_a$ and $\boldsymbol{w}_b$ have no element on which they both are nonzero; so, in this case $\langle \boldsymbol{v}_a, \boldsymbol{w}_b \rangle = 0$ (see Fig 2). If $a \cap b_1 \neq \emptyset$ then $a \cap b_0 = \emptyset$, and hence, in this case, we have $\langle \boldsymbol{v}_a, \boldsymbol{w}_b \rangle = \langle \boldsymbol{v}_a, M\boldsymbol{u}_b \rangle$, implying that

$$\langle \boldsymbol{v}_a, \boldsymbol{w}_b \rangle = \langle \boldsymbol{v}_a, M\boldsymbol{u}_b \rangle = \langle \boldsymbol{v}_a^\top M, \boldsymbol{u}_b \rangle = \langle \boldsymbol{1}, \boldsymbol{u}_b \rangle = 1 .$$

This shows that D is a matrix of scalar products of vectors of dimension r, implying that $\mathrm{rk}(D) \leq r$. □

Since, by Theorem 6.1, monotone span programs are not weaker than monotone switching networks (and hence, are not weaker than monotone formulas), Theorem 6.7 directly yields the following

COROLLARY 6.8. *If the pair A,B is locally intersecting, then any monotone DeMorgan formula separating this pair has size at least* $\mathrm{rk}(D_{A,B})$.

We now show how Theorem 6.1 can be used to prove a lower bound $n^{\Omega(\log n)}$ for an explicit monotone function in n variables. Then we show that this is the best what can be proved using rank arguments.

6.4.1. Super-polynomial lower bound. The *general disjointness matrix* D_n is a $2^n \times 2^n$ $(0,1)$ matrix whose rows and columns are labeled by the subsets a of an n-element set, and the (a,b)-th entry is 1 if and only if $a \cap b = \emptyset$.

LEMMA 6.9. $\mathrm{rk}(D_n) = 2^n$.

PROOF. Follows easily by the induction on n together with the following recursive construction of D_n:

$$D_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \qquad D_n = \begin{bmatrix} D_{n-1} & D_{n-1} \\ D_{n-1} & 0 \end{bmatrix} .$$

Since the lower right submatrix of D_n is an all-0 matrix, and since (by the induction hypothesis) the rows of D_{n-1} are linearly independent, the rows of the entire matrix D_n must be linearly independent as well. □

The *disjointness matrix* of a (single) family A of sets is a $(0,1)$ matrix D_A whose rows are labeled by members of A and columns are labeled by all subsets of each member of A. That is, for every $a \in A$ and for every $b \subseteq a$ there is a column labeled by b. Like in the case of matrices D_n, the entry in the a-th row and b-th column is defined by: $D_A[a,b] = 1$ iff $a \cap b = \emptyset$.

LEMMA 6.10. *Let A be a family of subsets that are incomparable by inclusion. Then* $\mathrm{rk}(D_A) = |A|$.

PROOF. Fix an $a \in A$ and let M be the submatrix of D_A consisting of all $2^{|a|}$ columns indexed by subsets of a. Since, by our assumption, *every* subset of a appears as a column of M, the rows of M are rows of the full disjointness matrix $D_{|a|}$, some of them repeated (just relabel each row $c \in A$ by $c \cap a$). By Lemma 6.9, we know that $D_{|a|}$ has full row rank. On the other hand, since a is not contained in any other member of A, the row indexed by a occurs in M only once.[2] This implies that this row cannot be a linear combination of other rows in M. Thus, the corresponding row of the entire matrix D_A cannot be a linear combination of others, as well. □

We will consider boolean functions defined by bipartite graphs as in Section 3.3. Let $G = (U, V, E)$ be a bipartite graph with $V = \{1, \ldots, n\}$ and $U = \{n+1, \ldots, 2n\}$. Recall that such a graph is *k-separated* if for every disjoint subsets X, Y of U of size at most k there exists a vertex $v \in V$ such that every vertex $u \in X$ is connected with v and no vertex $u \in Y$ is connected with v.

For a bipartite graph satisfying this condition we define A to be the family of sets $a \subseteq U \cup V$ such that $|a \cap U| = k$ and $a \cap V$ is the set of all vertices that are joined to every vertex of $a \cap U$, i.e., maximal complete bipartite graph with the part in U of size k. Consider the monotone boolean function

$$f_{A,G}(x) = \bigvee_{a \in A} \bigwedge_{i \in a} x_i \,.$$

We are now able to extend the lower bound on the formula size of such functions, given in Theorem 3.8, to a more general model of monotone span programs.

THEOREM 6.11. *If the graph G is k-separated, then every monotone span program computing $f_{A,G}$ requires size* $\binom{n}{k}$.

PROOF. Define B to be the family of sets $b = b_0 \cup b_1$ such that $b_0 \subseteq U$, $|b_0| \leq k$ and b_1 consists of all vertices of V that have no neighbor in b_0.

Since each $a \in A$ induces a complete bipartite graph and $b = b_0 \cup b_1$ an empty graph, a cannot intersect both b_0 and b_1. Moreover, the condition that the underlying graph is k-separated guarantees that $a \cap b_0 = \emptyset$ iff $a \cap b_1 \neq \emptyset$. That is, the pair of families A, B is locally intersecting. Theorem 6.7 implies that every monotone span program separating the pair A, B, and hence, any such program computing $f_{A,G}$ must have size at least $\mathrm{rk}(D_{A,B})$.

Relabel now each row $a \in A$ of $D_{A,B}$ by $a' := a \cap U$ (a k-element subset of U), and column $b \in B$ of $D_{A,B}$ by $b' := b_0$ (an at most k-element subset of U), and let

[2] Would the row of some other $a' \in A$ be the same, this would in particular mean that a' would intersect (that is, contain) each single element of a, since a does this.

M be the resulting matrix; this matrix differs from $D_{A,B}$ only in labelings of rows and columns—the entries remain the same.

Since b_0 ranges over *all* at most k-element subsets of U, and since we have:

$$D_{A,B}[a,b] = 1 \text{ iff } a \cap b_1 \neq \emptyset \text{ iff } a \cap b_0 = \emptyset \text{ iff } a' \cap b' = \emptyset,$$

the matrix M is the disjointness matrix $D_{A'}$ of the family $A' = \{a \cap U \mid a \in A\}$. By Lemma 6.10,

$$\mathrm{rk}(D_{A,B}) = \mathrm{rk}(D_{A'}) = |A'| = \binom{n}{k},$$

and we are done. □

There are several constructions of k-separated graphs that achieve $k = \Omega(\log n)$, the most popular is the Paley graph (see Section 3.3). For these graphs we obtain lower bounds of the form $n^{\Omega(\log n)}$.

We shall now show that this is already the limit: the approach, based on locally intersecting set families, cannot yield larger lower bound than $n^{\Omega(\log n)}$.

LEMMA 6.12. *Let A and B be families of subsets of $[n]$. If the pair A,B is locally intersecting, then*

$$\mathrm{rk}(D_{A,B}) \leq n^{O(\log n)}.$$

PROOF. Let $D = D_{A,B}$, and let $\mathrm{Dec}(D)$ be the minimum number of mutually disjoint monochromatic submatrices of D whose union covers all entries of D. Hence, $\mathrm{rk}(D) \leq \mathrm{Dec}(D)$. On the other hand, we will show later (see Theorem 7.17 in Section 7.2) that, for every $(0,1)$ matrix M, $\log_2 \mathrm{Dec}(M)$ is at most a constant times $(\log_2 \mathrm{Cov}(M))^2$, where $\mathrm{Cov}(M)$ is the minimum number of monochromatic (but not necessarily disjoint!) submatrices whose union covers all entries of M.

Each entry $D[a,b]$ of our matrix D is either 0 or 1 depending on whether a intersects b in its first block b_0 or in the second one b_1. Hence, all entries of D can be covered by $2n$ submatrices

$$M_i^\sigma = \{(a,b) : i \in a \cap b_\sigma\}, \qquad i = 1,\ldots,n;\ \sigma = 0,1.$$

It is clear that these submatrices cover all entries of D. Moreover, since the pair A,B is locally intersecting, we have that all the entries of M_i^σ are equal to σ. Hence, we have found a covering of D by $2n$ monochromatic submatrices M_i^σ, implying that $\mathrm{Cov}(D) \leq 2n$. By the above mentioned result,

$$\log_2 \mathrm{rk}(D) \leq \log_2 \mathrm{Dec}(D) \leq O((\log \mathrm{Cov}(D)^2) \leq O((\log n)^2).$$ □

We already know (see, for example, Section 4.6) that some explicit monotone boolean functions (clique-like functions) require *monotone* circuits of exponential size whereas their *non-monotone* circuit size is polynomial. The existence of such a gap between monotone and non-monotone span programs remains open.

RESEARCH PROBLEM 6.13. *Do there exist monotone functions admitting polynomial size span programs which require superpolynomial size monotone span programs?*

Exercises

EX. 6.1. (Karchmer–Wigderson 1993). Let M be a span program computing f over $\mathbb{F}_2$. Such a program is *canonical* if the columns of M are in one-to-one correspondence with the vectors in $f^{-1}(0)$, and for every $b \in f^{-1}(0)$, the column corresponding to b in M_b is an all-0 column. Show that every span program can be converted to a canonical span program of the same size (= the number of rows) and computing the same function.

Hint: Take a vector $b \in f^{-1}(0)$. By (6.2), there is an odd vector $\boldsymbol{r} = \boldsymbol{r}_b$ for which $M_b \cdot \boldsymbol{r}_b = \mathbf{0}$. Define the column corresponding to b in a new span program M' to be $M \cdot \boldsymbol{r}_b$. Do this for all $b \in f^{-1}(0)$. Show that, for every vector $a \in \mathbb{F}_2^n$, the rows of M'_a span the all-1 vector $\mathbf{1}$ if and only if $f(a) = 1$.

EX. 6.2. **Research problem.** Let k be the minimal number for which the following holds: there exist n colorings $c_1, \ldots, c_n$ of the n-cube $\{0,1\}^n$ in k colors $\{1, \ldots, k\}$ such that, for every triple of vectors x, y, z there exists a coordinate i on which not all three vectors agree and the three colors $c_i(x), c_i(y), c_i(z)$ are distinct. Bound the smallest number k of colors for which such a good collection of colorings $c_1, \ldots, c_n$ exists. *Comment*: This problem is connected with proving lower bounds on the size of *non-monotone* span programs, see Wigderson (1993).

EX. 6.3. (Wigderson 1993). Consider the version of the problem above where we additionally require that the colorings c_i are *monotone*, i.e., $x < y$ implies $c_i(x) \leq c_i(y)$. Prove that in this case $k = \Omega(n)$.

The goal of the next exercises is to show that we cannot replace the acceptance condition "accept vector a iff the rows of M_a span vector $\mathbf{1}$" of span programs by "accept vector a iff the rows of M_a are linearly dependent" because then very simple boolean functions require programs of exponential size.

A *monotone dependency program* over a field $\mathbb{F}$ is given by a matrix M over $\mathbb{F}$ with its rows labeled by variables $x_1, \ldots, x_n$. For an input $a = (a_1, \ldots, a_n) \in \{0,1\}^n$, let (as before) M_a denote the submatrix of M obtained by keeping those rows whose labels are satisfied by a. The program M *accepts* the input a if and only if the rows of M_a are linearly dependent (over $\mathbb{F}$). A program computes a boolean function f if it accepts exactly those inputs a where $f(a) = 1$. The *size* of a dependency program is the number of rows in it.

EX. 6.4. Suppose that a boolean function $f \not\equiv 1$ is computed by a monotone dependency program M of size smaller than the number of minterms of f. Prove that then there exists a set of minterms A, $|A| \geq 2$, such that for any non-trivial partition $A = A_0 \cup A_1$, the set

$$S(A_0, A_1) := \left(\bigcup_{a \in A_0} a \right) \cap \left(\bigcup_{b \in A_1} b \right)$$

contains at least one minterm of f.

Hint: For every minterm a of f choose some linear dependence v_a of the rows of M, i.e., v_a is a vector such that $v_a \cdot M = 0$, and v_a has a nonzero coordinates only at rows labeled by variables in a. The vectors v_a are linearly dependent (why?). Let A be a *minimal* set of minterms such that $\{v_a \mid a \in A\}$ are linearly dependent. Thus, $\sum_{a \in A} \alpha_a v_a = 0$ for some coefficients $\alpha_a \neq 0$. Observe that for any non-trivial partition $A = A_0 \cup A_1$,

$$v := \sum_{a \in A_0} \alpha_a v_a = -\sum_{a \in A_1} \alpha_a v_a \neq \mathbf{0}.$$

Let b be the set of variables labeling the rows of M corresponding to nonzero coordinates of v. This set lies in $S(A_0, A_1)$ and contains at least one minterm of f.

Ex. 6.5. Use the previous fact to show that the function

$$f = (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \cdots \wedge (x_{2n-1} \vee x_{2n})$$

cannot be computed by a monotone dependency program of size smaller than 2^n. Show that this function has a small monotone span program.

Hint: Each minterm a of f has precisely one variable from each of the sets $\{x_{2i-1}, x_{2i}\}$, $i = 1, \ldots, n$; hence, there are 2^n minterms. Suppose that f has a program of size smaller than 2^n, and let A be the set of minterms guaranteed by (i). Pick i such that both sets of minterms $A_0 = \{a \in A \mid x_{2i-1} \notin a\}$ and $A_1 = \{a \in A \mid x_{2i} \notin a\}$ are non-empty (why this is possible?). By (i), the set $S(A_0, A_1)$ must contain at least one minterm b of f. But, by the definition of A_0 and A_1, this minterm can contain neither x_{2i-1} nor x_{2i}, a contradiction.

Bibliographic Notes

The model of span programs was introduced by Karchmer and Wigderson (1993), where (among other results) Theorem 6.1 as well as the inequalities $\mathrm{SP}(f) \leq 2 \cdot \oplus\mathrm{BP}(f)$ and $\oplus\mathrm{BP}(f) \leq \mathrm{SP}(f)^{O(1)}$ were proved. Theorem 6.4 was proved by Babai, Gál and Wigderson (1999). Theorems 6.7 and 6.11 were proved by Gál and Pudlák (2003). The model of dependency programs was introduced by Pudlák and Sgall (1998), who also proved that the size of such programs is polynomialy related to $\mathrm{SP}(f)$, but for *monotone* versions no similar correspondence is possible (Exercise 6.5).

Part 2

Communication Complexity

CHAPTER 7

Two Players

In this and the next chapter we will consider games between two players, Alice and Bob, where communication between players is allowed. However, we will assume that the players are far away from each other, so that each communicated bit costs money. The goal is to pay, and hence, to communicate as few as possible. The players are not adversaries—they help and trust each other.

The goal of players is to compute the values of a given boolean function $f : \{0,1\}^{2m} \to \{0,1\}$ on all input vectors. The restriction is that each player has only partial access to the input. In the *fixed-partition* communication game, the players are given a functions as well some partition of the input variables into two disjoint blocks of size m. Hence, inputs have the form (x,y) with $x,y \in \{0,1\}^m$, and the players must compute $f(x,y)$ for all inputs. The restriction is that Alice can only see x and Bob can only see y.

There is also another, *best-partition* model of communication where, given a function f, the players are allowed to choose a most suitable for this function partition (x,y) of its inputs. Yet more trickier is the communication model where we have more than two players, each seeing all but a small piece of the input vector. We will consider this model later in Chapter 9.

7.1. Fixed partition games

In this case we actually consider communication games for matrices, rather than for boolean functions. If a partition of the $2m$ variables of f into equal sized parts is fixed, then we can look at f as an $n \times n$ $(0,1)$ matrix A with $n = 2^m$ such that $A[x,y] = f(x,y)$. Such a matrix A is usually referred to as the *communication matrix* of f.

Thus, in a fixed-partition game, the players are given a boolean $n \times n$ matrix A. The goal of players is to evaluate the matrix, that is, for every its entry (x,y), to compute the value $A[x,y]$ of this entry. The matrix A itself is known to both players! The restriction, however, is that the players only have a partial access to the input: Alice can only see x and Bob can only see y. Hence, Alice only knows which row it is, Bob only knows which column it is, and they must determine the value in their intersection.

7.1.1. Deterministic communication. Before the game starts, the players agree on a "protocol" for exchanging messages. After that, given an input pair (x,y), the protocol dictates to each player what messages to send at each point, based on her/his input and the messages received so far. It also dictates when to stop, and how to determine the answer from the information received. There is no limit on the computational complexity of these decisions, which are free of charge. The cost of the protocol is the number of bits they have to exchange on the *worst case* choice of input pair (x,y). The

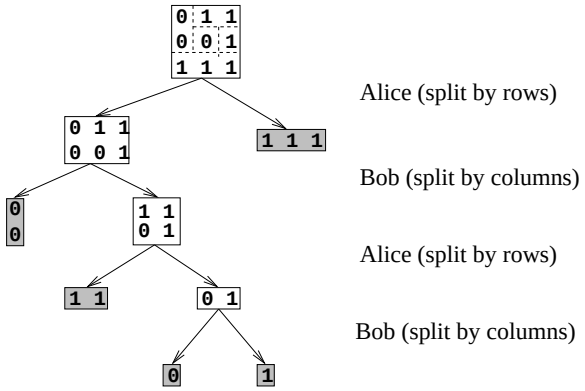


FIGURE 1. An example of a communication tree. Shaded rectangles are monochromatic. Dashed lines indicate the resulting decomposition of the original matrix into monochromatic rectangles. The communication complexity of this protocol is 4.

communication complexity $C(A)$ of the matrix A is the cost of the best protocol for this game.

More formally, this measure can be defined as follows.

By sending bits 0 and 1, the players actually split the rows (if this bit is send by Alice) or columns (if this bit is send by Bob) into two disjoint parts. A *communication protocol* (or a *communication tree*) of a game is a binary tree, each inner node of which correspond to a decision made by one of the players at this node. Each node of the tree is labeled by a submatrix of A so that the following holds (see Fig. 1).

a. The root is labeled by the whole matrix A.
b. If a node u is labeled by a matrix M, then the sons of u are labeled by the corresponding submatrices M_0 and M_1 of M. Moreover, these submatrices are obtained from M by splitting the rows of M (if u is Alice's node) or by splitting the columns of M (if u is Bob's node).
c. If w is a leaf and R is its label, then R is *monochromatic*, i.e., is either all-0 matrix or all-1 matrix.

Since at each node, the rows (or columns) of the corresponding submatrix are splitted into *disjoint* parts, the protocol is *deterministic*: each pair (x, y) will reach precisely one leaf. The depth of a tree is the maximum number of edges from the root to a leaf. In these terms, the communication complexity $C(A)$ of a matrix A is just the minimum depth of a communication tree for this matrix.

It is clear that for any $n \times n$ $(0,1)$ matrix A (n being a power of two) we have that

$$C(A) \leq \log_2 n$$

since Alice can just tell Bob the binary code of her row x.

Lower bounds on $C(A)$ can be shown using the rank $\mathrm{rk}(A)$ as well as the decomposition number of A. The *decomposition number*, $\mathrm{Dec}(A)$, of a boolean matrix A is defined as the smallest number of mutually disjoint[1] monochromatic submatrices of A covering all entries of A.

[1]Two submatrices are *disjoint* if they do not share a common entry.

	y	y'	y''
x	1	0	0
x'	1	1	1
x''	0	0	1

FIGURE 2. A decomposition that does not correspond to any protocol. Show this!

REMARK 7.1. Unlike arbitrary decompositions of a given matrix A into monochromatic submatrices, decompositions arising from communication protocols have special form: they are produced inductively by splitting the resulting submatrices only row-wise or column-wise. And indeed, there are decompositions that cannot be produced by any communication protocol, like one depicted in Fig. 2.

Since the submatrices occurring on the leaves of any communication tree for A must be disjoint, we immediately have that $C(A) \geq \log_2 \mathrm{Dec}(A)$. On the other hand, the subadditivity of rank implies that $\mathrm{rk}(A) \leq \mathrm{Dec}(A)$. So, we have the following estimates:

$$C(A) \geq \log_2 \mathrm{Dec}(A) \geq \log_2 \mathrm{rk}(A)\,. \tag{7.1}$$

Hence, already simplest matrices, like the identity matrix I_n, have maximal communication complexity. The goal however is (as in the case of other complexity measures) to understand what properties of a given matrix A force its communication complexity be large. Having 1's on the diagonal and 0's elsewhere is just one of these properties.

Using the rank one can show that a lot of matrices have large deterministic communication complexity. For a matrix A, let $|A|$ denote the number of its nonzero entries.

PROPOSITION 7.2. *If A is a symmetric $n \times n$ $(0,1)$ matrix with 1's on the diagonal, then*

$$\mathrm{Dec}(A) \geq \frac{n^2}{|A|}\,.$$

PROOF. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of A, then their sum $t = \sum_{i=1}^n \lambda_i$ is the trace of A (sum of diagonal entries of A), and at most $r = \mathrm{rk}(A)$ of them are nonzero. Thus, the Cauchy–Schwarz inequality yields $\mathrm{tr}(A^2) = \sum_{i=1}^n \lambda_i^2 \geq r(t/r)^2 = t^2/r$. Since A is a $(0,1)$ matrix, we also have that $\mathrm{tr}(A^2) = |A|$: the ith diagonal entry of A^2 is the number of 1s in the ith row of A. This implies $\mathrm{rk}(A) = r \geq \mathrm{tr}(A)^2/|A|$, where $\mathrm{tr}(A) = n$ since A has 1's on the diagonal. □

7.1.2. Rank Conjecture. We already know that $C(A) \geq \log_2 \mathrm{rk}(A)$ holds for any matrix A. But how tight this lower bound is?

CONJECTURE 7.3. *There is a constant c such that, for every $n \times n$ $(0,1)$ matrix A,*

$$C(A) \leq (\log_2 \mathrm{rk}(A))^c\,.$$

If $\mathrm{mono}(A)$ denotes the maximum number of entries in a monochromatic submatrix of A, then

$$C(A) \geq \log_2 \mathrm{Dec}(A) \geq \log_2 \frac{n^2}{\mathrm{mono}(A)}\,.$$

Hence, Rank Conjecture implies the following (seemingly "easier" to tackle) conjecture stating that every $(0,1)$ matrix of small rank must contain a large monochromatic submatrix.

CONJECTURE 7.4. *For every $n \times n$ $(0,1)$ matrix A of rank r,*

$$\text{mono}(A) \geq \frac{n^2}{2^{(\log r)^{O(1)}}}.$$

In fact, Nisan and Wigderson (1995) have shown that this last conjecture is equivalent to the Rank Conjecture! Moreover, they showed that Rank Conjecture *does not* hold for $c = 1/\log_3 2 \approx 1.6$. They also gave a support for Conjecture 7.4: every matrix of small rank must contain a submatrix of large "discrepancy."

Let A be an $n \times n$ ± 1 matrix. The *discrepancy*, $\text{disc}(A)$, of A is the maximum, over all its submatrices B, of the absolute value of the sum of entries in B. Hence, small discrepancy means that the matrix is very balanced: every submatrix has almost the same number of positive and negative entries. If A is a $(0,1)$ matrix, then its discrepancy is the discrepancy of its ± 1 version A' with $A'[x,y] = 1 - 2 \cdot A[x,y]$.

Since monochromatic submatrices have maximal discrepancy, we have that $\text{disc}(A) \geq \text{mono}(A)$. Interestingly, if we replace $\text{mono}(A)$ by $\text{disc}(A)$, then Conjecture 7.4 is true in a very strong sense!

THEOREM 7.5. *For every $n \times n$ ± 1 matrix A of rank r,*

$$\text{disc}(A) \geq \frac{n^2}{16r}.$$

PROOF. We are given a ± 1 matrix $A = (a_{ij})$ of low rank $r = \text{rk}(A)$ and wish to find in it a submatrix of hight discrepancy. For this, we first observe that

$$\text{disc}(A) = \max |x^\top A y| = \max \left| \sum_{i,j=1}^{n} a_{ij} x_i y_j \right|,$$

where the maximum is over all $(0,1)$ vectors x and y: each pair of such vectors correspond to a submatrix of A. So, we only need to find $(0,1)$ vectors x and y for which $x^\top A y$ is large. As an intermediate step we shall consider the set

$$\text{Ball} = \{u \in \mathbb{R}^n : |u_i| \leq 1 \text{ for all } i\}$$

of real vectors of small maximum norm and show that $\text{disc}(A)$ can be lower bounded by the maximum of $u^\top A v$ over the vectors $u, v \in \text{Ball}$.

CLAIM 7.6. For any $u, v \in \text{Ball}$ we have that

$$\text{disc}(A) \geq \frac{u^\top A v}{4}.$$

PROOF. Letting $z = Av$, we have that $u^\top A v = \sum_{i=1}^n u_i z_i$. Hence, $\sum_{i \in K} u_i z_i \geq u^\top A v/2$, where K is either the set of coordinates i where both u_i and z_i are positive or the set of coordinates in which both are negative. Assume the first case (the second case is similar by using vector $-v$ instead of v). Then letting $x \in \{0,1\}^n$ to be the characteristic vector of K and using the fact that $|u_i| \leq 1$ for all i, we have $x^\top A v = \sum_{i=1}^n x_i z_i \geq \sum_{i \in K} u_i z_i \geq u^\top A v/2$. Repeating this argument with $z = x^\top A$, we can replace v with a $(0,1)$ vector y obtaining that $x^\top A y \geq u^\top A v/4$. Hence, $\text{disc}(A) \geq x^\top A y \geq u^\top A v/4$, as claimed. □

To finish the proof of the theorem it is enough, by Claim 7.6, to find two vectors $u, v \in$ Ball for which $u^\top A v \geq n^2/4r$. For this we will use a known relation between spectral norm, Euclidean norm and the rank of a matrix.

The *spectral norm* of a matrix A is defined as the maximum $\|A\| = \max |u^\top A v|$ over all vectors $u, v \in \mathbb{R}^n$ whose Euclidean norm $\|u\| = (\sum_{i=1}^n u_i^2)^{1/2}$ is equal to 1. The name "spectral norm" comes from the fact that

$$\|A\| = \max\{\sqrt{\lambda} \mid \lambda \text{ is an eigenvalue of } A^\top A\}\,.$$

The *Euclidean norm* (known also as Frobenius norm) of A is just the Euclidean norm $W(A) = (\sum_{i,j} a_{ij}^2)^{1/2}$ of the corresponding to the matrix vector of length n^2.

CLAIM 7.7. For every real matrix A,

$$\frac{W(A)}{\sqrt{\mathrm{rk}(A)}} \leq \|A\| \leq W(A)\,.$$

PROOF. Observe that $W(A)^2$ is equal to the trace $\mathrm{tr}(B)$ i.e. the sum of diagonal elements, of the matrix $B = A^\top A$. On the other hand, the trace of any real matrix is equal to the sum of its eigenvalues. Hence, $W(A)^2 = \mathrm{tr}(B) = \sum_{i=1}^n \lambda_i$ where $\lambda_1 \geq \ldots \geq \lambda_n$ are the eigenvalues of B. Since B has only $r = \mathrm{rk}(B) = \mathrm{rk}(A)$ non-zero eigenvalues, and since all eigenvalues of B are nonnegative (B is symmetric), the largest eigenvalue λ_1 is bounded by $W(A)^2/r \leq \lambda_1 \leq W(A)^2$. It remains to use the (mentioned above) fact that $\|A\| = \sqrt{\lambda_1}$. □

We will now construct the desired vectors $u, v \in$ Ball with

$$u^\top A v \geq \frac{n^2}{4r}\,.$$

We start with two vectors $x, y \in \mathbb{R}^n$ of Euclidean norm $\|x\| = \|y\| = 1$ for which $x^\top A y = \|A\|$. Let $p \geq 1$ be a parameter (to be specified later), and consider the sets of indices

$$I = \{i : |x_i| > 1/\sqrt{p}\} \quad \text{and} \quad J = \{j : |y_j| > 1/\sqrt{p}\}\,.$$

Since $1 = \|x\|^2 = \sum_{i=1}^n x_i^2 \geq |I|/p$, we have that $|I| \leq p$, and similarly, $|J| \leq p$. Consider the vectors a and b defined by:

$$a_i = \begin{cases} 0 & \text{if } i \in I, \\ x_i & \text{otherwise} \end{cases} \quad \text{and} \quad b_j = \begin{cases} 0 & \text{if } j \in J, \\ y_j & \text{otherwise.} \end{cases}$$

We claim that

$$a^\top A b \geq \frac{n}{\sqrt{r}} - p\,. \tag{7.2}$$

To show this, consider the matrix B which agrees with A on all entries (i, j) with $i \in I$ and $j \in J$, and has 0's elsewhere. Then

$$a^\top A b = x^\top A y - x^\top B y\,.$$

Since $W(A) = n$, Claim 7.7 yields $x^\top A y \geq n/\sqrt{r}$. The same claim also yields $x^\top B y \leq \|B\| \leq W(B) \leq p$, where the last inequality follows since B has at most p nonzero rows and p nonzero columns. So,

$$a^\top A b = x^\top A y - x^\top B y \geq \frac{n}{\sqrt{r}} - p\,.$$

Set now $p := n/(2\sqrt{r})$, and consider the vectors $u := \frac{1}{\sqrt{p}}a$ and $v := \frac{1}{\sqrt{p}}b$. Both vectors u, v belong to Ball, and we have

$$u^\top A v = p \cdot a^\top A b \geq \frac{n^2}{4r}\,. \qquad \square$$

7.1.3. Nondeterministic communication. The *cover number*, $\mathrm{Cov}(A)$, of a $(0,1)$ matrix A is the smallest number of all-1 submatrices of A covering all its 1's; this time the matrices in a cover need not be disjoint. The *nondeterministic communication complexity*, $NC(A)$, of a matrix A is defined by:

$$NC(A) = \log_2 \mathrm{Cov}(A)\,.$$

Perhaps, the best way to view a *nondeterministic* communication protocol between two parties, Alice and Bob, wishing to evaluate a given matrix A, is a scheme by which a third party, Carole (a "superior being"), knowing the whole input (x, y), can convince Alice and Bob what the value of $A[x, y]$ is. Hence, we have three players, Alice, Bob and Carole. Given an input (x, y), Carole's goal is to convince Alice and Bob that $A[x, y] = 1$. For this purpose, she announces to both players some binary string, a *witness* for (or a *proof* of) the fact that "$A[x, y] = 1$." Having this witness, Alice and Bob verify it *independently* and respond with either Yes or No. Alice and Bob agree that $A[x, y] = 1$ (and accept the input (x, y)) if and only if they both replied with Yes. If $A[x, y] = 0$ then Alice and Bob must be able to detect that the witness is wrong no matter what Carole says. The protocol is correct if, for every input (x, y), Alice and Bob accept it if and only if $A[x, y] = 1$. The communication complexity of this game is the length of the witness in the worst case.

EXAMPLE 7.8. For example, Carole can easily convince Alice and Bob that two binary strings x and y of length n are *not* equal: using only $\lceil \log_2 n \rceil + 1$ bits she announces (the binary code of) a position i with $x_i \neq y_i$ and the bit x_i; Alice checks whether the bit she received is the ith bit of the string she can see, and Bob checks whether $y_i \neq x_i$. If however Carole would like to convince that $x = y$, then she would be forced to send n bits, just because $\mathrm{Cov}(I_n) = 2^n$ for a $2^n \times 2^n$ identity matrix I_n.

7.1.3.1. *An upper bound.* The following lemma says that only sparse matrices can have large nondeterministic communication complexity. For a $(0,1)$ matrix, let $|A|$ denote the number of its 1-entries.

LEMMA 7.9. *Let A be a $(0,1)$ matrix. If every column or every row of A contains at most d zeroes, then*

$$\mathrm{Cov}(A) = O(d \ln |A|)\,.$$

PROOF. We only consider the column case, the row case is the same. To cover the ones of A we construct an all-1 submatrix B with row set I and column set J via the following probabilistic procedure: pick every row of A with probability $p = 1/(d+1)$ to get a random subset I of rows, and let J be the set of all columns of A that have no zeroes in the rows of B.

A 1-entry (x, y) of A is covered by B if x was chosen in I and none of (at most d) rows with a 0 in the y-th column was chosen in I. Hence,

$$\Pr[(x, y) \text{ is covered by } B] \geq p(1-p)^d \geq pe^{-pd} \geq p/e\,.$$

If we apply this procedure t times to get t all-1 submatrices, then the probability that (x, y) is covered by *none* of these submatrices does not exceed $(1 - p/e)^t \leq e^{-tp/e}$.

Hence, the probability that some 1-entry of A remains uncovered is at most

$$|A| \cdot e^{-tp/e} = \exp(\ln|A| - t/(e(d+1)))\,,$$

which is < 1 for $t > e(d+1)\ln|A|$. □

7.1.3.2. *Lower bounds.* We now turn to *lower* bounds on the covering number $\mathrm{Cov}(A)$, and hence, on the nondeterministic communication complexity.

Given a $(0,1)$ matrix A and a nonzero $(0,1)$ matrix $B \le A$, let $w_A(B)$ denote the largest possible number of 1-entries in B that can be covered by some all-1 submatrix R of A. (Note that R needs not be a *submatrix* of B.) Since no all-1 submatrix of A can cover more than $w_A(B)$ 1's of B, at least $|B|/w_A(B)$ all-1 submatrices are needed to cover all 1's of A. Hence, the following *greedy covering number*,

$$\mu(A) = \max_{B \le A} \frac{|B|}{w_A(B)}\,,$$

is a lower bound on $\mathrm{Cov}(A)$. Interestingly, this lower bound is not very far from the truth.

LEMMA 7.10. *For every* $(0,1)$ *matrix* A*, we have*

$$\mathrm{Cov}(A) \le \mu(A) \cdot \ln|A| + 1\,.$$

PROOF. Consider a greedy covering $R_1, \ldots, R_t$ of A by all-1 submatrices. That is, in the i-th step we choose an all-1 submatrix $R_i \le A$ covering the largest number of all yet uncovered 1's in A. Let B_i be a $(0,1)$ matrix containing all 1's of A that are left uncovered after the i-th step. That is, $B_i[x,y] = 1$ iff $A[x,y] = 1$ and $R_1[x,y] = \ldots = R_i[x,y] = 0$. Hence, $B_0 = A$ and $B_t = 0$ (all-0 matrix). Let $b_i = |B_i|$ and $w_i = w_A(B_i)$. Since, by the definition of $\mu = \mu(A)$, none of the fractions b_i/w_i can exceed μ, we have that $b_{i+1} = b_i - w_i \le b_i - b_i/\mu$. This yields

$$b_i \le b_0(1 - 1/\mu)^i \le |A| \cdot e^{-i/\mu}\,.$$

For $i = t-1$, we obtain $1 \le b_{t-1} \le |A| \cdot e^{-(t-1)/\mu}$, and the desired upper bound $\mathrm{Cov}(A) \le t \le \mu \ln|A| + 1$ follows. □

A natural choice for a "difficult to cover" matrix $B \le A$ is to take a permutation matrix. This leads to the following, easy to apply lower bounds. Say that two 1-entries in a matrix are *independent* if they do not lie in one row or in one column.

The *term-rank* $\mathrm{trk}(A)$ of A is the largest number of its pairwise independent 1-entries. The *clique number* $\omega(A)$ of A is the largest number r such that A contains an $r \times r$ all-1 submatrix. Finally, the *line weight* $\ell(A)$ of A is the largest number of 1's in a line (row or column), that is, $w(A)$ is the maximum degree of the corresponding to A bipartite graph. Using these matrix parameters we can lower bound the cover number as follows:

$$\mathrm{Cov}(A) \ge \frac{\mathrm{trk}(A)}{\omega(A)} \ge \frac{|A|}{\ell(A) \cdot \omega(A)}\,. \tag{7.3}$$

The first inequality follows since any $r \times r$ all-1 submatrix of A can have at most r independent 1's. The second inequality is a direct consequence of a classical result of König-Egervary saying that the term-rank $\mathrm{trk}(A)$ of A is exactly the minimum number of lines (rows and columns) covering all 1's in A.

Although simple, the first lower bound in (7.3)—known as the *fooling set bound*—is one of the main tools for proving lower bounds on the nondeterministic communication complexity of boolean functions.

For sparse matrices, we have a somewhat better bound. Proposition 7.2 implies that, for any symmetric matrix A, the fraction $\mathrm{trk}(A)^2/|A|$ is a lower bound on the decomposition number $\mathrm{Dec}(A)$ of A. We now show that this fraction is also a lower bound on the covering number $\mathrm{Cov}(A)$.

LEMMA 7.11. *For every non-zero* $(0,1)$ *matrix* A*, we have*

$$\mathrm{Cov}(A) \geq \frac{\mathrm{trk}(A)^2}{|A|}\,.$$

PROOF. Take a largest set I of $|I| = \mathrm{trk}(A)$ independent 1-entries in A, and let $R_1,\ldots,R_t$ be a covering of the 1-entries in A by $t = \mathrm{Cov}(A)$ all-1 submatrices. Define a mapping $f : I \to \{1,\ldots,t\}$ by $f(x,y) = \min\{i \mid R_i[x,y] = 1\}$, and let $I_i = \{(x,y) \in I \mid f(x,y) = i\}$. That is, I_i consists of those independent 1-entries in I that are covered by the ith all-1 submatrix R_i for the *first time*. Note that some of the I_i's may be empty, so let $I_1,\ldots,I_k$ be the nonempty ones. Say that an entry (x,y) is *spanned* by I_i if $(x,y') \in I_i$ for some column y' and $(x',y) \in I_i$ for some row x'.

Let S_i be the submatrix of R_i spanned by I_i. Hence, $S_1,\ldots,S_k$ are disjoint all-1 submatrices of A covering all 1-entries in I. Moreover, each S_i is an $r_i \times r_i$ matrix with $r_i = |I_i|$. Since the S_i's are disjoint, we have that

$$r_1 + \cdots + r_k = |I| = \mathrm{trk}(A)$$

and

$$r_1^2 + \cdots + r_k^2 \leq |A|\,.$$

By the Cauchy–Schwarz inequality,

$$\mathrm{trk}(A)^2 = (r_1 + \cdots + r_k)^2 \leq k \cdot (r_1^2 + \cdots + r_k^2) \leq k \cdot |A|\,,$$

and the desired lower bound $t \geq k \geq \mathrm{trk}(A)^2/|A|$ follows. □

REMARK 7.12. For all $(0,1)$ matrices A with $|A| < \mathrm{trk}(A) \cdot \omega(A)$ ones, Lemma 7.11 yields somewhat better lower bounds than those given by the fooling set bound (7.3). If, for example, an $N \times N$ matrix A contains an identity matrix and some constant number c of $r \times r$ all-1 matrices with $r = \sqrt{N}$, then Lemma 7.11 yields $\mathrm{Cov}(A) \geq N^2/(cr^2 + N) = \Omega(N)$, whereas the fooling set bound (7.3) only yields $\mathrm{Cov}(A) \geq N/r = \sqrt{N}$.

7.1.3.3. *Communication with restricted advice.* Recall that $\mathrm{Cov}(A) \leq t$ iff all 1-entries of A can be covered by at most t all-1 submatrices. When doing this, one 1-entry of A may be covered many times. Let us now consider a version of this measure, where the cover frequency is restricted. This corresponds to nondeterministic communication, where Carole cannot use one and the same witness for many inputs; this situation is usually referred to as a nondeterministic communication with a restricted number of advice bits.

Let $\mathrm{Cov}_k(A)$ be the smallest number of all-1 submatrices of A covering all its 1-entries in such a way that no 1-entry of A is covered by more than k of these submatrices. Let $\mathrm{rk}(A)$ denote the rank of A over the real numbers.

LEMMA 7.13. *For every* $(0,1)$ *matrix* A *and any integer* $k \geq 1$*, we have*

$$\mathrm{Cov}_k(A) = \Omega(k \cdot \mathrm{rk}(A)^{1/k})\,.$$

PROOF. Let $R_1, \ldots, R_t$ be $t = \mathrm{Cov}_k(A)$ $(0,1)$ matrices of rank 1 such that $A \leq \sum_{i=1}^{t} R_i \leq kJ$, where J is the all-1 matrix. This is an equivalent definition of $\mathrm{Cov}_k(A)$: the 1-entries in each of the R_i's correspond to an all-1 submatrix of A. For a subset $I \subseteq \{1, \ldots, t\}$, let R_I be a $(0,1)$ matrix with $R_I[x,y] = 1$ iff $R_i[x,y] = 1$ for all $i \in I$. By the inclusion-exclusion formula, we can write the matrix A as a linear ± 1 combination

$$A = \sum_{I \neq \emptyset} (-1)^{|I|+1} R_I \,. \tag{7.4}$$

The condition $\sum_{i=1}^{t} R_i \leq kJ$ implies that $R_I = 0$ for all I of size $|I| > k$. Hence, the right hand of (7.4) has at most $\sum_{i=1}^{k} \binom{t}{i}$ non-zero terms. Using the estimates $\binom{t}{i} \leq \binom{t}{k} \leq (et/k)^k$, the subadditivity of rank yields

$$\mathrm{rk}(M) \leq \sum_{i=1}^{k} \binom{t}{i} \leq k \left(\frac{et}{k} \right)^k ,$$

from which the desired lower bound on $t = \mathrm{Cov}_k(A)$ follows. □

The following example shows that the lower bound in Lemma 7.13 cannot be improved.

EXAMPLE 7.14. Let I be an identity $n \times n$ matrix with $n = 2^m$ for some m divisible by k, and let $\overline{I} = J - I$ be its complement. Then $\mathrm{rk}(\overline{I}) = n$, but we have that

$$\mathrm{Cov}_k(\overline{I}) \leq k \cdot n^{1/k} \,.$$

To see this, encode the rows and the columns by vectors $x \in \{0,1\}^m$; hence, $\overline{I}[x,y] = 1$ iff $x \neq y$. Split the set $[m]$ into k disjoint subsets $S_1, \ldots, S_k$, each of size m/k. For every $j \in [m]$ and $a \in \{0,1\}^{m/k}$, define the rectangle:

$$R_{j,a} = \{(x,y) \mid \text{projection of } x \text{ onto } S_j \text{ coincides with } a \text{ and that of } y \text{ doesn't}\} \,.$$

These $k2^{m/k} = kn^{1/k}$ rectangles cover all 1's of $\overline{I}$, and each pair (x,y) with $x \neq y$ appears in at most k of them (since we take only k projections).

7.2. P = NP ∩ co-NP for fixed-partition games

Having deterministic and nondeterministic modes and having the (far-fetched) analogy with the **P** versus **NP** question, it is natural to consider the relations between the corresponding complexity classes. Here for convenience (and added thrill) we use the common names for the analogs of the complexity classes:

> Let **P** (resp., **NP**) consist of all boolean functions in $2m$ variables whose deterministic (resp., nondeterministic) communication complexity is polynomial in $\log m$.

The *complement* of a $(0,1)$ matrix A is the matrix $\overline{A} = A - J$, where J is the all-1 matrix (of the same dimension). Note that in the case of *deterministic* protocols, there is no difference what of the two matrices A or $\overline{A}$ we consider: we always have that $C(A) = C(\overline{A})$, because each deterministic protocol must cover all 0's as well as all 1's of A. In the case of *nondeterministic* communication, the situation is different in two respects:

a. we only need to cover the 1's of A, and
b. the submatrices need not be disjoint.

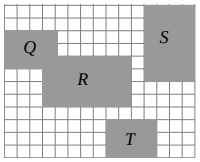


FIGURE 3. R intersects S in rows, intersects T in columns, and intersects Q in both rows and columns.

This is where an asymmetry between nondeterministic protocols for A and $\overline{A}$ comes from. And indeed, we have already seen that the nondeterministic communication complexities of the identity matrix and its complement are exponentially different.

But what if *both* A and $\overline{A}$ have small nondeterministic communication complexity, what can be than said about the *deterministic* communication complexity of A. This is a version of the famous **P** versus **NP** ∩ **co-NP** question in communication complexity. To answer questions of this type (in the communication complexity frame), we now give a general upper bound on the deterministic communication complexity.

7.2.1. Making non-disjoint coverings disjoint. Let X and Y be two finite sets. A *rectangle* is a subset $R \subseteq X \times Y$ of the form $R = R^0 \times R^1$ with $R^0 \subseteq X$ and $R^1 \subseteq Y$. That is, a subset R is a rectangle iff for every two points (x, y) and (x', y') of R, the combined points (x, y') and (x', y) belong to R as well.

Let us consider the following general scenario of covering a rectangle by ist subrectangles. We are given a finite set $\mathscr{R}$ of (not-necessarily disjoint) rectangles, as well as a labeling of rectangles. The only requirement is that the labeling must be *legal* in the following sense:

$(*)$ Any two rectangles with different labels must be disjoint.

Let P be the set of all points (x, y) belonging to at least one of these rectangles. We consider the following search problem for $\mathscr{R}$: given a point $(x, y) \in P$, find a label of a rectangle containing this point. Note that, if the point belongs to more than one rectangle then, by $(*)$, all these rectangles must have the same label.

We want to solve this problem using a communication game between two players, Alice and Bob, where Alice obtains the first coordinate x and Bob obtains the second coordinate y. Let $\mathrm{cc}(\mathscr{R})$ denote the deterministic communication complexity of such a game for $\mathscr{R}$.

Say that a rectangle $S = S^0 \times S^1$ intersects a rectangle $R = R^0 \times R^1$ in *rows*, if $S^0 \cap R^0 \neq \emptyset$, and intersects R in *columns*, if $S^1 \cap R^1 \neq \emptyset$ (see Fig. 3). Note that, $S \cap R \neq \emptyset$ if and only if S intersects R in rows *and* in columns. This immediately leads to the following basic observation about *disjoint* rectangles.

OBSERVATION 7.15. Let S be a rectangle and $\mathscr{R}$ a set of rectangles. If $S \cap R = \emptyset$ for all $R \in \mathscr{R}$, then either S intersects at most half of rectangles $R \in \mathscr{R}$ in rows or S intersects at most half of these rectangles in columns.

Using this observation, we can give a general upper bound on $\mathrm{cc}(\mathscr{R})$.

LEMMA 7.16. *For every finite set $\mathscr{R}$ of legally labeled rectangles,*

$$\mathrm{cc}(\mathscr{R}) \leq 2(\log_2 |\mathscr{R}|)^2 .$$

PROOF. Let $r = \lceil \log_2 |\mathscr{R}| \rceil$, Say that two rectangles in $\mathscr{R}$ are *consistent* if they have the same label, and *inconsistent* otherwise. A rectangle $R = R^0 \times R^1$ *contains* a row x (a column y) if $x \in R^0$ (resp., $y \in R^1$). The protocol consists of at most r rounds and in each round at most $1 + r$ bits are communicated. After each round the current set of rectangles is updated. Given a input (x, y), the goal is to decrease the number of rectangles in each round by at least one half.

a. Alice checks whether all rectangles in $\mathscr{R}$, containing her row x, are consistent. If yes, then the (unique) label i of all these rectangles is a correct answer, and she announces it.
b. Otherwise, Alice tries to find a rectangle $R \in \mathscr{R}$ containing x such that R intersects in *rows* at most half of rectangles that are inconsistent with R. If such a rectangle R exists, then Alice sends its name (using r bits) to Bob and they both update $\mathscr{R}$ so that it only contains the rectangles that intersect with R in rows (the other rectangles cannot contain (x, y)).
c. If Alice is unable find such a rectangle then she communicates this to Bob (using one bit).
d. Now is Bob's turn. Since Alice failed, Observation 7.15 ensures that there must be a rectangle $R \in \mathscr{R}$ that contains y and intersects in *columns* at most half of rectangles that are inconsistent with R. Bob takes any of such rectangles R and sends its name (using r bits) to Alice and they both update $\mathscr{R}$ so that it only contains the rectangles that intersect with R in columns (the other rectangles cannot contain (x, y)). At this point the round is definitely over since they successfully eliminated at least half of the rectangles in $\mathscr{R}$, and we can proceed by induction.

After at most r rounds the players will agree on a rectangle containing (x, y), and the label of this rectangle is the correct answer. □

As a direct consequence we obtain the following important result due to Aho, Ullman and Yannakakis (1983), implying that $\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$ holds for the fixed-partition communication complexity.

THEOREM 7.17. *For every* $(0,1)$ *matrix* A,

$$C(A) \leq 2 \max\{NC(A), NC(\overline{A})\}^2 .$$

PROOF. Let $\mathscr{R} = \mathscr{R}_0 \cup \mathscr{R}_1$ where $\mathscr{R}_0$ is a set of $|\mathscr{R}_0| \leq 2^{NC(\overline{A})}$ all-0 submatrices covering all zeroes of A, and $\mathscr{R}_1$ is a set of $|\mathscr{R}_1| \leq 2^{NC(A)}$ all-1 submatrices covering all ones of A. Assign label "0" to all rectangles in $\mathscr{R}_0$, and label "1" to all rectangles in $\mathscr{R}_1$. It is clear that this is a legal labeling, since every rectangle in $\mathscr{R}_0$ must be disjoint from every rectangle in $\mathscr{R}_1$. Hence, on a given input (x, y), the players have only to find out the label of a rectangle containing (x, y). By Lemma 7.16, this can be done using at most $2(\log_2 |\mathscr{R}|)^2 \leq 2 \max\{NC(A), NC(\overline{A})\}^2$ bits of communication. □

Theorem 7.17 cannot be essentially improved. To show this, consider the disjointness matrix $D_{n,k}$ introduced in Section 3.3.1. Recall that its rows and columns are labeled by all $\sum_{i=0}^{k} \binom{n}{i}$ subsets a of $[n]$ of size at most k, and the entry in the a-th row and b-th column is defined by:

$$D_{n,k}[a, b] = \begin{cases} 0 & \text{if } a \cap b \neq \emptyset, \\ 1 & \text{if } a \cap b = \emptyset. \end{cases}$$

We already know that these matrices have full rank, even over $GF(2)$. Hence,

$$C(D_{n,k}) \geq \log_2 \mathrm{rk}(D_{n,k}) = \Omega(k \log(n/k)) . \tag{7.5}$$

It is also easy to see that

$$NC(\overline{D_{n,k}}) \leq \log_2 n \tag{7.6}$$

(just guess a point in the intersection of a and b). It turns out that the nondeterministic communication complexity of the matrix $D_{n,k}$ itself is also not very large.

CLAIM 7.18. $\mathrm{Cov}(D_{n,k}) \leq 2k4^k \ln n$.

PROOF. The rows as well as columns of $D_{n,k}$ are labeled by elements of the the set $[n]^{\leq k}$ of all subsets of $[n]$ of size at most k. Say that a subset $Y \subseteq [n]$ *separates* pair (a,b) of two disjoint members a and b of $[n]^{\leq k}$ if $a \subseteq Y$ and $b \cap Y = \emptyset$. Let $\boldsymbol{Y}$ be a random subset of $[n]$ chosen uniformly with probability 2^{-n}. Then for a fixed pair (a,b),

$$\begin{aligned} \Pr[\boldsymbol{Y} \text{ does not separate } (a,b)] &= 1 - \Pr[a \subseteq \boldsymbol{Y} \text{ and } b \cap \boldsymbol{Y} = \emptyset] \\ &= 1 - \frac{2^{n-|a|-|b|}}{2^n} = 1 - 2^{-|a|-|b|} . \end{aligned}$$

Let $\ell := 2k4^k \ln n$, and take ℓ independent copies $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_\ell$ of $\boldsymbol{Y}$. Then the probability that none of them separates a given pair (a,b) is at most

$$\left(1 - 2^{-|a|-|b|}\right)^\ell \leq \left(1 - 2^{-2k}\right)^\ell < e^{-\ell \cdot 2^{-2k}} .$$

Since there are no more than n^{2k} pairs (a,b), the probability that at least one of the pairs (a,b) is left unseparated by all the sets $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_\ell$, is smaller than

$$n^{2k} \cdot e^{-\ell \cdot 2^{-2k}} = n^{2k} \cdot e^{-2k \ln n} = 1 .$$

So, there must exists a sequence $Y_1, \ldots, Y_\ell$ of subsets of $[n]$ such that $D_{n,k}[a,b] = 1$ iff (a,b) is separated by at least one of these sets. Since the set $\{(a,b) \mid a \subseteq Y_i, b \cap Y_i = \emptyset\}$ of all pairs separated by the ith set Y_i corresponds to an all-1 submatrix of $D_{n,k}$, this implies $\mathrm{Cov}(D_{n,k}) \leq \ell$, as desired. □

Claim 7.18 together with bounds (7.5) and (7.6) implies

COROLLARY 7.19. *Let $A = D_{n,k}$ with $k = \log_2 n$. Then both $NC(A)$ and $NC(\overline{A})$ are $O(\log n)$, but $C(A) = \Omega(\log^2 n)$.*

Recall that the decomposition number $\mathrm{Dec}(A)$ of a $(0,1)$ matrix is the smallest number of its mutually disjoint monochromatic submatrices covering all entries of A. We already know that $C(A) \geq \log_2 \mathrm{Dec}(A)$. Since both $NC(A)$ and $NC(\overline{A})$ do not exceed $\log_2 \mathrm{Dec}(A)$, Theorem 7.17 implies a partial converse of this inequality.

THEOREM 7.20. *For every $(0,1)$ matrix A,*

$$C(A) \leq 2(\log_2 \mathrm{Dec}(A))^2 .$$

7.2.1.1. *Clique vs. independent set game.* In fact, an even stronger fact holds. In the definition of the decomposition number $\mathrm{Dec}(A)$ we require that all 1-entries *and* all 0-entries of A be decomposed into monochromatic submatrices. But what if we only know that all 1-entries of A can be decomposed in a small number of all-1 submatrices—does then $C(A)$ is small?

THEOREM 7.21. *If the* 1*-entries of a* $(0,1)$ *matrix can be decomposed into* m *mutually disjoint all-*1 *submatrices, then*

$$C(A) = O(\log^2 m)\,.$$

We will derive the theorem from a more general result about the communication complexity of the following "clique versus independent set" *decision* game CIS_G of a given graph G:

- Alice gets a clique $C \subseteq V$ of G.
- Bob gets an independent set $I \subseteq V$ of G.
- Answer "1" iff $C \cap I = \emptyset$.

Note that we always have that $|C \cap I| \leq 1$.

LEMMA 7.22. *For every* n*-vertex graph* G*,* $C(\mathrm{CIS}_G) = O(\log^2 n)$.

PROOF. Given an n-vertex graph $G = (V,E)$ we describe an appropriate communication protocol for the game CIS_G. The protocol works in $\log n$ rounds, and in each round at most $O(\log n)$ bits are communicated. The idea is to do binary search for intersection.

For a subset $U \subseteq V$ of vertices and a vertex $v \in U$, let $d_U(v)$ denote the number of neighbors of v in U. At each stage, the players maintain a subset $U \subseteq V$ of vertices with the following property:

$(*)$ If C and I intersect then the intersection is in U.

Initially $U = V$. In each stage the players do the following:

a. Alice looks for a vertex $u \in C \cap U$ such that $d_U(u) \leq |U|/2$. She sends "0" if no such vertex exists; otherwise she sends "1" followed by the name of u ($\log n$ bits).
b. If Alice sends a name of a vertex u, then Bob checks whether $u \in I$. If so, he gives the answer "0" (there is an intersection), and the game is over. If $u \notin I$ then the players update the current set U by removing from it all non-neighbors of u, and the next round starts. Since $d_U(u) \leq |U|/2$, the size of the new set U is at most half of the old size.
c. If Alice fails to find a needed vertex, then it is Bob's trun. He looks for a vertex $v \in I \cap U$ such that $d_U(v) > |U|/2$. If there is no such vertex v, then he gives the answer "1" (C and I are disjoint), and the game is over. Assuming $(*)$, this is a correct answer, for if there would be a vertex $w \in C \cap I \cap U$ then we would have that $d_U(w) > |U|/2$ (since w does not suited Alice) as well as $d_U(w) \leq |U|/2$ (since w does not suited Bob). If there is a vertex $v \in I \cap U$ such that $d_U(v) > |U|/2$, then Bob sends its name to Alice.
d. Alice checks whether $v \in C$. If so, she gives the answer "1" (there is an intersection, and the game is over. Otherwise, both players update the set U by removing from it all neighbors of v, and the next round starts. Since, $d_U(v) > |U|/2$, the size of the new set U is again at most half of the old size.

Since after each round the size of U is halved, the number of rounds is at most $\log_2 n$, leading to a total communication of $O(\log^2 n)$ bits. It remains, therefore, to ensure that the property $(*)$ is kept through all rounds. Suppose that $C \cap I \neq \emptyset$ and let $w \in C \cap I$ be the unique vertex in their intersection. The vertex w could be lost (removed from U) if Alice sends a vertex u and this vertex is different than w. But since $w \in C$ and C is a clique, w is a neighbor of u, and hence, cannot be removed when updating U (only non-neighbors of u are removed during this update). The only other possibility to "loose" the vertex w is when Bob sends a vertex v and this vertex is different than w. But since $w \in I$ and I is an independent set, w is a non-neighbor of v, and hence, cannot be removed when updating U (only neighbors of v are removed during this update). □

PROOF OF THEOREM 7.21. Let $R_1, \ldots, R_m$ be a decomposition of 1-entries of A into m mutually disjoint all-1 submatrices. Consider the graph G on m vertices $1, \ldots, m$ in which

$$i \text{ and } j \text{ are adjacent iff } R_i \text{ and } R_j \text{ intersect in rows.}$$

Now, given an input (x, y), Alice and Bob transform them to sets

$$X = \{i \mid x \text{ is a row of } R_i\} \quad \text{and} \quad Y = \{i \mid y \text{ is a column of } R_i\}.$$

Note that X is a clique in G. Moreover, since the submatrices $R_1, \ldots, R_m$ are disjoint, Y is an independent set of G. Also, $X \cap Y \neq \emptyset$ iff (x, y) is in a 1-rectangle. Hence, the players can use the protocol for CIS_G. □

The clique vs. independent sets game CIS_G is important in the context of understanding the power of linear programming for **NP**-hard problems. Namely, Yannakakis (1991) has shown that any n-vertex graph G, for which this game requires $\omega(\log n)$ bits of *nondeterminisitic* communication, would give a super-polynomial lower bound for the size of linear programs expressing Vertex Packing and Traveling Salesman Problem polytopes. Note that $NC(\neg\text{CIS}_G) \leq \log_2 n$ for any n-vertex graph: just guess a vertex in the intersection. But for the problem CIS_G itself only graphs G with $NC(\text{CIS}_G) = \Omega(\log n)$ are known.

RESEARCH PROBLEM 7.23. *Exhibit a sequence G_n of n-vertex graphs such that*

$$NC(\text{CIS}_G) = \omega(\log n).$$

7.2.1.2. *Rank upper bound.* There is yet another upper bound, similar in its form to that of Theorem 7.17. Instead of $\text{Cov}(A)$ it uses the following matrix parameter. Say that a square $(0,1)$ matrix $\Delta = (\delta_{ij})$ is *triangular* if $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i > j$. For example, a 3×3 triangular matrix has the form:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & * \\ 1 & * & * \end{bmatrix}.$$

For a $(0,1)$ matrix A, define

$$\Delta(A) = \min\{d \mid A \text{ contains an } d \times d \text{ triangular submatrix}\}.$$

It is clear that $\Delta(A) \leq \min\{\text{rk}(A), \text{Cov}(A)\}$.

THEOREM 7.24. *For every $(0,1)$ matrix A we have that*

$$C(A) \leq (2 + NC(\overline{A})) \cdot (\log_2 \Delta(A)).$$

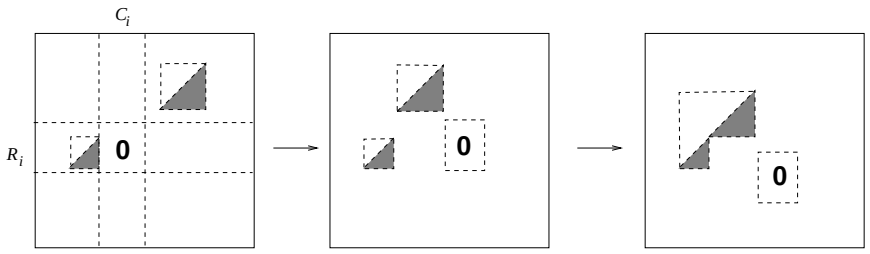


FIGURE 4. Proof of (7.7): Since A_i is an all-0 submatrix, no triangular submatrix of R_i can share a row or a column with a triangle submatrix of C_i. Permute rows and columns of A to "glue" these triangular submatrices into a triangular submatrix of A.

PROOF. Let $r = NC(\overline{A})$, and let $A_1, \ldots, A_{2^r}$ be the all-0 submatrices of A covering all 0's of A. For every matrix A_i, consider the matrix R_i formed by the rows of A intersecting A_i, and C_i be the matrix formed by the columns of A intersecting A_i. Since A_i consists only of 0's, we have that (see Fig. 4 for a proof):

$$\Delta(R_i) + \Delta(C_i) \leq \Delta(A)\,. \tag{7.7}$$

The protocol consists of $\log_2 \Delta(A)$ rounds, in each of which at most $2 + r = 2 + NC(\overline{A})$ bits are communicated.

In each round, the players do the following. First, Alice checks whether there is an index i such that her row intersects A_i and $\Delta(R_i) \leq \frac{1}{2}\Delta(A)$. If yes, then (using $1 + r$ bits) she sends "1" and the index i of this submatrix to Bob. If not, then she sends "0". Now Bob checks whether there is an index i such that his column intersects A_i and $\Delta(C_i) \leq \frac{1}{2}\Delta(A)$. If yes, then (using $1 + r$ bits) he sends "1" and the index i to Alice. If not, then he sends "0".

If either Alice or Bob find a suitable index i in this round then, by communicating at most $2+r$ bits, they have restricted the problem to a matrix A' ($= R_i$ or C_i) for which $\Delta(A') \leq \frac{1}{2}\Delta(A)$. Hence, in this case, the theorem follows by induction.

If both players have sent "0" in this round, then they can finish the protocol: the answer is "$A[x, y] = 1$". Indeed, if there would be a 0 in the intersection of Alice's row and Bob's column, then this 0 would belong to some submatrix A_i. However, for this submatrix we have on the one hand $\Delta(R_i) > \frac{1}{2}\Delta(A)$ (since i did not suit Alice), on the other hand $\Delta(C_i) > \frac{1}{2}\Delta(A)$ since i did not suit Bob. But this contradicts (7.7).

Thus, we have shown that $C(M) \leq (2 + r) \cdot \log_2 \Delta(A)$, as desired. □

Together with Lemma 7.13, Lemma 7.24 implies that nondeterministic communication complexity with a small number k of witnesses cannot be much smaller than the deterministic communication complexity. Define

$$NC_k(A) := \log_2 \mathrm{Cov}_k(A)\,.$$

COROLLARY 7.25. *There is a constant $\varepsilon > 0$ such that, for any $(0,1)$ matrix A, we have*

$$NC_k(A) \geq \varepsilon \sqrt{\frac{C(A)}{k}}\,.$$

PROOF. Since $C(A) = C(\overline{A})$ and $\mathrm{rk}(\overline{A}) \geq \mathrm{rk}(A) - 1$, Lemma 7.24 implies that $C(A)$ is at most about $NC(A) \cdot \log_2 \Delta(\overline{A}) \leq NC_k(A) \cdot \log_2 \Delta(\overline{A})$, and hence, at most about $NC_k(A) \cdot \log_2 \mathrm{rk}(A)$. On the other hand, by Lemma 7.13, we have that $NC_k(A)$ must be at least about $(\log_2 \mathrm{rk}(A))/k$. This implies $\log_2 \mathrm{rk}(A) = O(NC_k(A)/k)$, and hence, the desired lower bound on $NC_k(A)$ follows. □

7.3. Randomized communication

In a *randomized* communication protocol, Alice and Bob are allowed to flip a coin. The coin can be public (seen by both players) or private. Alice and Bob are allowed to get a wrong result with probability smaller than ε. That is, a randomized communication protocol $P(x, y, \mathbf{r})$ using a string $\mathbf{r}$ of random bits is an ε-error protocol for a $(0,1)$ matrix A if, for all entries (x, y),

$$\Pr[P(x, y, \mathbf{r}) \neq A[x, y]] \leq \varepsilon\,.$$

As before, the complexity of the protocol is defined to be the maximal number of bits sent over all inputs.

For a $(0,1)$ matrix A, let $R_\varepsilon(A)$ denote the complexity of the best randomized protocol for A that uses a public random string and errs with probability smaller than ε. If the players must flip their coins privately, then the corresponding measure is denoted by $R_\varepsilon^{\mathrm{privat}}(A)$.

EXAMPLE 7.26. To see the difference between these two measures, let $n = 2^m$ and consider the $n \times n$ identity matrix I_n. Then $C(I_n) = m = \log_2 n$ since $\mathrm{Dec}(I_n) = n$. But randomized protocols can do much better: $R_{1/3}(I_n) = O(1)$. Indeed, the players pick a random string $\mathbf{r} = (r_1, \ldots, r_m)$ in $\{0,1\}^m$. Alice sends the salar product $\langle \mathbf{r}, x\rangle$, Bob checks whether $\langle \mathbf{r}, y\rangle = \langle \mathbf{r}, x\rangle$ and sends the answer. Since every nonzero $(0,1)$ vector $v \neq \mathbf{0}$ is orthogonal over $GF(2)$ to exactly half of all vectors, the error probability is $\varepsilon = 1/2$: juts take $v = x \oplus y$. To get error $\varepsilon < 1/3$, just repeat the protocol two times.

If the random strings $\mathbf{r}$ are *private* (a much more realistic situation), the protocol is less trivial. Still, also in this case it is enough to communicate exponentially fewer bits than in the deterministic case: $R_{1/3}^{\mathrm{privat}}(A) = O(\log \log n)$. Alice picks a random prime number p between 1 and m^2, and sends $4 \log m$ bits encoding p as well as $x \mod p$ to Bob. He checks whether $y \mod p = x \mod p$, and sends the answer to Alice. If $x = y$ the result is always correct. If $x \neq y$ the protocol can err. The protocol errs when Alice picks a prime number p such that p divides $|x - y|$. Since $|x - y| < 2^m$, there are at most $\log_2 2^m = m$ such "bad" primes numbers. On the other hand, the number of prime numbers in the interval $1, \ldots, k$ is at least $k/\ln k$. Hence, Alice is choosing her number p with equal probability from a collection of at least $\Omega(m^2/\ln m^2)$ numbers. Therefore the error probability, that is, the probability to pick one of at most m "bad" primes is $\varepsilon \leq (\ln m^2)/m \to 0$.

A note aside: we have completely ignored a subtle issue on *how* to choose a random prime number. In the communication complexity the players are assumed to be "superior beings:" if an object exists, they can find it immediately—only communication between these "beings" is costly.

We have seen that randomized protocols with private random bits have harder to do. Still, Newman (1991) have proved that any randomized communication protocol with public random bits can be simulated by a protocol with privant random bits at

the cost of relatively small increase of the number of communicated bits: for every boolean $n \times n$ matrix A and for every constant $\varepsilon < 1/4$,

$$R_{2\varepsilon}^{\text{privat}}(A) \le R_\varepsilon(A) + O(\log\log n)\,.$$

Let us now look at how to prove that some matrices are hard for randomized protocols. Let A be a $(0,1)$ matrix with rows X and columns Y. The result of a randomized communication protocol of each input $(x,y) \in X \times Y$ is a *random variable*. To lower bound $R_\varepsilon(A)$ from below, it is often easier to give a lower bound on a "dual" measure. Instead of requiring that, on each input (x,y), the randomized protocol can err with probability at most ε, we now consider *deterministic* protocols and require that they output correct value everywhere except an ε-fraction of inputs (x,y). Or more generally, given a probability measure $\mu : X \times Y \to [0,1]$, we require that the (deterministic) protocol can err on set of inputs of μ-measure at most ε.

Namely, an *ε-error distributional complexity* $D_\varepsilon(A|\mu)$ of a matrix A with respect to a measure μ is the smallest communication complexity of a deterministic protocol P such that

$$\mu\big(\{(x,y) \mid P(x,y) \neq A[x,y]\}\big) \le \varepsilon\,.$$

It was proved by Yao (1979) for uniform μ and generalized by Babai, Frankl and Simon (1986) to arbitrary μ that

$$R_\varepsilon(A) \ge \tfrac{1}{2} D_{2\varepsilon}(A|\mu)$$

for any A, μ and $\varepsilon > 0$.

Consider now the *disjointness matrix* D_n. This is a $2^n \times 2^n$ $(0,1)$ matrix whose rows and columns are labeled by subsets $x \subseteq [n]$, and

$$D_n[x,y] = 1 \quad \text{iff} \quad x \cap y = \emptyset\,.$$

THEOREM 7.27. *For every $\varepsilon > 0$, the ε-error randomized communication complexity of the disjointness matrix is*

$$R_\varepsilon(D_n) = \Omega(\sqrt{n})\,.$$

This was later substantially improved to $R_\varepsilon(D_n) = \Omega(n)$ by Kalyanasundaram and Schnitger (1992); a simpler proof was then found by Razborov (1992a). Still the proof of the weaker bound is more intuitive, and we present it.

PROOF. Let $X = Y$ consist of all subsets of size $\sqrt{n}$ of $[n]$; we assume that n is a perfect square divisible by 12. We concentrate on the submatrix of D_n with row-set X and column-set Y. We shall select the pairs (x,y) at random from the uniform distribution over $X \times Y$. That is, $\mu(x,y) = 1/|X \times Y|$ for $(x,y) \in X \times Y$, and $\mu(x,y) = 0$ for $(x,y) \notin X \times Y$. Since the sets in X and in Y have size $\sqrt{n}$, a random pair (x,y) in $X \times Y$ has probability about $1/e$ to be disjoint: the probability that two random s-element subsets x and y of $[n]$ are disjoint is

$$\binom{n}{s}\binom{n-s}{s}\binom{n}{s}^{-2} = \binom{n-s}{s}\binom{n}{s}^{-1} \approx \left(1 - \frac{s}{n}\right)^s \approx e^{-s^2/n}\,.$$

Take a rectangle $R = F \times G \subseteq X \times Y$ such that $A[x,y] = 1$ (that is, $x \cap y = \emptyset$) for all but an ε fraction of R, that is,

$$|\{(x,y) \in R \mid x \cap y \neq \emptyset\}| \le \varepsilon |R|\,. \tag{7.8}$$

To show that $D_\varepsilon(D_n|\mu) = \Omega(\sqrt{n})$, it is enough to show that

$$|R| \le |X \times Y| \cdot 2^{-c\sqrt{n}} \tag{7.9}$$

for some constant c. We will prove this by showing that either $|F| < |X|2^{-c\sqrt{n}}$ or $|G| < |Y|2^{-c\sqrt{n}}$ (or both) must hold.

Let F_1 be the set of all rows $x \in F$ such that x intersects fewer than $2\varepsilon|G|$ columns $y \in G$. Clearly $|F_1| \geq |F|/2$, for otherwise (7.8) would not hold.

CLAIM 7.28. Given any $x_1, \ldots, x_k \in F_1$, at most $|G|/2$ of the $y \in G$ intersect more than $4\varepsilon k$ of the x_i.

PROOF. If more than a half of the $y \in G$ would intersect more than $4\varepsilon k$ of the x_i, then some x_i would intersect more than $2\varepsilon|G|$ of the $y \in G$, a contradiction with $x_i \in F_1$. □

CLAIM 7.29. If $|F| \geq |X|2^{-c\sqrt{n}}$ then there exists $x_1, \ldots, x_k \in F$ such that $k \geq \sqrt{n}/3$ and for every $p \leq k$,

$$\left| x_p \cap \bigcup_{i<p} x_i \right| < \sqrt{n}/2 .$$

PROOF. Select the $x_i \in F$ inductively. Suppose $x_1, \ldots, x_{p-1}$ have been selected ($p - 1 \leq \sqrt{n}/3$) and let $z = \bigcup_{i<p} x_i$. Then $|z| < p\sqrt{n} \leq n/3$. The number of those $x \in X$ satisfying $|z \cap x| > \sqrt{n}/2$ is therefore less than[2]

$$n \binom{n/3}{\sqrt{n}/2} \binom{2n/3}{\sqrt{n}/2} < \binom{n}{\sqrt{n}} 2^{-c\sqrt{n}} = |X|2^{-c\sqrt{n}},$$

where the inequality follows from the well-known estimates $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} < \left(\frac{en}{k}\right)^k$. Therefore $|x_p \cap z| < \sqrt{n}/2$ for some $x_p \in F_1 - \{x_1, \ldots, x_{p-1}\}$. □

We now turn to the actual proof of (7.9), and hence, of Theorem 7.27. If the condition $|F| \geq |X|2^{-c\sqrt{n}}$ of Claim 7.29 does not hold, we are done. Otherwise, there are at most $\binom{n}{4\varepsilon k}$ ways to select those $4\varepsilon k$ of the x_i which, by Claim 7.28, a given $y \in G$ is allowed to intersect. By Claim 7.29, the union of the remaining x_i (not intersected by y) has size larger than $(k - 4\varepsilon k)\sqrt{n}/2 > k\sqrt{n}/3 \geq n/9$. Therefore

$$|G| < 2 \binom{n}{4\varepsilon k} \binom{n - n/9}{\sqrt{n}} \leq 2^{-c\sqrt{n}} \binom{n}{\sqrt{n}} = 2^{-c\sqrt{n}} |Y| ,$$

and we again conclude that $|R| \leq |X| \cdot |Y| \cdot 2^{-c\sqrt{n}}$. □

7.4. $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ for best-partition games

If $f : \{0,1\}^{2n} \to \{0,1\}$ is a boolean function, then any balanced partition (x, y) of its variables into two blocks of equal size gives us a *communication matrix* M_f of f: this is a $2^n \times 2^n$ $(0,1)$ matrix with $M_f[x, y] = f(x, y)$. The communication complexity of this matrix is then referred to as the communication complexity of f under this (particular) partition. Note however, that different partitions may result in different communication matrices of the same boolean function f. The *best-partition communication complexity* of f is the minimum, over all balanced partitions (x, y), of the communication complexity of M_f under partition (x, y).

For many functions, this (possibility to chose a suitable partition) can drastically reduce the number of communicated bits. For example, the equality function ($f(x, y) = 1$ iff $x_i = y_i$ for all i) has maximal possible communication complexity

[2] Recall that $|x| = \sqrt{n}$ for all $x \in X$.

equal to n (even nondeterministic), if the players are forced to used this "bad" partition (x, y). If, however, Alice receives the first half of x and y, and Bob receives the remaining variables, then they can locally test whether their pieces are equal and tell this the other player. Thus, under this "good" partition, just two bits of communication are enough!

Theorem 7.17 implies that $\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$ in the case of fixed partition games. But what about best-partition complexity? The question is important because it exposes something about the power of lower bound arguments. We can prove a lower bound on the deterministic communication complexity of a function f by arguing about either f or $\neg f$. But if both the function and its negation have low nondeterministic complexity under *some* partitions of variables, other arguments are needed to show that the deterministic communication complexity must be large for *any* partition.

It turns out that *no* analogon of Theorem 7.17 holds in the best-partition case.

THEOREM 7.30. *For the best-partition games, we have that* $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$.

Recall that in the best-partition case the players can choose different (most suitable) partitions for a function f and its negation $\neg f$. To visualize the effect of this choice, we define our function $f(X)$, separating $\mathbf{P}$ from $\mathbf{NP} \cap \mathbf{co\text{-}NP}$, as boolean functions in n^2 variables, arranged into an $n \times n$ matrix. Hence, inputs for f are 0/1 matrices $A : X \to \{0, 1\}$. We define $f(X)$ in such a way that a partition of X according to columns is suitable for computing f, and that according to rows is suitable for $\neg f$.

Say that a row/column of a $(0, 1)$ matrix is *good* if it contains exactly two 1's, and *bad* otherwise. Define $f(X)$ by: $f(A) = 1$ if and only if

a. at least one row of A is good and
b. all columns of A are bad.

Theorem 7.30 is a direct consequence of the following two claims.

CLAIM 7.31. The nondeterministic best-partition communication complexities of both f and $\neg f$ are $O(\log_2 n)$.

PROOF. In the protocol for f Alice takes the first half of *columns* whereas in the protocol for $\neg f$ she takes the first half of *rows*.

To compute $f(A)$ for a given matrix $A : X \to \{0, 1\}$, the protocol first guesses a row r (a candidate for a good row). Then, using 3 bits, Alice tells Bob whether all her columns are bad, and whether the first half of the row r contains none, one, two or more 1's. After that Bob has the whole information about the value $f(A)$ and can announce the answer. The negation $\neg f(A)$ can be computed in the same manner by replacing the roles of rows and columns. □

CLAIM 7.32. The deterministic best-partition communication complexity of f is $\Omega(n)$.

PROOF. The set-disjointness function $DISJ(x, y)$ is a boolean function in $2n$ variables which outputs 1 iff $\sum_{i=1}^{n} x_i y_i = 0$. Since the disjointness matrix has full rank (see Section 3.3.1), the lower bound (7.1) implies that the deterministic communication complexity of $DISJ$, as well as of $\neg DISJ$, under this partition is $\Omega(n)$. (In fact, even nondeterministic and randomized communication complexity of this function is $\Omega(n)$, but we will not need this important fact.)

Take an arbitrary deterministic protocol for $f(X)$. The protocol uses some balanced partition of X into two halves where the first half is seen by Alice and the second by Bob. Say that a column is seen by Alice (resp., by Bob) if Alice (resp., Bob) can see all its entries.

A column is *mixed* if it is seen by none of the two players, that is, if each player can see at least one its entry. Let m be the number of mixed columns. We consider two cases depending on how large this number m is. In both cases we describe a "hard" subset of inputs, i.e. a subset of input matrices on which the players need to communicate many bits.

Case 1: $m < n/2$. In this case each player can see at least one column. Take one column seen by Alice and another column seen by Bob, and let Y be the $(n-3) \times 2$ submatrix of X formed by these two columns without the last three rows. We restrict the protocol to input matrices $A : X \to \{0,1\}$ defined as follows. We first set all entries in the last three rows to 1. This way we ensure that all columns of A are already bad. Then we set all remaining entries of X outside Y to 0. The columns x and y of Y may take arbitrary values. Such a matrix looks like:

$$\begin{bmatrix} x_1 & y_1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ x_{n-4} & y_{n-4} & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}.$$

In each such matrix all columns are bad and, for $n \geq 3$, the last three all-1 rows are also bad. Thus, given such a matrix, the players must determine whether some of the remaining rows is good. Since all these rows have 0's outside the columns x and y, this means that the players must determine whether $x_i = y_i = 1$ for some $1 \leq i < n-3$. That is, they must compute $\neg DISJ(x,y)$ which requires $\Omega(n)$ bits of communication.

Case 2: $m \geq n/2$. Let Y be the $n \times m$ submatrix of Y formed by the mixed columns. Select from the i-th column of Y one entry x_i seen by Alice and one entry y_i seen by Bob. Since $m \leq n$ and we select only $2m$ entries, there must be a row r with $t \leq 2$ selected entries. Let Y be the $n \times (m-t)$ submatrix consisting of the mixed columns with no selected entries in the row r. We may assume that $m-t$ is odd and that $m-t \leq n-2$ (if not, then just include in Y fewer columns).

Now restrict the protocol to input matrices $A : X \to \{0,1\}$ defined as follows. First we set to 1 some two entries of the row r lying outside Y, and set to 0 all the remaining entries of r. This ensures that the obtained matrices will already contain a good row. After that we set all the remaining non-selected entries of X to 0. A typical matrix looks like:

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 & \mathbf{1} & \mathbf{1} & 0 & \dots & 0 \\ x_1 & y_2 & 0 & \dots & x_{n-t} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & x_2 & 0 & \dots & y_{n-t} & 0 & 0 & 0 & \dots & 0 \\ y_1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & y_3 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \\ 0 & 0 & x_3 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix},$$

where r is the first row.

Since each obtained matrix A contains a good row (such is the row r) and all columns outside the submatrix Y are bad (each of them can have a 1 only in the row r),

the players must determine whether all columns of A in Y are also bad. Since all non-selected entries of Y are set to 0, the players must determine whether $x_i + y_i \leq 1$ for all $i = 1, \ldots, m-t$. Hence, the players must decide whether $\sum_{i=1}^{m-t} x_i y_i = 0$, that is, to compute the set-disjointness function $DISJ(x, y)$, which again requires $\Omega(m-t) = \Omega(n)$ bits of communication.

This completes the proof of Claim 7.32, and thus, the proof of Theorem 7.30. □

Exercises

Ex. 7.1 (Threshold matrices). Let A be an $n \times n$ $(0,1)$ matrix whose rows and columns are subsets of $[r] = \{1, \ldots, r\}$, and whose entries are defined by: $A[x, y] = 1$ iff $|x \cap y| \geq k$. Show that then either (i) A contains an all-1 submatrix with at least $n^2/4\binom{r}{k}^2$ entries, or (ii) A contains an all-0 submatrix with at least $n^2/4$ entries.

Hint: Let $\alpha = 1/2\binom{r}{k}$ and call a subset $S \subseteq [r]$ *row-popular* (resp., *column-popular*) if S is contained in at least αn subsets corresponding to rows (resp., to columns) of A. Look at what happens if at least one k-element subset of $[r]$ is both row-popular *and* column popular, at what happens when this is not the case.

Ex. 7.2. For a graph G, let $q(G)$ be the smallest number t with the following property: There is a sequence $S_1, \ldots, S_t$ of subsets of V such that, for every clique $C \subseteq V$ and every independent set $I \subseteq V$ of G such that $C \cap I = \emptyset$, there is an i such that $C \subseteq S_i$ and $I \cap S_i = \emptyset$. Let $NC(\text{CIS}_G)$ be the nondeterministic communication complexity of the "cliqe vs. independent set" game consideren in Section 7.2.1.1. Prove that

$$NC(\text{CIS}_G) = \log_2 q(G).$$

Ex. 7.3. Let A be a $(0,1)$ matrix with rows X and columns Y. Given a probability measure $\mu : X \times Y \to [0,1]$, the *discrepancy* of a submatrix B of A is the absolute value of the difference between the μ-measure of the set of 1-entries and the μ-measure of the set of 0-entries of B. Let $\text{Disc}_\mu(A)$ denote the maximum discrepancy of a submatrix of A.

Prove that matrices of small discrepancy have large distributional, and hence, also randomized communication complexity: for every constant $0 \leq \varepsilon < 1/2$,

$$D_\varepsilon(A|\mu) \geq \log_2 \frac{1-2\varepsilon}{\text{Disc}_\mu(A)}.$$

Hint: Given a deterministic communication protocol $P(x, y)$ for A, achieving $D_\varepsilon(A|\mu)$, estimate the difference

$$\mu(\{(x,y) \mid P(x,y) = A[x,y]\}) - \mu(\{(x,y) \mid P(x,y) \neq A[x,y]\})$$

from above in terms of $\text{Disc}_\mu(A)$.

We now consider the discrepancy $\text{Disc}(A) = \text{Disc}_\mu(A)$ under a uniform distribution assigning each entry the probability 2^{-2n}.

Ex. 7.4. Consider a $2^n \times 2^n$ Sylvester matrix S_n. Its rows and columns are labeled by vectors in $GF(2)^n$, and the entries of S_n are the scalar products of these vectors over $GF(2)$. Show that

$$\text{Disc}(S_n) \leq 2^{-n/2},$$

and hence, that $R_\varepsilon(S_n) = \Omega(n)$ for any constant $\varepsilon > 0$.

Hint: Use Lindsey's Lemma (Lemma 10.25) from Section 10.4.1.

Ex. 7.5. Consider the following *grater than function* $GT_n(x, y)$: Alice gets a non-negative n-bit integer x, Bob gets a nonnegative n-bit integer y, and their goal is to decide whether $x \geq y$.

Show that $R_{1/n}(GT_n) = O(\log^2 n)$.

Hint: Let the players recursively examine segments of their strings until they find the lexicographically first bit in which they differ—this bit determines whether $x \geq y$. Alice can randomly select a prime number $p \leq n^3$, compute $x' \pmod p$ where x' is the first half of x, and send p and $x' \pmod p$ to Bob; this can be done using $O(\log n)$ bits. If $x' \pmod p \neq y' \pmod p$, then x' is different from y', and the players can continue on the first half of their strings. Otherwise the players assume that $x' = y'$, and they continue on the second half of their strings. The players err in this later case when $x' \neq y'$ but $x' \pmod p = y' \pmod p$. Estimate the probability of this error, keeping in mind that there are $\Theta(m/\ln m)$ primes $p \leq m$.

Bibliographic Notes

The concept of communication complexity was invented by Yao (1979, 1981). Theorem 7.5 is due to Nisan and Wigderson (1995). Lemma 7.9 is a modification of a probabilistic argument used by Alon (1986). Lemma 7.13 is due to Grolmusz and Tardos (2003); a slightly weaker bound was earlier proved by Karchmer, Newman, Saks and Wigderson (1994). Lemma 7.22 is due to Yannakakis (1991). Theorem 7.24 is due to Lovász and Saks (1993). Theorem 7.27 is due to Babai, Frankl and Simon (1986). The best-partition communication complexity was introduced by Lipton and Sedgewick (1981). Papadimitriou and Sipser (1984) proved that **NP** $\neq$ **co-NP** in this last model.

CHAPTER 8

Communication and Circuit Depth

We now consider a generalization of communication games which captures the depth of boolean circuits as well as the leafsize of DeMorgan formulas. These games are played on combinatorial rectangles.

8.1. Karchmer–Wigderson games

Recall that an n-dimensional *combinatorial rectangle*, or just a *rectangle*, is a non-empty Cartesian product $S = A \times B$ of two disjoint subsets A and B of vectors in $\{0,1\}^n$. Vector pairs $e = (x, y)$ with $x \neq y$ are referred to as *edges*. The rectangle of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is the rectangle

$$S_f := f^{-1}(0) \times f^{-1}(1).$$

A rectangle S is *monochromatic* if there exists a position $i \in \{1, \ldots, n\}$ such that $x_i \neq y_i$ for all edges $(x, y) \in S$; in this case we say that i is a *separating position* of S.

The communication game on a rectangle $S = A \times B$, introduced by Karchmer and Wigderson (1990), is the following game.

- Alice gets a vector $x \in A$.
- Bob gets a vector $y \in B$.
- The goal is to find a position i such that $x_i \neq y_i$.

These games constitute an extension of the communication games for $(0,1)$ matrices, considered in Section 7.1, to matrices whose entries are *sets*. This time we have an $|A| \times |B|$ matrix M (the *communication matrix* of this game) whose entries are subsets $M[x, y] = \{i \mid x_i \neq y_i\}$ of positions, and the goal of players, on input (x, y), is to find an element in $M[x, y]$.

The game itself can be looked at as a procedure of covering the rectangle S by disjoint monochromatic subrectangles. Recall that a rectangle R is monochromatic if there is a position i such that $x_i \neq y_i$ for all pairs $(x, y) \in R$. (see Section 2.5). As in the case of $(0,1)$ matrices, a *communication protocol* (or a *communication tree*) of a Karchmer–Wigderson game is a binary tree, each inner node of which correspond to a decision made by one of the players at this node. The only difference is that now, instead of submatrices, the nodes are labeled by subrectangles of S so that the following holds:

a. The root is labeled by the whole rectangle S.
b. If a node u is labeled by a rectangle R, then the sons of u are labeled by the corresponding subrectangles S and T of R. Moreover, these subrectangles are obtained from R by splitting the rows of R (if u is Alice's node) or by splitting the columns of R (if u is Bob's node).
c. Leafs are labeled by monochromatic rectangles.

Since at each node, the rows (or columns) of the corresponding submatrix are splitted into *disjoint* parts, the protocol is *deterministic*: each edge $(x, y) \in S$ will reach precisely one leaf. The depth of a tree is the maximum number of edges from the root to a leaf. The minimum depth of a communication tree is the *communication complexity* $\mathrm{cc}(S)$ of the game on the rectangle S.

Recall that the *partition number* $\mathsf{D}(S)$ of a rectangle S is the smallest number t such that S can be decomposed into t disjoint monochromatic rectangles (see Section 2.5). Since each protocol for the game on S gives such a decomposition, we immediately have

PROPOSITION 8.1. *For every rectangle S, $\mathrm{cc}(S) \geq \log_2 \mathsf{D}(S)$.*

To give an example of a communication protocol, let us consider the game on a *parity rectangle* $S = A \times B$, where all vectors in A haven an even and all vectors in B an odd number of 1's.

PROPOSITION 8.2. *For every n-dimensional parity rectangle S, we have*

$$\mathsf{D}(S) \leq 4n^2 .$$

PROOF. We will only show that $\mathsf{D}(S) \leq n^2$ if n is a power of two. The general case then follows by adding redundant zeroes to the strings so that their length is a power of two. The resulting strings will have length at most $2n$, and the upper bound $\mathsf{D}(S_f) \leq 4n^2$ follows.

Consider the communication game for the rectangle S_f. That is, given a pair (x, y) of binary strings of length n such that x has a even and y and odd number of 1's, the goal of Alice and Bob is to find an i with $x_i \neq y_i$.

The basic idea is binary search. Bob begins by saying the parity of the left half of y. Alice then says the parity of the left half of x. If these parities differ, then they continue playing on the left half, otherwise they continue playing on the right half. With each round they halve the size of the playing field, and use two bits of communication. Thus after $\log_2 n$ rounds and $2\log_2 n$ bits of communication they determine an i on which x and y differ. This gives a partition of S_f into $2^{2\log_2 n} = n^2$ disjoint monochromatic rectangles. □

8.2. Games and circuit depth

Let $\mathrm{Depth}(f)$ be the minimum depth of a circuit with AND, OR and NOT gates computing f. The following theorem, which is much in a spirit of Khrapchenko–Rychkov approach (cf. Lemma 1.8) connects communication with computation.

THEOREM 8.3. *For every boolean function f,*

$$\mathrm{Depth}(f) = \mathrm{cc}(S_f) .$$

We prove lower and upper bounds on $\mathrm{Depth}(f)$ separately.

LEMMA 8.4 (Circuit to protocol). $\mathrm{cc}(S_f) \leq \mathrm{Depth}(f)$.

PROOF. We may assume that Alice and Bob have agreed on a circuit of smallest depth computing f. Further, we may assume, using de Morgan's laws, that negations are applied only to the variables. That is, inputs to the circuit are variables and negated variables, and there are only AND and OR gates otherwhere. This does not increase the depth of a circuit.

Now suppose Alice gets an input x such that $f(x) = 0$, and Bob gets an input y such that $f(y) = 1$. In oder to find an i such that $x_i \neq y_i$, the players use the information provided by the underlying circuit. At AND gates speaks Alice, and at OR gates speaks Bob.

Suppose the output gate is an AND gate, that is, we can write $f = f_0 \wedge f_1$. Then Alice sends a bit i corresponding to a function f_i such that $f_i(x) = 0$; if both $f_0(x)$ and $f_1(x)$ output 0, then Alice sends 0. We know that we must have $f_i(y) = 1$. We can then repeat this step at the gate corresponding to the output gate of f_i, where Alice sends a bit if the gate is an AND gate and Bob sends a bit if the gate is an OR gate (he sends a bit corresponding to a function which outputs 1). Alice and Bob repeat this process until they reach a leaf of the circuit. This leaf is labeled by some variable x_i or its negation $\neg x_i$. Hence, $x_i \neq y_i$ implying that i is a correct answer. □

Recall that a boolean function f *separates* a rectangle $S = A \times B$ if $f(x) = 0$ for all $x \in A$, and $f(y) = 1$ for all $y \in B$.

LEMMA 8.5 (Protocol to circuit). *For every rectangle $S = A \times B$ there is a boolean function f such that f separates S and* $\mathrm{Depth}(f) \leq \mathrm{cc}(S)$. *In particular,*

$$\mathrm{Depth}(f) \leq \mathrm{cc}(S_f).$$

PROOF. We prove the lemma by induction on $c = \mathrm{cc}(S)$. Suppose $c = 0$. Then we must have, for some index i, that $x_i \neq y_i$ for all pairs $(x, y) \in S$. Thus we may choose either $f = x_i$ or $f = \neg x_i$ according to which function satisfies $f(A) = 0$ and $f(B) = 1$.

Next, we prove the claim is true for c assuming it is true for $c - 1$. Consider a protocol for the communication game on S that uses at most c bits. Let us assume Alice sends the first bit. Then there is a partition $A = A_0 \cup A_1$, $A_0 \cap A_1 = \emptyset$, such that for $x \in A_0$, Alice sends the bit 0 and for $x \in A_1$, Alice sends the bit 1. After that we are left with two disjoint rectangles $A_0 \times B$ and $A_1 \times B$ whose communication complexity is at most $c - 1$. Applying our induction hypothesis, we find there exists a function f_0 such that

$$f_0(A_0) = 0, \quad f_0(B) = 1 \quad \text{and} \quad \mathrm{Depth}(f_0) \leq c - 1,$$

and there exists a function f_1 such that

$$f_1(A_1) = 0, \quad f_1(B) = 1 \quad \text{and} \quad \mathrm{Depth}(f_1) \leq c - 1.$$

We define $f = f_0 \wedge f_1$. Then $f(A) = 0, f(B) = 1$, and

$$\mathrm{Depth}(f) \leq 1 + \max\{\mathrm{Depth}(f_0), \mathrm{Depth}(f_1)\} \leq c$$

as desired. Note that, if Bob had sent the first bit, we would have partitioned B and defined $f = f_0 \vee f_1$. □

In a *monotone version* of Karchmer–Wigderson game on a rectangle S, given an input pair $(x, y) \in S$, the players must find an i such that $x_i = 0$ and $y_i = 1$. In general, this game may be not well defined: if x is the all-1 vector and y is the all-0 vector, then no valid answer exists. However, if $S \subseteq S_f$ for a *monotone* boolean function f, then every pair (x, y) has a valid answer.

In the case of monotone boolean functions f the game can be described as a search for an element in the intersection of 0-terms and 1-terms of f. Recall that a 0-term of a monotone boolean function is a set of its variables such that, if we set these variables to 0, the function will output 0 independent of the values of other variables. 0-terms whose no proper subset is a 0-term are called maxterms of f. The concept of 1-terms and minterms are defined dually. Main property of these terms is

their "cross intersection:" if p is a 1-term and q a 0-term of the same boolean function, then $p \cap q \neq \emptyset$. Given a monotone boolean function f, the game is a follows. [1]

- Alice gets a 1-term p of f.
- Bob gets a 0-term q of f.
- The goal is to find a variable in $p \cap q$.

For a monotone boolean function f, let $\mathrm{cc}_+(f)$ be the communication complexity of a monotone version of the Karchmer–Wigderson game on the rectangle S_f. Let also $\mathrm{Depth}_+(f)$ be the minimum depth of a boolean formula with AND and OR gates computing f.

The same argument as in the proof of Theorem 8.3 gives

THEOREM 8.6. *For every monotone boolean function f,*

$$\mathrm{Depth}_+(f) = \mathrm{cc}_+(S_f).$$

For a boolean function, let (as before) S_f denote the rectangle $S_f = f^{-1}(0) \times f^{-1}(1)$. Recall that $L(f)$ denotes the smallest size of a DeMorgan formula computing f. Let $\Gamma(S)$ be the smallest number of leaves in a communication tree for the Karchmer–Wigderson game on the rectangle S.

THEOREM 8.7. *For every boolean function f,*

$$L(f) = \Gamma(S_f).$$

EXERCISE 8.8. Prove this theorem. *Hint*: Argue as in the proof of Theorem 8.3 using the fact that the underlying graph of a formula is a tree.

Recall that the *partition number* $\mathsf{D}(S)$ of a rectangle S is the smallest number t such that S can be decomposed into t disjoint monochromatic rectangles. We already know that $\mathrm{cc}(S) \geq \log_2 \mathsf{D}(S)$ (Proposition 8.1), just because each protocol for the rectangle S produces a decomposition of S into monochromatic rectangles. In the opposite direction we have the following

LEMMA 8.9. *For every rectangle S,*

$$\mathrm{cc}(S) \leq 2(\log_2 \mathsf{D}(S))^2.$$

PROOF. Let $\mathscr{R}$ be an optimal covering of S_f by disjoint monochromatic rectangles. Since each $R \in \mathscr{R}$ is monochromatic, there must be a position $i \in \{1,\ldots,n\}$ such that $x_i \neq y_i$ for all $(x,y) \in R$. Label each R by the *smallest* i with this property. Since all rectangles in $\mathscr{R}$ are disjoint, this is a legal labeling (in the sense of Section 7.2). By Lemma 7.16, for every input (x,y), the players can find out the (unique) rectangle containing (x,y) by communicating at most $2(\log_2 |\mathscr{R}|)^2 = 2(\log_2 \mathsf{D}(S_f))^2$ bits. □

8.3. Games on graphs

Given an n-vertex graph $G = (V,E)$ and an integer $2 \leq k \leq n$, let $\textsc{game}(G,k)$ be the following Karchmer–Wigderson type communication game:

- Alice gets an edge $x \in E$.
- Bob gets an independent set $I \subseteq V$ of size $|I| \leq k$.
- The goal is to find a vertex $v \in x - S$; this vertex must known to both players.

[1] For convenience, we have reversed the roles of players (now Alice gets inputs from $f^{-1}(1)$, not from $f^{-1}(0)$). This does not change anything: since we consider only *deterministic* games, a function and its complement have the same complexity.

Let $c_k(G)$ denote the communication complexity of GAME(G,k).

It is clear that $c_k(G) \le 1+2\log_2 n$ for any k: Alice just sends the codes of both endpoints of her edge. Moreover, $\log_2 n$ bits are also necessary just because both players must know an answer $v \in x - S$.

THEOREM 8.10. *Let $G = (V,E)$ be a triangle-free graph without 4-cycles, and let $k = 2(d-1)$ where d is the maximum degree of its vertices. Then*

$$c_k(G) \ge \log_2 |E| - 1\,.$$

PROOF. The proof is essentially the same as that of Theorem 3.4 in Section 3.2. As in that proof we look at vertices as one-element and edges as two-element sets. For a vertex $y \in V$, let I_y be the set of its neighbors. For an edge $y \in E$, let I_y be the set of all its *proper* neighbors; that is, $v \in I_y$ precisely when $v \notin y$ and v is adjacent with an endpoint of y. Since G has no triangles and no 4-cycles, the sets I_y are independent sets. Moreover, $|I_y| \le 2(d-1)$. Hence, for any two edges $x, y \in E$, the protocol must output a vertex $v \in x - I_y$.

Consider the rectangle $R = R^1 \times R^0$ where $R^1 = E$ is the set of edges and $R^0 = \{I_y \mid y \in E \cup V\}$ is the set of all independent sets defined by edges and vertices of G. A subrectangle $M = M^1 \times M^0$ of R is monochromatic if there is a vertex $v \in V$ such that, for all $x \in M^1$ and $I \in M^0$, we have that $v \in x$ and $v \notin I$.

As in the proof of Theorem 3.4, associate with the rectangle R a $(0,1)$ matrix A whose rows correspond to edges $x \in E$, and columns to independent sets I_y with $y \in V \cup E$. The entries are defined by

$$A[x,y] = \begin{cases} 1 & \text{if } x \cap y \neq \emptyset, \\ 0 & \text{if } x \cap y = \emptyset. \end{cases}$$

We have already shown that this matrix has full column-rank; hence, $\mathrm{rk}(A) = |E|$. On the other hand, if M is a monochromatic subrectangle of R, then we also know (Claim 3.5) that the matrix A_M, obtained from A by setting to 0 all entries outside M, has rank at most 2. The subadditivity of rank therefore implies that we need at least $\mathrm{rk}(A)/2 = |E|/2$ mutually disjoint monochromatic rectangles to cover the whole rectangle R. Hence, at least $\log_2(|E|/2)$ bits must be communicated. □

As we mentioned in Section 3.2, explicit n-vertex graphs G of degree $d = \Theta(\sqrt{n})$ with $\Omega(n^{3/2})$ edges and no 4-cycles, are known; such are, for example, the point-line incidence graphs of projective planes. Taking $k = 2d = \Theta(\sqrt{n})$, Theorem 8.10 yields that $c_{2d}(G) \ge 1.5\log_2 - O(1)$. Most interesting, however, is the case $k = 2$, that is, when Bob gets non-edges.

RESEARCH PROBLEM 8.11. *Find an explicit bipartite $n \times n$ graph G such that*

$$c_2(G) \ge \log_2 n + c \cdot \log_2 \log_2 n \;\; \textit{for} \;\; c > 3\,.$$

By the Magnification Lemma (Lemma 1.12) for graph complexity, this would give us an explicit boolean function f_{2m} in $2m$ variables (with $m = \log_2 n$) such that any DeMorgan formula for f_{2m} must have leafsize $\Omega(m^c)$.

Why the rank argument does not work in case $k = 2$? Just because we don't know what matrix should we associate with the corresponding rectangle. If we take a matrix A_G whose rows correspond to non-edges x and columns to edges y, and define $A_G[x,y] = 1$ iff $x \cap y \neq \emptyset$, then we are already lost because then A_G is a matrix of

scalar products (over the reals) of vectors of length n, implying that $\mathrm{rk}(A_G) \le n$ for any graph G.

Note that the communication game corresponding to the matrix A_G is a special version of the "clique vs. independent" game for the graph G, which we considered in Section 7.2.1.1: edges are cliques (of size 2) and non-edges are independent sets (of size 2).

An interesting question is: how the communication complexity $C(A_G)$ the matrix A_G (a decision problem) is related to the communication complexity $c_2(G)$ of the edge-nonedge game on G (a search problem)?

It is not difficult to see that $C(A_G) \le c_2(G)+\log_2 n+1$. Indeed, having an endpoint $v \in x$ of her edge $y = \{u, v\}$ such that $v \notin y$, Alice can just send the binary code of the other endpoint u of her edge (using $\log_2 n$ bits) to Bob, and he just replies (using one bit) whether u is in his non-edge x. Much more interesting is the other direction: how, knowing that $x \cap y = \emptyset$ or not, to determine a vertex $v \in x - y$ using fewer than $2\log_2 n$ bits of communication?

If $x \cap y = \emptyset$, then Alice can just announce (using $\log_2 n$ bits) any one of the endpoints of her edge x, and the game is over. But what if $x \cap y \neq \emptyset$? It seems like then the original task of the players has not been made simpler: they must still determine the unique vertex, the endpoint of Alice's edge which is not present in Bob's non-edge. Kushilevitz and Weinreb (2009) have recently shown that this is not true: knowing a communication protocol for the matrix A_G one can design a protocol for $\textsc{game}(G, 2)$ which uses *fewer* than $2\log_2 n$ bits of communication. Namely, for every graph G,

$$c_2(G) \le 0.886 \cdot C(A_G) + \log_2 n + O(\log\log n)\,.$$

8.4. A cn lower bound for matching

Let $MATCH_n(x)$ be a monotone boolean function in $\binom{n}{2}$ variables encoding the edges of a graph on $n = 3m$ vertices. The function computes 1 iff the graph contains an m-matching, that is, a set of m vertex disjoint edges.

THEOREM 8.12. *For the function $f = MATCH_n$ we have*

$$\mathrm{Depth}_+(f) = \Omega(n)\,.$$

PROOF. Minterms of this function are m-matchings. What are its 0-terms? If q is a subset of $m-1$ vertices, then the clique c_q on its complement $\overline{q} = [n] - q$ is a 0-term: if we set all edges of that clique to 0 then no m-matching is possible since every such matching p must have at least one edge lying in that clique.

In a monotone version of Karchmer–Wigderson game for $MATCH$, Alice (holding a minterm p) and Bob (holding a maxterm c_q) must find a variable in their intersection (recall that variables correspond to edges, not to vertices). It will be convenient to give Bob not cliques c_q but rather the sets q themselves; then $p \cap c_q \neq \emptyset$ iff some edge of p has no endpoint in q.

Thus, $\mathrm{Depth}_+(f)$ is at least the communication complexity $C(MATCH_n)$ of the following game.

$MATCH_n$: Alice gets an m-matching p and Bob gets an $(m-1)$-element set q of vertices.
Find an edge e such that $e \in p$ and $e \cap q = \emptyset$.

This is a *search* problem: find a desired edge. Our proof strategy is to reduce this problem to a *decision* problem: given two subsets of $[m]$ decide whether they are disjoint. Since this last problem is known to have randomized communication

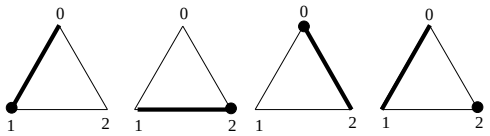


FIGURE 1. $x = (2,0,1,2)$ and $y = (1,2,0,2)$

complexity at least $\Omega(m)$ (see Theorem 7.27 and the comment after it), we will be done. To construct the desired reduction, we consider several intermediate decision problems.

$MATCH'_n$: Alice gets an m-matching p and Bob gets an m-element set q' of vertices. Is there an edge e such that $e \in p$ and $e \cap q' = \emptyset$?

$DIST_m$: Alice gets $x \in \{0,1,2\}^m$ and Bob gets $y \in \{0,1,2\}^m$. Is $x_i \neq y_i$ for all $i = 1,\ldots,n$?

$DISJ_m$: Alice gets $x \in \{0,1\}^m$ and Bob gets $y \in \{0,1\}^m$. Is $x_i \wedge y_i = 0$ for all $i = 1,\ldots,m$?

Through a series of reductions we will show that

$$\Omega(m) \overset{(a)}{=} R_{1/3}(DISJ_m) \overset{(b)}{\leq} R_{1/3}(DIST_m) \overset{(c)}{\leq} R_{1/3}(MATCH'_n) \overset{(d)}{\leq} C(MATCH_n).$$

The lower bound (a) was proved by Kalyanasundaram and Schnitger (1992), and by Razborov (1992a) (see Theorem 7.27 for the proof of a slightly weaker bound).

(b) $R_{1/3}(DISJ_m) \leq R_{1/3}(DIST_m)$. Transform an input $(x,y) \in \{0,1\}^{2m}$ for $DISJ_m$ into an input $(x,y') \in \{0,1,2\}^{2m}$ for $DIST_m$ by setting $y'_i = 1$ if $y_i = 1$, and $y'_i = 2$ if $y_i = 0$. Then $\exists i\ x_i = y_i = 1$ iff $\exists i\ x_i = y'_i$.

(c) $R_{1/3}(DIST_m) \leq R_{1/3}(MATCH'_n)$. Since each randomized protocol for a function f is also a randomized protocol for its negation $\neg f$, it is enough to reduce $DIST_m$ to $\neg MATCH'_n$. For this, it is again enough to encode inputs for $DIST_m$ as inputs for $MATCH'_n$. To do this, split all $n = 3m$ vertices into m vertex-disjoint triples, and number the three vertices in each triple by $0,1,2$. Given a vector $x \in \{0,1,2\}^m$, Alice chooses from the ith triple the edge $e = \{0,1,2\} - \{x_i\}$. Similarly, given a vector $y \in \{0,1,2\}^m$, Bob chooses from the ith triple the vertex y_i. Since the triples are vertex-disjoint, Alice obtains an m-matching p_x, and Bob obtains an m-element set q'_y of vertices. It remains to observe that an edge e with $e \in p_x$ and $e \cap q'_y = \emptyset$ exists iff $x_i = y_i$ for some $i \in [m]$ (see Fig. 1).

(d) $R_{1/3}(MATCH'_n) \leq C(MATCH_n)$. This is the only non-trivial reduction. Let P be a deterministic protocol for $MATCH_n$. We first turn it into a randomized protocol $\widetilde{P}$ for $MATCH_n$ as follows. Alice has an m-matching p and Bob has an $(m-1)$-element set q. The players flip coins publicly and choose a permutation $\pi : [n] \to [n]$ on the set of vertices of the graph. Then they execute the protocol P on $\pi(p)$ and $\pi(q)$. If $e_1,\ldots,e_k \in p$ were the edges in p which do not intersect q, then $\widetilde{P}$ returns each edge from $\{e_1,\ldots,e_k\}$ with equal probability. Note that $k \geq 1$ since $|q| \leq m-1$.

We now construct a randomized protocol P' for $MATCH'_n$ as follows.

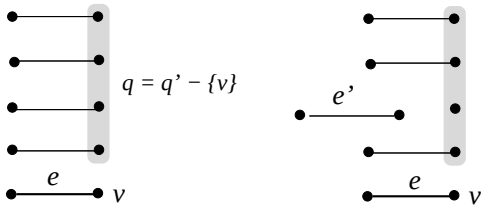


FIGURE 2. In the situation left the gamble is correct: $e \cap q' \neq \emptyset$ for all $e \in p$. In the situation right the gamble is wrong since $e' \cap q' = \emptyset$. But the error probability is then $\leq 1/2$ since, in this case, the protocol $\widetilde{P}$ with *not* choose e with probability at least $1/|\{e, e'\}| = 1/2$.

a Given an m-matching p (for Alice) and an m-element set q' of vertices (for Bob), Bob chooses a random vertex $v \in q'$ and defines $q := q' - \{v\}$.
b Alice and Bob run $\widetilde{P}$ on p and q, and eventually agree on an edge e such that $e \in p$ and $e \cap q = \emptyset$. Bob checks whether $v \in e$ and reports this to Alice.
c If $v \notin e$ then $e \cap q' = \emptyset$, and the players know that the answer is "1". Otherwise they gamble on "0".

It remains therefore to show that the gamble can only be wrong with probability at most $1/2$. Let C be the set of all edges in p that contain no endpoint in q'. The gamble is wrong if $C \neq \emptyset$ and $v \in e$ (see Fig. 2). But the protocol $\widetilde{P}$ outputs each edge in $C \cup \{e\}$ with the same probability $p = 1/|C \cup \{e\}| \leq 1/2$. In particular, it will pick the edge e (and not some edge in C) with such a probability. So the probability of error is at most $1/2$. To decrease the error probability, just repeat the protocol P' twice.

This completes the reductions, and thus, the proof of Theorem 8.12. □

Theorem 8.12, together with the monotone version of Spira's theorem (Theorem 2.3) gives an exponential lower bound on the monotone size of DeMorgan formulas. Recall that $f_N = MATCH_n$ is a monotone boolean function in $N = \binom{n}{2} = \Theta(n^2)$ variables.

COROLLARY 8.13.

$$L_+(f_N) = 2^{\Omega(\sqrt{N})}.$$

Borodin et al. (1982) observed that a randomized algorithm for matching, proposed by Lovász (1979b), can be implemented by shallow circuits, that is, $\mathrm{Depth}(f_N) = O(\log^2 N)$. Together with the lower bound $\mathrm{Depth}_+(f_N) = \Omega(\sqrt{N})$ of Theorem 8.12, this gives an exponential gap between the *depth* of monotone and non-monotone circuits, just like Theorem 4.16 gave such a gap for the *size* of circuits.

Yet another consequence of Theorem 8.12 is for switching networks. Such a network is *monotone* if it has no negated variables as contacts.

COROLLARY 8.14. *Every monotone switching network for f_N must have $2^{\Omega(N^{1/4})}$ contacts.*

PROOF. Every switching network with s contacts can be simulated by a DeMorgan circuit of depth $O((\log s)^2)$. We leave this as an exercise. (Hint: binary search.) □

8.5. A $\log^2 n$ lower bound for connectivity

We already know (Proposition 1.1) that switching networks are not weaker than DeMorgan formulas. In this section we will show that *monotone* switching networks can be even exponentially more powerful than monotone formulas.

We consider directed graphs on n vertices with two additional vertices s (the source) and t (the target). The *st-connectivity problem* STCON_n is, given a directed graph on these n vertices with a source node s and a target node t, to determine whether it contains a path from s to t. Hence, this is a boolean function in $\Theta(n^2)$ variables, and is monotone: if we add edges we cannot disconnect an existing path from s to t.

EXERCISE 8.15. Show that STCON_n can be computed by a monotone switching network of size $O(n^2)$. *Hint*: Take a contact for each potential edge.

We will use the communication complexity approach to show that any monotone circuit solving this problem has depth $\Omega((\log_2 n)^2)$, and hence, any monotone DeMorgan formula has super-polynomial leafsize $n^{\Omega(\log n)}$.

We will do this by proving this lower bound on the communication complexity of the corresponding game:

- Alice gets a graph G with s-t path and Bob gets a graph H with no s-t paths. Find an edge which is present in G but is absent in H.

Note that this is a *monotone* game: an edge which is present in H but absent in G is *not* a correct answer. Since we are interested in proving *lower* bounds on the communication complexity of this game, we can restrict our attention to special inputs.

- Game STCON_n: Alice gets a directed path p form s to t and Bob gets a coloring c of vertices by the colors 0 and 1 such that $c(s) = 0$ and $c(t) = 1$. Find an edge $(u, v) \in p$ such that $c(u) = 0$ and $c(v) = 1$.

Note that the path p must have at least one such edge (u, v) because the path p starts in the node s colored 0 and ends in the node t colored 1.

Let $C(\text{STCON}_n)$ denote the communication complexity of this last game. Note that every protocol for the original game can be used to solve this (restricted) game: given a coloring c, Bob converts it into a graph H in which (u, v) is an edge iff $c(u) = c(v)$.

EXERCISE 8.16. Prove that $C(\text{STCON}_m) = O((\log_2 n)^2)$. *Hint*: Use binary search; in fact one of the players may do most of the talking, with the other player communicating only $O(\log_2 n)$ bits overall.

So as it is, the second game is no more "symmetric" since the players receive objects of different types: Alice receives paths and Bob colorings. Still, it is possible to reduce this game to a symmetric one.

Let $n = km$, and assume that the n vertices are partitioned into k levels (layers) $L_1, \ldots, L_k$ with m vertices in each layer. We also have the 0-th layer $L_0 = \{s\}$ containing only the source vertex s, and the $(k+1)$-th layer $L_{k+1} = \{t\}$ containing only the target vertex t. Each s-t path then corresponds to a string in the grid $[m]^k$ consisting of all strings $a = (a_1, \ldots, a_k)$ with $a_i \in [m] = \{1, \ldots, m\}$.

Given two paths (strings) a and b in $[m]^k$, say that $i \in [k]$ is a *fork position* of a, b if either $i = 1$ and $a_1 \neq b_1$, or $i > 0$ and $a_{i-1} = b_{i-1}$ but $a_i \neq b_i$. Note that any two *distinct* strings must have at least one fork position: either they differ in the first coordinate, or there must be a coordinate where they differ "for the first time", that is, the proceeding coordinate is the same in both strings.

We will be interested in the following symmetric games on subsets $S \subseteq [m]^k$.

- Game FORK(S): Alice gets a string $a \in S$ and Bob gets a string $b \in S$. Find a fork position i of a and b, if $a_k \neq b_k$. If $a_k = b_k$ then $i = k+1$ is also a legal answer.

If, for example, $a = (1,2,4,3,4)$ and $b = (3,2,2,5,4)$ then $i = 1$, $i = 3$ and $i = 4$ are legal answers, and players can output any of them.

Let $C(\text{FORK}_{m,k})$ denote the communication complexity of the fork game on the whole set $S = [m]^k$.

We can relate this game to the previous (s-t connectivity) game as follows. When doing this, we restrict our attention to graphs on $n = m \cdot k$ vertices, where only edges, connecting nodes from adjacent levels, are allowed.

LEMMA 8.17. $C(\text{FORK}_{m,k}) \leq C(\text{STCON}_n)$.

PROOF. Suppose we have a protocol Π for STCON_n. We will show that this protocol can be used for the game $\text{FORK}_{m,k}$. To use the protocol Π, the players must convert their inputs $a = (a_1, \ldots, a_k)$ and $b = (b_1, \ldots, b_k)$ (for the fork game) to inputs for the s-t connectivity game.

Alice converts her input $(a_1, \ldots, a_k)$ into a path $p = (u_0, u_1, \ldots, u_k, u_{k+1})$ where $u_0 = s$, $u_{k+1} = t$, and $u_i = a_i$ for $1 \leq i \leq k$. Bob converts his input $(b_1, \ldots, b_k)$ into a coloring c by assigning color 0 to all vertices $s, b_1, \ldots, b_k$, and assigning color 1 to the remaining vertices; hence, $c(s) = 0$ and $c(t) = 1$ (because $t \notin \{s, b_1, \ldots, b_k\}$).

The players now can use the protocol Π for STCON_n to find an edge (u_{i-1}, u_i) in p such that $c(u_{i-1}) = 0$ and $c(u_i) = 1$. This means that u_{i-1} is in the path $(s, b_1, \ldots, b_k)$ and u_i is not. We claim that i is a valid answer for the fork game on the pair a, b.

If $i = 1$ then $u_{i-1} = u_0 = s$ and $u_1 = a_1$. Hence, $c(s) = 0$ and $c(a_1) = 1 \neq 0 = c(b_1)$, implying that $a_1 \neq b_1$ (no vertex can receive two colors).

Let now $1 < i \leq k$. Recall that c assigns color 0 to exactly one vertex in each layer L_i, namely, to the vertex b_i. Hence, the fact that $c(a_{i-1}) = c(u_{i-1}) = 0$ means that $a_{i-1} = b_{i-1}$, and the fact that $c(a_i) = c(u_i) = 1 \neq 0 = c(b_i)$ means that $a_i \neq b_i$.

Finally, let $i = k+1$. Then $u_{i-1} = a_k$ and $u_i = t$. Since $c(a_k) = c(u_{i-1}) = 0$ and since only the vertex b_k on the kth layer can receive color 0, this implies $a_k = b_k$. Since, in this case, $i = k+1$ is a legal answer for the form game, we are done. □

By Lemma 8.17 and Exercise 8.16 we know that the communication complexity of the fork game on $[m]^k$ is at most about $(\log_2(km))^2$. We will show that this upper bound is almost optimal.

THEOREM 8.18.

$$C(\text{FORK}_{m,k}) = \Omega((\log_2 m) \cdot (\log_2 k)).$$

PROOF. Call a two-player protocol an (α, k)-*protocol* if it is a protocol for the game FORK(S) on some subset $S \subseteq [m]^k$ such that $|S| \geq \alpha m^k$. Denote by $C(\alpha, k)$ the minimum communication complexity of an (α, k)-protocol. That is, if $C(\alpha, k) = d$ then there exists a subset $S \subseteq [m]^k$ of $|S| \geq \alpha m^k$ strings and a protocol Π of communication complexity d such that Π works correctly on S. In particular, $C(1, k) = C(\text{FORK}_{m,k})$.

We start with two simple claims.

CLAIM 8.19. For any $k \geq 1$ and any $\alpha > 1/m$, $C(\alpha, k) > 0$.

PROOF. Suppose that $C(\alpha, k) = 0$. Thus, there exists a subset of strings $S \subseteq [m]^k$ such that $|S| \geq \alpha m^k > m^{k-1}$ and the players must know the unique answer $i \in \{1, \ldots, k, k+1\}$ for all input pairs $a, b \in S$ without any communication. Since $|S|$

is strictly larger than m^{k-1}, there must be two strings $a, b \in S$ with $a_k \neq b_k$. Hence, on this input pair (a, b) the answer must be some $i \leq k$. But on input pair (a, a) the only legal answer is $i = k+1$, a contradiction. □

CLAIM 8.20. For every $k \geq 1$ and α, if $C(\alpha, k) > 0$ then $C(\alpha, k) \geq C(\alpha/2, k) + 1$.

PROOF. Let $d = C(\alpha, k)$. Thus, there exists a subset $S \subseteq [m]^k$ such that $|S| \geq \alpha m^k$ and there is a protocol Π such that for all $a, b \in S$, the protocol correctly solves the game on these inputs. Assume w.l.o.g. that Alice speaks first (the case when Bob speaks first is similar). Hence Alice sends either 0 or 1. After this (first) bit is communicated, the set S is splitted into two parts S_0 and S_1. Assume w.l.o.g. that $|S_0| \geq |S_1|$. Let Π_0 be the rest of the protocol Π, after assuming that the first bit send by Alice was 0. That is, Π_0 works exactly like Π, but without sending the first bit, and continuing as if the value of the first bit was 0. The communication complexity of Π_0 is at most $d-1$. Obviously, Π_0 must work correctly on S_0, because Π does this on the whole set $S = S_0 \cup S_1$. Hence, Π_0 is an $(\alpha/2, k)$-protocol. Thus, $C(\alpha/2, k) \leq d - 1 = C(\alpha, k)$. □

Starting with $\alpha = 1$ and applying Claim 8.20 $t = (\log_2 m)/2$ times, we obtain that $C(1, k) \geq C(\alpha, k) + t$ with $\alpha = 1/\sqrt{m}$. Since $\alpha > 1/m$, Claim 8.19 yields $C(\text{FORK}_{m,k}) = C(1, k) = \Omega(\log_2 m)$. This lower bound is, however, too weak. What we claim in Theorem 8.18 is that the actual lower bound is about $\log k$ times larger.

The reason why Claims 8.19 and 8.20 alone cannot yield larger lower bounds is that, when compared to the whole universum $[m]^k$, the density of the sets S (on which a protocol is still correct) drops down very fast. In such situations it is usually helpful to take a projection $S\restriction_I$ of S onto some subset $I \subseteq [k]$ and work in smaller universum $[m]^I$. A hope is that then the relative density of $S\restriction_I$ within $[m]^I$ will be much larger than that of S within the whole universum $[m]^k$. This trick is usually called the "amplification" of density.

We now turn to an amplification step: given an (α, k)-protocol (with $k \geq 2$ and α not too small), we convert it to a $(\sqrt{\alpha}/2, k/2)$-protocol. Thus α may be amplified greatly[2] while k is cut in half. By amplifying α after every about $\log_2 k$ steps, we may keep $\alpha > 1/m$ until k reaches 1, showing the protocol must have a path of length at least $\log_2 m$ times $\log_2 k$.

LEMMA 8.21 (Increasing Density). *For every $k > 0$ and $\alpha \geq 16/m$,*

$$C(\alpha, k) \geq C(\sqrt{\alpha}/2,\, k/2)\,.$$

PROOF. We are given an (α, k)-protocol working correctly on some set $S \subseteq [m]^k$ of $|S| \geq \alpha m^k$ strings (paths). Consider a bipartite graph $G = (U, V, S)$ with parts U and V where U consists of all $m^{k/2}$ possible strings on the first $k/2$ levels, and V consists of all $m^{k/2}$ possible strings on the last $k/2$ levels. We connect $u \in U$ and $v \in V$ if their concatenation $u \circ v$ is a string in S; in this case we say that v is an *extension* of u. Hence, G is a bipartite graph with parts of size $m^{k/2}$ and $|S| \geq \alpha m^k = \alpha |U \times V|$ edges.

We need the following combinatorial fact about dense matrices. Let A be an $M \times N$ $(0,1)$ matrix. We say that A is *α-dense* if at least an α-fraction of all its entries are 1's. Similarly, a row (or column) is *α-dense* if at least an α-fraction of all its entries are 1's.

CLAIM 8.22. If A is 2α-dense then either: (a) there exists a row which is $\sqrt{\alpha}$-dense, or (b) at least a fraction $\sqrt{\alpha}$ of the rows are α-dense.

[2]Note that $\sqrt{\alpha}$ is at least twice larger than α, if $\alpha \leq 1/4$.

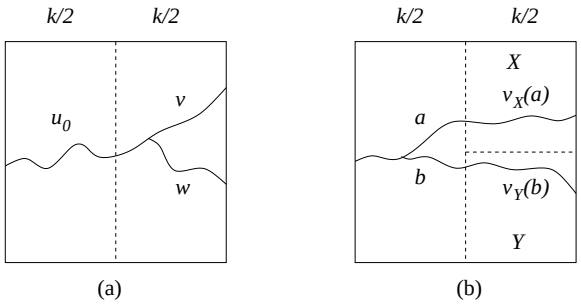


FIGURE 3. Two cases for constructing a protocol for strings of length $k/2$.

PROOF. Suppose that the two cases do not hold. We calculate the density of the entire matrix. Since (b) does not hold, less than $\sqrt{\alpha}M$ of the rows are α-dense. Since (a) does not hold, each of these rows has less than $\sqrt{\alpha}N$ 1's; hence, the fraction of 1's in α-dense rows is strictly less than $(\sqrt{\alpha})(\sqrt{\alpha}) = \alpha$. We have at most M rows which are not α-dense, and each of them has less than αN 1's. Hence, the fraction of 1's in these rows is also less than α. Thus, the total fraction of 1's in the matrix is less than 2α, a contradiction with the 2α-density of A. □

By Claim 8.22, when applied to the adjacency matrix

$$A[u,v] = \begin{cases} 1 & \text{if } uv \in S; \\ 0 & \text{if } uv \notin S, \end{cases}$$

of our bipartite graph $G = (U, V, S)$, at least one of the following two must hold:

(a) There is an $u_0 \in U$ with

$$\left|\{v \in V \mid u_0 \circ v \in S\}\right| \geq \sqrt{\tfrac{\alpha}{2}}|V| = \sqrt{\tfrac{\alpha}{2}}m^{k/2}.$$

(b) There is an $S' \subseteq U$ such that $|S'| \geq \sqrt{\frac{\alpha}{2}}m^{k/2}$ and

$$\left|\{v \in V \mid u \circ v \in S\}\right| \geq \tfrac{\alpha}{2}|V| = \tfrac{\alpha}{2}m^{k/2} \quad \text{for all} \quad u \in S'.$$

In both cases (a) and (b), we show how to construct a $(\sqrt{\alpha}/2, k/2)$-protocol (see Fig. 3).

Case (a): In this case, we have one string u_0 on the left that has many extensions v on the right such that $u_0 \circ v \in S$. Thus we can recover a $(\sqrt{\alpha/2}, k/2)$-protocol as follows: let S' be the set of all extensions of u_0. Given two strings $v, w \in S'$, the players can play the S'-game on these inputs by following the S-protocol for the pair of strings $u_0 \circ v$ and $u_0 \circ w$. Since these strings are identical on the first $k/2$ coordinates, the answer i must correspond to a point where (the paths corresponding to) v and w diverge.

Case (b): In this case, we take a random partition of the $km/2$ nodes in the right $k/2$ levels. More precisely, take $m/2$ nodes at random from each of the right $k/2$ levels, and call their union X; call the set of remaining $km/4$ right nodes Y. Say that a string $u \in U$ is *good* if it has an extension $v_X(u)$ lying entirely in X and another extension $v_Y(u)$ lying entirely in Y.

CLAIM 8.23. The expected number of good strings in S' is at least $0.9|S'|$.

PROOF. We can construct a subset of X as follows. Take $m/2$ uniformly distributed paths $p_1, \ldots, p_{m/2}$ of the right $k/2$ layers, color their vertices red and let X be the union of these vertices. The paths are not necessarily vertex disjoint and some layers may have fewer than $m/2$ vertices. To correct the situation, we randomly color additional vertices red in each layer to ensure that all layers have exactly $m/2$ red vertices. Finally, we color all remaining vertices blue.

Take now a path $u \in S'$. By (b) we know that each red path p_i is an extension of u with probability at least $\alpha/2$. That is, p_i is *not* and extension of u with probability at most $1 - \alpha/2$. Since $\alpha \geq 12/m$, the union bound implies that the probability that none of $m/2$ red paths is an extension of u does not exceed

$$(1 - \alpha/2)^{m/2} \leq (1 - 6/m)^{m/2} \leq e^{-3} < 0.05\,.$$

Since the red and blue vertices are identically distributed, the same also holds for blue paths. Therefore, each $u \in S'$ is good with probability at least $1 - 2 \cdot 0,05 = 0.9$, implying that the expected fraction of good strings in S' is at least 0.9. □

This yields a $(\sqrt{\alpha}/2, k/2)$-protocol as follows. Let $S'' \subseteq S'$ be the set of all good strings in S'. By Claim 8.23 and since $0.9/\sqrt{2} > 0.5$, the density of the set S'' within $[m]^{k/2}$ is at least $0.9\sqrt{\alpha/2} \geq \sqrt{\alpha}/2$, as desired. Given strings $a, b \in S''$, the players follow the S-protocol on the inputs $a \circ v_X(a)$ and $b \circ v_Y(b)$. Since the S-protocol is correct on these strings, and since they share no vertices in the right $k/2$ levels, the protocol must return an answer i in the first $k/2$ levels, hence the answer is in fact valid for a and b.

This completes the proof of Lemma 8.21. □

Now we can finish the proof of Theorem 8.18 as follows.

By $r = \lfloor \log_2(\sqrt{m}/8) \rfloor$ applications of Claim 8.20 and one application of Lemma 8.21, we obtain that

$$C(2/\sqrt{m}, k) \geq C(16/m, k) + r \geq C(2/\sqrt{m}, k/2) + r\,.$$

Applying the last inequality $s = \lfloor \log_2 k \rfloor$ times, we obtain

$$C(2/\sqrt{m}, k) \geq C(2/\sqrt{m}, 1) + r \cdot s \geq r \cdot s\,.$$

Hence,

$$C(\text{FORK}_{m,k}) = C(1, k) \geq C(2/\sqrt{m}, k) \geq r \cdot s = \Omega((\log_2 m) \cdot (\log_2 k))\,. \qquad \square$$

Eercises

EX. 8.1. The game FORMULA is a game of two players Up (upper) and Lo (lower), Up will try to prove an upper bound for the formula size of a boolean function; Lo will try to interfere him. A *position* in this game is a triplet (U, V, t) where $U, V \subseteq \{0.1\}^n$, $U \cap V = \emptyset$ and $t \geq 1$ is an integer. Up begins the game. He obtains a position (U, V, t), chooses one of the two sets U, V (say, U), somehow represents U and t in the form

$$U = U' \cup U'' \quad t = t' + t'' \quad (t', t'' \geq 1)$$

and hands to Lo the two positions (U', V, t') and (U'', V, t''). If Up chooses the set V, the description of his actions is given in the analogous way.

Lo chooses one of the two positions offered to him and returns it to Up (the remaining position is thrown out). Then Up moves as above (in the new position) and so on. The game is over when Up receives a position of the form $(U^*, V^*, 1)$. Up wins

if $U^* \times V^*$ forms a monochromatic rectangle, that is, if there is an $i \in [n]$ such that $x_i \neq y_i$ for all $x \in U^*$ and $y \in V^*$.

Prove that Up has a winning strategy in a position (U, V, t) iff there exists a boolean function $f : \{0,1\}^n \to \{0,1\}$ such that: $f(U) = 0$, $f(V) = 1$ and f has a DeMorgen formula of leafsize $\leq t$.

Hint: Argue by induction on t as in the proof of Theorem 8.3.

Ex. 8.2. Say that a string $x \in [m]^k$ is a *limit* for a subset $S \subseteq [m]^k$ of strings if $x \in S$ and for every position $i = 1, \ldots, k$ there is a string $y \in S$ such that $x \neq y$ and $x_i = y_i$.

Prove: If $S \subseteq [m]^k$ and $|S| > km$ then S has a limit for itself.

Hint: What does it mean that *S does not* have a limit for itself?

Ex. 8.3. One somewhat artificially looking thing in the definition of the fork game is that the players need not necessarily output a fork position, even when $a \neq b$ (note that then at least one fork position must exist). Instead, they are also allowed to answer "$k+1$", if $a_k = b_k$. It makes therefore sense to look at what happens if we consider the following modified fork game:

a. Alice gets a string $a \in S$ and Bob gets a string $b \in S$.
b. Find a fork position i of a and b, if there is one.

That is, the only difference from the original fork game is that, if an input pair $a \neq b$ coincides in the last position (i.e., $a_k = b_k$) $i = k+1$ was allowed as a legal answer whereas in the modified game they must output some other position $i \leq k$ (such a fork position exists since $a \neq b$).

Prove: the modified fork game on $[m]^k$ has communication complexity $\Omega(k \cdot \log m)$.

Hint: Assume that d bits of communication are enough, where $2^d < m^k/(km)$. Use the previus exercise to get a contradiction.

Ex. 8.4. Let G be a bipartite $n \times n$ graph, and consider the following "edge-nonedge" game on it:

a. Alice gets an edge x of G.
b. Bob gets a non-edge y of G (a pair y of two nonadjacent vertices).
c. Find a vertex $v \in x - y$.

Let (as before) $c_2(G)$ be the communication complexity of this game. Let also A be the adjacency matrix of G. Prove that

$$c_2(G) \leq \log_2 \mathrm{Cov}(A) + \log_2 n + 1 .$$

Hint: Alice can tell in which of the all-1 submatrices of A her edge lies.

Ex. 8.5. Given a graph $G = (V, E)$, consider now a *decision* version of the "edge-nonedge" game, where the playerst must give answer "1" iff $x \cap y = \emptyset$. Let $nc_2(G)$ be the *nondeterministic* communication complexity of this game.

Let $q(G)$ be the smallest number t with the following property: There is a sequence $S_1, \ldots, S_t$ of subsets of V such that, for every edge x and every non-edge y of G, there is an i such that $x \subseteq S_i$ and $y \cap S_i = \emptyset$.

Prove that $nc_2(G) = \log_2 q(G)$.

Bibliographic Notes

The relation between communication complexity and circuit depth was discovered and Theorem 8.3 was proved by Karchmer and Wigderson (1990). Theorem 8.12 is

due to Raz and Wigderson (1992). Theorem 8.18 is due to Grigni and Sipser (1991). The yet another game-theoretic formulation of formula size in Exercise 8.1 is due to Razborov (1990).

CHAPTER 9

Many Players

A general scenario of a k-party communication is as follows. We have some function $f(x)$ whose input x is splitted into k equal sized parts $x = (x_1, \ldots, x_k)$. There are k players who wish to collaboratively evaluate a given function f on every input x. Each player has unlimited computational power and full knowledge of the function. As in the case of two players, the players are not adversaries—they help and trust each other. Depending on what parts of the input x each player can see, there are two main models of communication:

a. In the "number in the hand" model, the ith player can only see x_i.
b. In the "number on the forehead" model, the ith player can see all the x_j *except* x_i.

Note that for $k = 2$ (two players) there is no difference between these two models. The difference comes when we have $k \geq 3$ players. In this case the second model seems to be (and actually is) more difficult to analyze because players share some common information. For example, the first two players *both* can see all inputs $x_3, \ldots, x_k$.

Players can communicate by writing bits 0 and 1 on a blackboard. The blackboard is seen by all players. The game starts with the empty blackboard. For each string on the blackboard, the protocol either gives the value of the output (in that case the protocol is over), or specifies which player writes the next bit and what that bit should be as a function of the inputs this player knows (and the string on the board). During the computation on one input the blackboard is never erased, players simply append their messages. The objective is to compute the function with as small amount of communication as possible.

The *communication complexity* of a k-party game for f is the minimal number c such that on every input $x \in X$ the players can decide whether $f(x) = 1$ or not, by writing at most c bits on the blackboard. Put otherwise, the communication complexity is the minimal number of bits written on the blackboard on the worst-case input.

Note a big difference between the two models of communication. If the number k of players increases, the communication complexity in the "number in hand" model can only increase (the pieces of input each player can see is smaller and smaller), whereas it can only decrease in the "number on the forehead" (the pieces of seen input are larger and larger). This is why the first model deserved much less attention. Still, the model becomes interesting if instead of computing a given function f exactly, the players are only required to *approximate* its values.

9.1. The "number in the hand" model

Consider the following *approximate disjointness* problem $Disj_n$. Each X_i consists of all subsets a_i of $[n] = \{1, \ldots, n\}$. Given a sequence $a = (a_1, \ldots, a_k)$ the k players are required to distinguish between the following two extreme cases:

- Answer "input is positive" if $\bigcap_{i=1}^{k} a_i \neq \emptyset$.
- Answer "input is negative" if $a_i \cap a_j = \emptyset$ for all $i \neq j$.
- If neither of these two events happens, then any answer is legal.

LEMMA 9.1. *In the "bit in hand" model, the approximate disjointness problem $Disj_n$ requires $\Omega(n/k)$ bits of communication.*

PROOF. First note that any c-bit communication protocol for the approximate disjointness problem partitions the space of inputs into at most 2^c "boxes", where a box is a Cartesian product $S_1 \times S_2 \times \cdots \times S_k$ with $S_i \subseteq 2^{[n]}$ for each i. Each box must be labeled with an answer, and thus the boxes must be "monochromatic" in the following sense: no box can contain both a positive instance and a negative instance. (There are no restrictions on instances that are neither negative nor positive.)

We will show that there are exactly $(k+1)^n$ positive instances, but any box that does not contain a negative instance can contain at most k^n positive instances. It then follows that there must be at least

$$(k+1)^n/k^n = (1+1/k)^n \approx e^{n/k}$$

boxes to cover all positive instances and thus the number of bits communicated must be at least the logarithm $\Omega(n/k)$ of this number, giving the desired lower bound.

To count the number of positive instances, note that any partition of the n items in $[n]$ between k players, leaving some items "unlocated", corresponds to a mapping $g : [n] \to [k+1]$, implying that the number of positive instances is exactly $(k+1)^n$.

Now consider a box $S = S_1 \times S_2 \times \cdots \times S_k$ that does not contain any negative instance. Note that for each item $x \in [n]$ there must be a player $i = i_x$ such that $x \notin a$ for all $a \in S_i$. This holds because otherwise there would be, in each S_i, a set $a_i \in S_i$ containing x, and we would have that $\bigcap_{i=1}^{k} a_i \supseteq \{x\} \neq \emptyset$—a negative instance in the box S.

We can now obtain an upper bound on the number of positive instances in S by noting that any such instance corresponds to a partition of the n items among k players and "unlocated", but now with an additional restriction that each item $x \in [n]$ can not be in the block given to the i_x-th player. Thus each item has only k possible locations for it and the number of such partitions is at most n^k. □

9.1.1. Approximate set packing problem. The *set packing* problem is, given a collection A of subsets of $[n] = \{1, \ldots, n\}$, to find the largest packing—that is, largest collection of pairwise disjoint sets. The *packing number* of A, is the largest number of sets of A in a packing of $[n]$.

The *set packing* communication problem is as follows: we have k players each holding a collection A_i of subsets of $[n]$, and the players are looking for the largest packing in the union $A = A_1 \cup \cdots \cup A_k$ of their collections. The goal of players is to approximate the packing number of A to within a given multiplicative factor λ.

PROPOSITION 9.2. *There is a k-player protocol approximating the packing number within a factor of $\lambda = \min\{k, \sqrt{n}\}$ and using $O(kn^2)$ bits of communication.*

PROOF. Getting an approximation factor k is easy by just picking the single player with the largest packing in her collection. If $k > \sqrt{n}$, we can do better by using the following simple greedy protocol: at each stage each player announces the smallest set $a_i \in A_i$ that is disjoint from all previously chosen sets; this requires n bits of communication from each of k players. The smallest such set is chosen to be in the packing.

This is repeated until no more disjoint sets exist; hence, the protocol ends after at most n stages. It remains to verify that this packing is by at most a factor of $\sqrt{n}$ smaller than the number of sets in an optimal packing.

Let $a_1, \ldots, a_t$ be the sets in A chosen by out protocol. The sets

$$B_i = \{a \in A \mid a \cap a_i \neq \emptyset\}$$

form a partition of A. Moreover, since all sets in B_i contain an element of a_i, the maximum number of disjoint sets in B_i is at most the cardinality of a_i. On the other hand, every set in B_i is of size at least $|a_i|$, so the maximum number of disjoint sets in B_i is also at most $\lfloor n/|a_i| \rfloor$. Thus, the optimal solution can contain at most

$$\min\{|a_i|, \lfloor n/|a_i| \rfloor\} \leq \max_{x \in \mathbb{N}} \min\{x, \lfloor n/x \rfloor\} = \lfloor \sqrt{n} \rfloor$$

sets from each B_i. □

On the other hand we have the following lower bound.

THEOREM 9.3. *Any k-player protocol for approximating the packing number to within a factor less than k requires $2^{\Omega(n/k^2)}$ bits of communication.*

In particular, as long as $k \leq n^{1/2-\varepsilon}$ for $\varepsilon > 0$, the communication complexity is exponential in n.

PROOF. We have k players, each holding a collection A_i of subsets of $[n]$. It is enough to prove a lower bound on the communication complexity needed in order to distinguish between the case where the packing number is 1 and the case where it is k. That is, to distinguish the case where there exist k disjoint sets $a_i \in A_i$, and the case where any two sets $a_i \in A_i$ and $a_j \in A_j$ intersect (packing number is 1).

Suppose now that ℓ bits of communication are enough to distinguish these two cases. We will show that then the approximate disjointness problem $Disj_N$ for $N = e^{\Omega(n/k^2)}$ can be also solved using at most ℓ bits of communication. Together with Lemma 9.1 this will immediately yield the desired lower bound $\ell = \Omega(N/k)$

The reduction uses a set of partitions $\mathscr{A} = \{a^s \mid s = 1, \ldots, N\}$, where each a^s is a partition $a^s = (a^s_1, \ldots, a^s_k)$ of $[n]$ into k disjoint blocks. Say that such a set $\mathscr{A}$ of partitions is *cross-intersecting* if

$$a^{s_i}_i \cap a^{s_j}_j \neq \emptyset \quad \text{for all} \quad 1 \leq i \neq j \leq k \quad \text{and} \quad 1 \leq s_i \neq s_j \leq N\,,$$

that is, if different blocks from different partitions have non-empty intersection.

CLAIM 9.4. A cross-intersecting set of $N = e^{n/(2k^2)}/k$ partitions exists.

PROOF. Let $f_1, \ldots, f_N$ be independent copies of a random function $f : [n] \to [k]$ where $\Pr[f(x) = i] = 1/k$ for every $x \in [n]$ and $i \in [k]$. Each function f_s gives us a partition $a^s = (a^s_1, \ldots, a^s_k)$ of $[n]$ with $a^s_i = \{x \mid f_s(x) = i\}$. Now fix $1 \leq i \neq j \leq k$ and two indices of partitions $1 \leq s_i \neq s_j \leq N$. For every fixed $x \in [n]$, the probability that $f_{s_i}(x) \neq i$ or $f_{s_j}(x) \neq j$ is $1 - 1/k^2$. Since $a^{s_i}_i \cap a^{s_j}_j = \emptyset$ holds iff this happens for all n elements x, we obtain that

$$\Pr[a^{s_i}_i \cap a^{s_j}_j = \emptyset] = (1 - 1/k^2)^n < e^{-n/k^2}\,.$$

Since there are at most k^2N^2 such choices of indices, we get that the desired set of partitions exist, as long as $k^2N^2 \leq e^{n/k^2}$. □

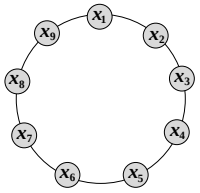


FIGURE 1. $k = 9$ players, the ith one with x_i with on his/her forehead.

We now describe the reduction of the approximate disjointness problem $Disj_N$ to the problem of distinguishing whether the packing number is 1 of k. Player i, who gets as input a set $b_i \subseteq [N]$ in the problem $Disj_N$, constructs the collection $A_i = \{a_i^s \mid s \in b_i\}$ of subsets of $[n]$.

Now, if there exists $s \in \bigcap_{i=1}^k b_i$, then a k-packing exists: $a_1^s \in A_1, \dots a_k^s \in A_k$. On the other hand, if $b_i \cap b_j = \emptyset$ for all $i \neq j$, then for any two sets $a_i^{s_i} \in A_i$ and $a_j^{s_j} \in A_j$, we have that $s_i \neq s_j$, and thus $a_i^{s_i} \cap a_j^{s_j} \neq \emptyset$, meaning that the packing number is 1. □

9.2. The "number on the forehead" model

This model is related to many other important problems in circuit complexity, and is much more difficult to deal with than the previous one. Recall that in this model the information seen by players on a given input $x = (x_1, \dots, x_k)$ can overlap: the ith player has access to all the x_j's *except* x_i. Recall also that each x_i is an element from some (fixed in advance) n-element set X_i. Thus, we have two parameters: the size n of a domain for each players, and the number k of players.

We can imagine the situation as k poker players sitting around the table, and each one is holding a number to his/her forehead for the others to see. Thus, all players know the function f but their access to the input vector is restricted: the first player sees the string $(*, x_2, \dots, x_k)$, the second sees $(x_1, *, x_3, \dots, x_k)$, …, the kth player sees $(x_1, \dots, x_{k-1}, *)$.

Let $C_k(f)$ denote the minimum communication complexity of f in this "bit on forehead" model.

It is clear that $C_k(f) \leq \log_2 n + 1$ for any f: the first player writes the binary code of x_2, and the second player announces the result. But what about the lower bounds? The twist is that (for $k \geq 3$) the players share some inputs, and (at least potentially) can use this overlap to encode the information in some wicked and non-trivial way (see Exercises 2 and 3).

The lower bounds problem for $C_k(f)$ can be re-stated as a Ramsey type problem about the minimal number of colors in a coloring of the hypercube which leaves no "forbidden sphere" monochromatic.

A *Hamming sphere*, or just a *sphere* in X around a vector $x = (x_1, \dots, x_k)$ is a set S of k vectors of the form:

$$x^1 = (x_1', x_2, \dots, x_k),\ x^2 = (x_1, x_2', \dots, x_k),\ \dots,\ x^k = (x_1, x_2, \dots, x_k'),$$

where for each i, $x_i' \neq x_i$ and $x_i, x_i' \in X_i$. The vector x is a *center* of this sphere. Hence, there are exactly $(n-1)^k$ spheres around each vector x.

Such a sphere is *forbidden* (for f) if it lies entirely in one of the parts $f^{-1}(0)$ or $f^{-1}(1)$, whereas its center belongs to the other part.

An r-coloring $c : X \to \{1, \ldots, r\}$ of X is *legal* (with respect to f) if it:

a. leaves no forbidden sphere monochromatic;
b. uses different colors for vectors in $f^{-1}(0)$ and in $f^{-1}(1)$.

Let $\chi_k(f)$ denote the minimal number of colors in a legal coloring of X.

This measure is motivated by the fooling-set bound in the case of $k = 2$ players (see (7.3) in Section 7.1.3). In this case each function $f : X \to \{0,1\}$ with $X = X_1 \times X_2$ can be looked at as an $|X_1| \times |X_2|$ matrix. A sphere $x^1 = (x'_1, x_2)$, $x^2 = (x_1, x'_2)$ around an entry $x = (x_1, x_2)$ (as well as around the entry $x' = (x'_1, x'_2)$) of this matrix has then the form

$$\begin{array}{ccc} (x'_1, x_2) & \cdots & (x'_1, x'_2) \\ \vdots & & \vdots \\ (x_1, x_2) & \cdots & (x_1, x'_2) \end{array}$$

Such a sphere is forbidden if one of the following two configurations appear:‘

$$\begin{array}{ccccc} 1 & & * & & \\ \vdots & & & \text{or} & \\ 0 & \cdots & 1 & & \end{array} \qquad \begin{array}{ccccc} 0 & & * \\ \vdots & & \\ 1 & \cdots & 0 \end{array}$$

Being forbidden in this case means that the sphere cannot entirely lie in one monochromatic rectangle.

Our starting point is the following combinatorial lower bound of the k-party communication complexity.

PROPOSITION 9.5. *For every $f : X \to \{0,1\}$, $C_k(f) \geq \log_2 \chi_k(f)$.*

PROOF. Take an optimal protocol of the communication game for f. Color each vector $x \in X$ by the string, which is written on the blackboard at the end of communication between the players on the input x. We have $2^{C_k(f)}$ colors and it remains to verify that the coloring is legal for f.

To show this, assume that some forbidden sphere $S = \{x^1, \ldots, x^k\}$ around some vector x is monochromatic. Assume w.l.o.g. that $f(x^1) = \ldots = f(x^k) = 1$ and $f(x) = 0$. An important fact is that given the first l bits communicated by the players, the $(l+1)$-th bit of communication (transmitted, say, by the ith player) must be defined by a function which does not depend on the ith coordinate of the input: player P_i cannot see it. Therefore, for every l, there is an i $(1 \leq i \leq k)$ such that the $(l+1)$-th communicated bit is the same for both inputs x and x^i. Since on all inputs $x^1, \ldots, x^k$ the players behave in the same way (i.e., write the same string on the blackboard), it follows that they will also behave in the same way on the input x. But this means that the players will accept x, a contradiction. □

9.3. Discrepancy bound

Let $X_1, \ldots, X_k$ be finite sets, and $X = X_1 \times \cdots \times X_k$. A subset T_i of X is called a *cylinder* in the ith dimension if membership in T_i does not depend on the ith coordinate. That is,

$$(x_1, \ldots, x_i, \ldots, x_k) \in T_i \text{ implies that } (x_1, \ldots, x'_i, \ldots, x_k) \in T_i \text{ for all } x'_i \in X_i.$$

A subset $T \subseteq X$ is a *cylinder intersection* if it is an intersection $T = T_1 \cap \cdots \cap T_k$, where T_i is a cylinder in the ith dimension. The (normalized) *discrepancy* of a function

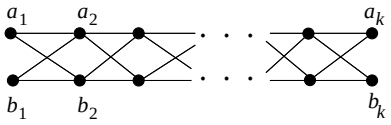


FIGURE 2. A cube

$f : X \to \{-1, 1\}$ on a set T is defined by

$$\mathrm{Disc}_T(f) = \frac{1}{|X|} \sum_{x \in T} f(x).$$

The *discrepancy* $\mathrm{Disc}(f)$ of f is the maximum, over all cylinder intersections T, of the absolute value $\left|\mathrm{Disc}_T(f)\right|$.

The case $k = 2$ is more intuitive. In this case, $X = X_1 \times X_2$ is just an $n \times n$ grid, and $f : X \to \{-1, 1\}$ is an $n \times n$ ± 1 matrix. Cylinder intersections T in this case are precisely the submatrices of X. Hence, in this case, $\mathrm{Disc}_T(f)$ is just the sum of all entries in the submatrix T divided by the size $|X|$ of the entire matrix.

The importance of the discrepancy stems from the fact that functions with small discrepancy have large multi-party communication complexity. For a function $F : X \to \{0, 1\}$, its ± 1 version $f : X \to \{-1, 1\}$ is defined by $f(x) = 1 - 2 \cdot F(x)$.

PROPOSITION 9.6. *For every $F : X \to \{0, 1\}$, $C_k(F) \geq -\log_2 \mathrm{Disc}(f)$.*

PROOF. It can be shown (see Exercise 9.8) that a set T is a cylinder intersection if and only if it does not separate a sphere from its center, i.e., if for every sphere S around a vector x, $S \subseteq T$ implies $x \in T$. Thus, a coloring $c : X \to \{1, \ldots, r\}$ is legal for a given function $F : X \to \{0, 1\}$ if and only if each color class $T = c^{-1}(i)$ is a cylinder intersection and the function F is constant on T. Since this last event is equivalent to $\left|\mathrm{Disc}_T(f)\right| = |T|/|X|$, no color class can have more than $|X| \cdot \mathrm{Disc}(f)$ vectors. This implies that we need at least $1/\mathrm{Disc}(f)$ colors, and Proposition 9.5 yields the desired lower bound on $C_k(f)$. □

However, this fact alone does not give immediate lower bounds for the multi-party communication complexity, because $\mathrm{Disc}(f)$ is very hard to estimate. Fortunately, the discrepancy can be bounded from above using the following more tractable measure.

A k-dimensional *cube* is defined to be a multi-set $D = \{a_1, b_1\} \times \cdots \times \{a_k, b_k\}$, where $a_i, b_i \in X_i$ (not necessarily distinct) for all i. Being a multi-set means that one element can occur several times. Thus, for example, the cube $D = \{a_1, a_1\} \times \cdots \times \{a_k, a_k\}$ has 2^k elements.

Given a function $f : X \to \{-1, 1\}$ and a cube $D \subseteq X$, define the *sign* of f on D to be the value

$$f(D) = \prod_{x \in D} f(x).$$

Hence, $f(D) = 1$ if and only if $f(x) = 1$ for an even number of vectors $x \in D$. We choose a cube D at random according to the uniform distribution. This can be done by choosing $a_i, b_i \in X_i$ for each i according to the uniform distribution. Let

$$\mathscr{E}(f) = \mathrm{E}\left[f(D)\right] = \mathrm{E}\left[\prod_{x \in D} f(x)\right]$$

be the expected value of the sign of a random cube D. To stress the fact that the expectation is taken over a particular random object (this time, over D) we will also write $\mathrm{E}_D\left[f(D)\right]$ instead of $\mathrm{E}\left[f(D)\right]$.

THEOREM 9.7. *For every $f : X \to \{-1,1\}$,*

$$\mathrm{Disc}(f) \le \mathscr{E}(f)^{1/2^k},$$

and hence,

$$C_k(f) \ge \frac{1}{2^k}\log_2\frac{1}{\mathscr{E}(f)}.$$

The theorem is very useful because $\mathscr{E}(f)$ is a much simpler object than $\mathrm{Disc}(f)$. For many functions f, it is relatively easy to compute $\mathscr{E}(f)$ exactly; we will demonstrate this in the next sections.

PROOF. We will only prove the theorem for $k = 2$; the general case is similar. So let $X = X_1 \times X_2$ and $f : X \to \{-1,1\}$ be a given function. Our goal is to show that $\mathrm{Disc}(f) \le \mathscr{E}(f)^{1/4}$. To do this, pick at random (uniformly and independently) an element $\boldsymbol{x} \in X$. The proof consists of showing two claims.

CLAIM 9.8. For all functions $h : X \to \{-1,1\}$, $\mathscr{E}(h) \ge (\mathrm{E}_{\boldsymbol{x}}[h(\boldsymbol{x})])^4$.

CLAIM 9.9. There exists h such that $\mathrm{E}_{\boldsymbol{x}}[h(\boldsymbol{x})] \ge \mathrm{Disc}(f)$ and $\mathscr{E}(h) = \mathscr{E}(f)$.

Together, these two claims imply the theorem (for $k = 2$). In the proof of these two claims we will use two known facts about the mean value of random variables:

$$\mathrm{E}\left[\xi^2\right] \ge \mathrm{E}[\xi]^2 \quad \text{for any random variable } \xi; \tag{9.1}$$

and

$$\mathrm{E}\left[\xi\cdot\xi'\right] = \mathrm{E}[\xi]\cdot\mathrm{E}\left[\xi'\right] \quad \text{if } \xi \text{ and } \xi' \text{ are independent.} \tag{9.2}$$

The first one is a consequence of the Cauchy–Schwarz inequality, and the second is a basic property of expectation.

PROOF OF CLAIM 9.8. Take a random 2-dimensional cube $D = \{a,a'\}\times\{b,b'\}$. Then

$$\begin{aligned}
\mathscr{E}(h) &= \mathrm{E}_D[h(D)] = \mathrm{E}_D\left[\prod_{x\in D} h(x)\right] \\
&= \mathrm{E}_{a,a'}\mathrm{E}_{b,b'}\left[h(a,b)\cdot h(a,b')\cdot h(a',b)\cdot h(a',b')\right] \\
&= \mathrm{E}_{a,a'}\left[\left(\mathrm{E}_b\left[h(a,b)\cdot h(a',b)\right]\right)^2\right] && \text{by (9.2)} \\
&\ge \left(\mathrm{E}_{a,a'}\mathrm{E}_b\left[h(a,b)\cdot h(a',b)\right]\right)^2 && \text{by (9.1)} \\
&= \left(\mathrm{E}_a\mathrm{E}_b\left[h(a,b)^2\right]\right)^2 && \Pr[a'] = \Pr[a] \\
&= \left(\mathrm{E}_a\left(\mathrm{E}_b[h(a,b)]\right)^2\right)^2 && \text{by (9.2)} \\
&\ge \left(\mathrm{E}_{a,b}[h(a,b)]\right)^4 && \text{by (9.1).}
\end{aligned}$$

□

PROOF OF CLAIM 9.9. Let $T = A\times B$ be a cylinder intersection (a submatrix of X, since $k = 2$) for which $\mathrm{Disc}(f)$ is attained. We prove the existence of h by the probabilistic method. The idea is to define a random function $\boldsymbol{g} : X_1 \times X_2 \to \{-1,1\}$ such that the expected value $\mathrm{E}\left[\boldsymbol{g}(x)\right] = \mathrm{E}_{\boldsymbol{g}}\left[\boldsymbol{g}(x)\right]$ is the characteristic function of T. For

this, define $\boldsymbol{g}$ as the product $\boldsymbol{g}(x) = \boldsymbol{g}_1(x) \cdot \boldsymbol{g}_2(x)$ of two random functions, whose values are defined on the points $x = (a, b) \in X_1 \times X_2$ by:

$$\boldsymbol{g}_1(a,b) = \begin{cases} 1 & \text{if } a \in A; \\ \text{set randomly to } \pm 1 & \text{otherwise} \end{cases}$$

and

$$\boldsymbol{g}_2(a,b) = \begin{cases} 1 & \text{if } b \in B; \\ \text{set randomly to } \pm 1 & \text{otherwise.} \end{cases}$$

These function have the property that $\boldsymbol{g}_1$ depends only on the rows and $\boldsymbol{g}_2$ only on the columns of the grid $X_1 \times X_2$. That is, $\boldsymbol{g}_1(a,b) = \boldsymbol{g}_1(a,b')$ and $\boldsymbol{g}_2(a,b) = \boldsymbol{g}_2(a',b)$ for all $a, a' \in X_1$ and $b, b' \in X_2$. Hence, for $x \in T$, $\boldsymbol{g}(x) = 1$ with probability 1, while for $x \notin T$, $\boldsymbol{g}(x) = 1$ with probability $1/2$ and $\boldsymbol{g}(x) = -1$ with probability $1/2$; this is so because the functions $\boldsymbol{g}_1, \boldsymbol{g}_2$ are independent of each other, and $x \notin T$ iff $x \notin A \times X_2$ or $x \notin X_1 \times B$. Thus, the expectation $\mathrm{E}\left[\boldsymbol{g}(x)\right]$ takes the value 1 on all $x \in T$, and takes the value $\frac{1}{2} + (-\frac{1}{2}) = 0$ on all $x \notin T$, i.e., $\mathrm{E}\left[\boldsymbol{g}(x)\right]$ is the characteristic function of the set T:

$$\mathrm{E}\left[\boldsymbol{g}(x)\right] = \begin{cases} 1 & \text{if } x \in T; \\ 0 & \text{if } x \notin T. \end{cases}$$

Let now $\boldsymbol{x}$ be a random vector uniformly distributed in $X = X_1 \times X_2$. Then

$$\mathrm{Disc}_T(f) = \mathrm{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x}) \cdot \mathrm{E}_{\boldsymbol{g}}\left[\boldsymbol{g}(\boldsymbol{x})\right]\right] = \mathrm{E}_{\boldsymbol{x}}\mathrm{E}_{\boldsymbol{g}}\left[f(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x})\right] = \mathrm{E}_{\boldsymbol{g}}\mathrm{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x})\right].$$

So there exists some choice of $g = g_1 \cdot g_2$ such that

$$\mathrm{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x}) \cdot g(\boldsymbol{x})\right] \geq \mathrm{Disc}_T(f) = \mathrm{Disc}(f)$$

and we can take $h(x) := f(x) \cdot g(x)$. Then $\mathrm{E}_{\boldsymbol{x}}\left[h(\boldsymbol{x})\right] \geq \mathrm{Disc}(f)$. Moreover, $\mathscr{E}(h) = \mathscr{E}(f)$ because g_1 is constant on the rows and g_2 is constant on the columns so the product $g(D) = \prod_{x \in D} g(x)$ cancels to 1. □

This completes the proof of Theorem 9.7 in case $k = 2$. To extend it for arbitrary k, just repeat the argument k times. □

9.4. Generalized inner product

Say that a $(0,1)$ matrix A is *odd* if the number of its all-1 rows is odd. Note that, if the matrix has only two columns, then it is odd iff the scalar (or inner) product of these columns over $GF(2)$ is 1. By this reason, a boolean function, detecting whether a given matrix is odd, is called "generalized inner product" function. We will assume that input matrices have n rows and k columns.

That is, the *generalized inner product function* $\textsc{gip}(x)$ is a boolean function in kn variables, arranged in an $n \times k$ matrix $x = (x_{ij})$, and is defined by:

$$\textsc{gip}(x) = \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{k} x_{ij}\,.$$

We consider k-party communication gates for $\textsc{gip}(x)$, where the the jth player can see all but the jth column of the input matrix x.

THEOREM 9.10. *The k-party communication complexity of* GIP *is* $\Omega\left(n4^{-k}\right)$.

It can be shown (see Exercise 9.6) that this lower bound is almost optimal: $C_k(\textsc{gip}) = O(kn/2^k)$.

PROOF. Since we want our function to have range $\{-1,1\}$, we will consider the function

$$f(x) = (-1)^{\mathrm{GIP}(x)} = \prod_{i=1}^{n} (-1)^{x_{i1}x_{i2}\cdots x_{ik}}. \tag{9.3}$$

By Theorem 9.7, it is enough to prove that $\mathscr{E}(f) \le 2^{-\Omega(n2^{-k})}$. In fact we will prove that

$$\mathscr{E}(f) = \left(1 - \frac{1}{2^k}\right)^n. \tag{9.4}$$

In our case, the function f is a mapping $f : X_1 \times X_2 \times \cdots X_k \to \{-1,1\}$, where the elements of each set X_j are column vectors of length n. Hence, a cube D in our case is specified by two $n \times k$ $(0,1)$ matrices $A = (a_{ij})$ and $B = (b_{ij})$. The cube D consists of all 2^k $n \times k$ matrices, the jth column in each of which is either the jth column of A or the jth column of B. By (9.3), we have that

$$\begin{aligned} f(D) &= \prod_{x\in D} f(x) = \prod_{x\in D}\prod_{i=1}^{n} (-1)^{x_{i1}x_{i2}\cdots x_{ik}} \qquad \text{with } x_{ij} \in \{a_{ij}, b_{ij}\} \\ &= \prod_{i=1}^{n}\prod_{x\in D} (-1)^{x_{i1}x_{i2}\cdots x_{ik}} \\ &= \prod_{i=1}^{n} (-1)^{(a_{i1}+b_{i1})(a_{i2}+b_{i2})\cdots(a_{ik}+b_{ik})}. \end{aligned}$$

Note that the exponent $(a_{i1}+b_{i1})(a_{i2}+b_{i2})\cdots(a_{ik}+b_{ik})$ is even if $a_{ij} = b_{ij}$ for at least one $1 \le j \le k$, and is equal to 1 in a unique case when $a_{ij} \ne b_{ij}$ for all $j = 1,\ldots,k$, that is, when the ith row of B is complementary to the ith row of A. Thus,

$$f(D) = -1 \text{ iff the number of complementary rows in } A \text{ and } B \text{ is odd.}$$

Now, $\mathscr{E}(f)$ is the average of the above quantity over all choices of matrices A and B. We fix the matrix A and show that the expectation over all matrices B is precisely the right-hand side of (9.4). Let $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n$ be the rows of A and $\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n$ be the rows of B. Then $f(D) = \prod_{i=1}^{n} g(\boldsymbol{b}_i)$, where

$$g(\boldsymbol{b}_i) := (-1)^{(a_{i1}+b_{i1})(a_{i2}+b_{i2})\cdots(a_{ik}+b_{ik})} = \begin{cases} +1 & \text{if } \boldsymbol{b}_i \ne \boldsymbol{a}_i \oplus \mathbf{1}, \\ -1 & \text{if } \boldsymbol{b}_i = \boldsymbol{a}_i \oplus \mathbf{1}. \end{cases}$$

Thus, for every fixed matrix A, we obtain that

$$\begin{aligned} \mathrm{E}_B\left[\prod_{i=1}^{n} g(\boldsymbol{b}_i)\right] &= \prod_{i=1}^{n} \mathrm{E}_{\boldsymbol{b}_i}[g(\boldsymbol{b}_i)] \qquad \text{by (9.2)} \\ &= \prod_{i=1}^{n} \frac{1}{2^k} \sum_{\boldsymbol{b}_i} g(\boldsymbol{b}_i) \\ &= \prod_{i=1}^{n} \frac{1}{2^k}\left(2^k - 1\right) \\ &= \left(1 - \frac{1}{2^k}\right)^n. \end{aligned}$$

□

9.5. Matrix multiplication

Let $X = X_1 \times \cdots X_k$, where each X_i is the set of all $m \times m$ matrices over the field $GF(2)$; hence, $|X_i| = n = 2^{m^2}$. For $x_1 \in X_1, \ldots, x_k \in X_k$, denote by $x_1 \cdots x_k$ the product of $x_1, \ldots, x_k$ as matrices over $GF(2)$. Let $F(x_1, \ldots, x_k)$ be a boolean function whose value is the element in the first row and the first column of the product $x_1 \cdots x_k$.

THEOREM 9.11. $C_k(F) = \Omega\left(m/2^k\right)$.

The theorem is a direct consequence of Theorem 9.7 and the following lemma. Define the function $f : X \to \{-1, 1\}$ by

$$f(x_1, \ldots, x_k) = (-1)^{F(x_1, \ldots, x_k)} = 1 - 2F(x_1, \ldots, x_k).$$

LEMMA 9.12. $\mathscr{E}(f) \le (k-1)2^{-m}$.

PROOF. For every cube $D = \{a_1, b_1\} \times \cdots \times \{a_k, b_k\}$,

$$f(D) = \prod_{x \in D} f(x) = \prod_{x \in D} (-1)^{F(x)} = (-1)^{\oplus_{x \in D} F(x)}.$$

Since F is linear in each variable,

$$f(D) = (-1)^{F(a_1 \oplus b_1, \ldots, a_k \oplus b_k)} = 1 - 2F(a_1 \oplus b_1, \ldots, a_k \oplus b_k),$$

where $a_i \oplus b_i$ denotes the sum of matrices a_i and b_i over $GF(2)$. If we choose D at random according to the uniform distribution, then $(a_1 \oplus b_1, \ldots, a_k \oplus b_k)$ is a random vector $\boldsymbol{x} = (x_1, \ldots, x_k)$ uniformly distributed over X. Therefore,

$$\begin{aligned} \mathscr{E}(f) &= \mathrm{E}_D\left[f(D)\right] = \mathrm{E}\left[1 - 2F(a_1 \oplus b_1, \ldots, a_k \oplus b_k)\right] \\ &= \mathrm{E}_{\boldsymbol{x}}\left[1 - 2F(\boldsymbol{x})\right] = \mathrm{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right]. \end{aligned}$$

To estimate the expectation $\mathrm{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right]$, where $\boldsymbol{x} = (x_1, \ldots, x_k)$ is uniformly distributed over X sequence of $m \times m$ matrices, let E_d denote the event that the first row of the matrix $x_1 \cdots x_d$ contains only 0's. Define $p_d = \Pr[E_d]$. Since p_1 is determined by x_1 and since x_1 is uniformly distributed, we have

$$p_1 = \Pr[E_1] = 2^{-m}.$$

Clearly we also have $\Pr[E_{d+1}|E_d] = 1$. On the other hand, since x_{d+1} is uniformly distributed, $\Pr[E_{d+1}|\neg E_d] = 2^{-m}$. Therefore, for all $1 \le d < k$,

$$\begin{aligned} p_{d+1} &= \Pr[E_{d+1}|E_d] \cdot \Pr[E_d] + \Pr[E_{d+1}|\neg E_d] \cdot \Pr[\neg E_d] \\ &= p_d + (1 - p_d) \cdot 2^{-m} \le p_d + 2^{-m}, \end{aligned}$$

implying that $p_d \le d \cdot 2^{-m}$ for all $d = 1, \ldots, k$.

If E_{k-1} occurs then $F(x_1, \ldots, x_k)$ is always 0, and hence, $f(x_1, \ldots, x_k)$ is always 1. If E_{k-1} does not occur then, since the first column of x_k is uniformly distributed, the value $F(x_1, \ldots, x_k)$ is uniformly distributed over $\{0, 1\}$, and hence, $f(x_1, \ldots, x_k)$ is uniformly distributed over $\{-1, 1\}$. Therefore,

$$\mathscr{E}(f) = \mathrm{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right] = \Pr[E_{k-1}] = p_{k-1} \le (k-1) \cdot 2^{-m}. \qquad \square$$

9.6. What about more than $\log n$ players?

Both lower bounds on the k-party communication complexity of the generalized inner product function $GIP_{m,k}$ and the matrix product function are useless when we have $k \geq \log_2 n$ players. To prove lower bounds for more than logarithmic number of players is a an old open problem. This problem remains open even for very special communication protocols, known as *simultaneous messages* protocols or *SM-protocols*. In this case no communication between the players is allowed. Instead of that, seeing his/her part of the input, each player sends a message to a "referee" who, having the messages from all k players, announces the answer.

Besides its own importance, proving lower bounds on the k-party communication complexity, when there are $k \geq \log_2 n$ players, is important since this would re-solve some other old open problems in circuit complexity. One of them is to prove super-polynomial lower bounds for ACC circuits. Recall that such circuits have unbounded fanin AND, OR and MOD_p gates, where

$$\mathrm{MOD}_p(x_1,\ldots,x_m) = 1 \quad \text{iff } x_1 + \ldots + x_m = 0 \bmod p\,.$$

When p is a prime power, exponential lower bounds for such circuits were proved by Razborov (1987) and Smolensky (1987). (We will do this for $p = 3$ in Section 11.4.) However, the case of composite moduli p—even the case of circuits with AND, OR and MOD_6 gates—remains widely open. On the other hand, ACC circuits are related to depth-2 circuits of the following special type.

A depth-2 *symmetric* (r,s)*-circuit* is a circuit of the form $\varphi(g_1,\ldots,g_s)$, where φ is a symmetric boolean function, and each g_i is an AND of at most r literals. Based on an earlier result of Yao (1990), Beigel and Tarui (1994) have shown that every ACC circuit can be simulated by a symmetric (r,s)-circuit with $r = \mathrm{polylog}(n)$ and $s = 2^{\mathrm{polylog}(n)}$.

Symmetric depth-2 circuits are related to communication games via the following fact, which actually holds for symmetric circuits with the g_i being *arbitrary* boolean functions of at most $k-1$ variables, not just AND gates.

LEMMA 9.13. *If a boolean function f in $n = km$ variables can be computed by a symmetric $(k-1,s)$-circuit then, for any equal-sized partition of the input among the players, the k-party communication complexity of f is $O(k \log s)$, and the SM-communication complexity of f is $O(\log s)$.*

PROOF. Since each bottom gate of our symmetric circuit has fanin at most $k-1$, there is at least one player who can evaluate that gate. Partition the bottom gates among the players such that all the gates assigned to a player can be evaluated by that player. Now each player broadcasts the *number* of her gates that evaluate 1. This takes $O(\log s)$ bits per player since the top gate has fanin at most s. Finally, one of the players can add up all the numbers broadcasted to compute the symmetric function given by the top gate and announce the answer.

It is obvious that this works in the SM model as well: each player sends to the referee the number of gates evaluating to 1 among her gates, and the referee adds these numbers to compute f. □

RESEARCH PROBLEM 9.14. *Prove a larger than polylog(n) lower bound on the SM-communication complexity for more than polylog(n) players.*

9.7. Best-partition k-party communication

Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function on $n = km$ variables. The "number on the forehead" communication protocols work with a *fixed* partition $x = (x_1, \ldots, x_k)$ of the input vector $x \in \{0,1\}^n$ into k blocks $x_i \in \{0,1\}^m$.

We now consider the situation where, given a function f, the players are allowed to choose the best, most suited for this particular function f balanced partition of its variables.

Say that a partition of a finite set into k disjoint blocks is *balanced* if the sizes of blocks differ by at most one. Note that, if there were no restriction on the sizes of the blocks, then the communication complexity of any function would be zero.

Let $C_k^{\text{best}}(f)$ denote the smallest possible k-party communication complexity of f over all balanced partitions of its input vector.

Recall that the generalized inner product function $\text{GIP}_{m,k}$ is a boolean function in $n = km$ variables which takes $m \times k$ $(0,1)$ matrix x as its input, and outputs 1 iff the number of all-1 rows in it is odd. We have already shown in Section 9.4 that, if we split the input matrix in such a way that the ith player can see all its columns but the ith one, then

$$C_k(\text{GIP}_{m,k}) = \Omega(n/k4^k). \tag{9.5}$$

On the other hand, the best-partition communication complexity of this function is very small: for every $k \geq 2$ we have that

$$C_k^{\text{best}}(\text{GIP}_{m,k}) \leq 2.$$

To see this, split the rows of the input $m \times k$ matrix x into m/k blocks and give to the ith player all but the ith block of these rows. Then the first player can write the parity of the number of all-1 rows she can see, and the second player can announce the answer.

So, what boolean functions have large k-party communication complexity under the best-partition of their inputs? To answer this question we use a graph theoretic approach.

Let $\mathscr{H}$ be a hypergraph on an n-element vertex set V, that is, a family of subsets $e \subseteq V$; the members of $\mathscr{H}$ are usually called *hyperedges*. Associate with each vertex $v \in V$ a boolean variable x_v and consider the following boolean function in these variables:

$$\text{GIP}_{\mathscr{H}}(x) = \bigoplus_{e \in \mathscr{H}} \bigwedge_{v \in e} x_v .$$

Note that, if $\mathscr{M}$ is a *k-matching*, that is, if the edges of $\mathscr{M}$ form a partition of V into $m = n/k$ blocks $e_1, \ldots, e_k$ of size $|e_i| = k$, then (up to renaming of variables),

$$\text{GIP}_{\mathscr{M}}(x) = \text{GIP}_{m,k}(x). \tag{9.6}$$

We have however just showed that for such hypergraphs, $C_k^{\text{best}}(\text{GIP}_{\mathscr{M}}) \leq 2$. Still, we could force $C_k^{\text{best}}(\text{GIP}_{\mathscr{H}})$ be large is we could show that, for *any* balanced partition of vertices into k parts, the hypergraph $\mathscr{H}$ must containsan *induced* k-matching on sufficiently many vertices.

If $\mathscr{H}$ is a hypergraph on a set V of vertices, and $S \subseteq V$ is a set of vertices, then the *sub-hypergraph induced by S* is the hypergraph $\mathscr{F}$ with the vertex set S and edge set $\mathscr{F} = \{e \in \mathscr{H} \mid e \subseteq S\}$. It is easy to see that then the function $\text{GIP}_{\mathscr{F}}$ is a subfunction of $\text{GIP}_{\mathscr{H}}$, and hence, the communication complexity of $\text{GIP}_{\mathscr{H}}$ is lower bounded by the

communication complexity of $\text{GIP}_{\mathscr{F}}$: just set $x_v = 0$ for all $v \notin S$. We fix this observation as

PROPOSITION 9.15. *Let $\mathscr{H}$ be a hypergraph on a set V of vertices. Suppose that for every balanced partition $V = V_1 \cup \cdots \cup V_k$ there is a subset $S \subseteq V$ of vertices such that $|S \cap V_i| = 1$ for all $i = 1, \ldots, k$, and the sub-hypergraph of $\mathscr{H}$ induced by S is a k-matching. Then*

$$C_k^{\text{best}}(\text{GIP}_{\mathscr{H}}) = \Omega\left(\frac{|S|}{k^2 4^k}\right).$$

PROOF. The induced by S k-matching $\mathscr{M}$ must have at least $|S|/k$ hyperedges, and the desired lower bound follows from (9.5) and (9.6). □

We will construct the desired hypergraphs $\mathscr{H}$ starting from (ordinary) "mixed enough" graphs $G = (V, E)$. Namely, call a graph *s-mixed* if, for any pair of disjoint s-element subsets of vertices, there is at least one edge between these sets. A *k-star* of a graph G is a set of its k vertices such that at least one of them is adjacent to all of the remaining $k - 1$ of these vertices.

THEOREM 9.16. *Let G be an s-mixed regular graph of degree $d \geq 2$ on n vertices. Let $2 \leq k \leq \min\{d, n/s\}$ and let $\mathscr{H}$ be the hypergraph whose hyperedges are all k-stars of G. Then*

$$C_k^{\text{best}}(\text{GIP}_{\mathscr{H}}) = \Omega\left(\frac{n - sk}{dk^2 4^k}\right).$$

Proof. Say that an n-vertex graph $G = (V, E)$ is *s-starry* if for any $2 \leq k \leq n/s$ and for any pairwise disjoint sets $S_1, \ldots, S_k \subseteq V$, each of size $|S_i| \geq s$, there exist vertices $v_1 \in S_1, \ldots, v_k \in S_k$ such that $\{v_1, \ldots, v_k\}$ forms a k-star of G.

Note that every s-starry graph is also s-mixed, since we can let $k = 2$. Interestingly, the converse is also true:

CLAIM 9.17. Every s-mixed graph is s-starry.

PROOF. Let $G = (V, E)$ be a s-mixed graph, and let $S_1, \ldots, S_k \subseteq V$ be pairwise disjoint subsets of its vertices each of size $|S_i| \geq s$. For $i \in \{1, \ldots, k\}$ let T_i be the set of all vertices $v \in V - S_i$ that are *not* not adjacent to *any* vertex in S_i. Since $|S_i| \geq s$ and since G is s-mixed, we have that $|T_i| \leq s - 1$. Hence, the set $T = \bigcup_{i=1}^k T_i$ can have at most

$$|T| \leq (s-1)k < sk \leq \sum_{i=1}^k |S_i| = \left|\bigcup_{i=1}^k S_i\right|$$

vertices. Thus, there must exist a vertex $v \in \left(\bigcup_{i=1}^k S_i\right) - T$. That is, v belongs to some S_i and does not belong to T. By the definition of T, $v \in S_i$ and $v \notin T$ means that v must be connected by an edge with at least one vertex in each of the sets S_j, $j \neq i$. But then v is a center of the desired star. □

Let now G be a graph satisfying the conditions of Theorem 9.16, and let $\mathscr{H}$ be the hypergraph of its k-stars. To finish the proof of Theorem 9.16, it is enough to prove the following

CLAIM 9.18. There is a subset $S \subseteq V$ of vertices satisfying the conditions of Proposition 9.15 and such that

$$|S| \geq \frac{n - sk}{kd}.$$

PROOF. Let $V = V_1 \cup \cdots \cup V_k$ be an arbitrary balanced partition of the set V into k blocks. Hence, $|V_j| \geq n/k \geq s$ for all j.

We define S recursively. Initially, let S be empty. In each stage, apply Claim 9.17 to find a k-star with vertices in each set of the partition. Add these k vertices to S. Delete the k vertices, and all their neighbors, from G. Repeat the procedure restricting the given partition to the remaining vertices of G.

After i stages, at most idk vertices have been removed from V, which means that each block in the partition (of the remaining vertices) has size at least $n/k - idk$. Since G is s-mixed, Claim 9.17 will apply as long as $n/k - idk \geq s$. Thus, the algorithm will run for at least $i \geq (n - sk)/(dk^2)$ stages.

Because all hyperedges of $\mathscr{H}$ (the k-stars of G) have exactly k vertices, and in each stage we remove all neighbors of the k vertices we added to S, the sub-hypergraph of $\mathscr{H}$ induced by S contains only one hyperedge for each stage, and these are pairwise disjoint. This, the absence of any other hyperedge of $\mathscr{H}$ lying in S, is the main reason to take k-stars as hyperedges of $\mathscr{H}$: since in each stage we remove all neighbors of the actual star, none of the remaining k-stars can intersect it.

This completes the proof of Claim 9.18, and thus, the proof of Theorem 9.16. □

Thus, what we need are explicit graphs satisfying the following two conditions:

(i) the graph must have small degree, but
(ii) any two sufficiently large subsets of vertices must be joined by at least one edge.

Graphs with these properties are known as *expander graphs*.

The following useful bound, observed by many researchers, ensures property (ii), as long as the second largest eigenvalue[1] of adjacency matrix is small enough.

LEMMA 9.19 (Expander Mixing Lemma). *If G is a d-regular graph on n vertices and $\lambda = \lambda(G)$ is the second largest eigenvalue of its adjacency matrix, then the number $e(S,T)$ of edges between every two (not necessarily disjoint) subsets S and T of vertices satisfies*

$$\left| e(S,T) - \frac{d|S| \cdot |T|}{n} \right| \leq \lambda \sqrt{|S| \cdot |T|}\,.$$

PROOF. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of the adjacency matrix M of G, and let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be the corresponding orthonormal basis of eigenvectors; here $\boldsymbol{x}_1$ is $\frac{1}{\sqrt{n}}$ times the all-1 vector $\mathbf{1}$. Let $\boldsymbol{v}_S$ and $\boldsymbol{v}_T$ be the characteristic vectors of S and T. Expand these two vectors as linear combinations

$$\boldsymbol{v}_S = \langle \boldsymbol{a}, \boldsymbol{v}_S \rangle = \sum_{i=1}^{n} a_i \boldsymbol{x}_i \quad \text{and} \quad \boldsymbol{v}_T = \langle \boldsymbol{b}, \boldsymbol{v}_T \rangle = \sum_{i=1}^{n} b_i \boldsymbol{x}_i$$

of the basis vectors. Since the $\boldsymbol{x}_i$ are orthonormal eigenvectors,

$$e(S,T) = \boldsymbol{v}_S^{\top} M \boldsymbol{v}_T = \Big(\sum_{i=1}^{n} a_i \boldsymbol{x}_i \Big)^{\top} M \Big(\sum_{i=1}^{n} b_i \boldsymbol{x}_i \Big) = \sum_{i=1}^{n} \lambda_i a_i b_i\,. \tag{9.7}$$

Since the graph G is d-regular, we have $\lambda_1 = d$. The first two coefficients a_1 and b_1 are scalar products of $\boldsymbol{x}_1 = \frac{1}{\sqrt{n}}\mathbf{1}$ with $\boldsymbol{v}_S$ and $\boldsymbol{v}_T$; hence, $a_1 = |S|/\sqrt{n}$ and $b_1 = |T|/\sqrt{n}$. Thus, the first term $\lambda_1 a_1 b_1$ in the sum (9.7) is precisely $\frac{d|S||T|}{n}$. Since $\lambda = \lambda_2$ is the second largest eigenvalue, the absolute value of the sum of the remaining $n-1$ terms

[1] Recall that λ is an eigenvalue of M if $M\boldsymbol{x} = \lambda \boldsymbol{x}$ for some vector $\boldsymbol{x} \neq \mathbf{0}$.

in this sum does not exceed $\lambda \sum_{i=2}^{n} |a_i b_i|$ which, by Cauchy–Schwarz inequality, does not exceed

$$\lambda \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\| = \lambda \|\boldsymbol{v}_S\| \cdot \|\boldsymbol{v}_T\| = \lambda \sqrt{|S| \cdot |T|}\,. \qquad \square$$

Thus, a graph G is mixed enough if $\lambda = \lambda(G)$ is small enough. Examples of such graphs are *Ramanujan graphs* $RG(n,q)$. These are $(q+1)$-regular graphs with the property that $\lambda \le 2\sqrt{q}$.

COROLLARY 9.20. *Ramanujan graphs $RG(n,q)$ are s-mixed for $s = 2n/\sqrt{q}$.*

PROOF. If $|S| = |T| = s$ then, by Lemma 9.19, there is at least one edge between S and T as long as $ds^2/n - \lambda s > 0$, which happens if $\lambda < ds/n$. Since for Ramanujan graph $RG(n,q)$, we have $d = q+1$ and $\lambda \le 2\sqrt{q}$, this graph is s-mixed as long as $(q+1)s/n > 2\sqrt{q}$ which, in particular, is the case for $s = 2n/\sqrt{q}$. $\square$

Explicit constructions of Ramanujan graphs on n vertices for every prime $q \equiv 1 \bmod 4$ and infinitely many values of n were given in Margulis (1973), Lubotzky, Phillips and Sarnak (1988); these were later extended to the case where q is an arbitrary prime power in Morgenstern (1994) and Jordan and Livné (1997).

Let q be a prime number lying between $16k^2$ and $32k^2$. Then the Ramanujan graph $G = RG(n,q)$ has degree $d = q+1$ and (by Corollary 9.20) is s-mixed for $s = 2n/\sqrt{q} \le n/2k$. Using such graphs, Theorem 9.16 yields the following

COROLLARY 9.21. *If $\mathscr{H}$ is the hypergraph of k-stars in G, then*

$$C_k^{\text{best}}(\text{GIP}_{\mathscr{H}}) = \Omega\left(\frac{n}{k^4 4^k}\right).$$

It can be shown that, this bound is tight with respect to the number k of players: for any balanced partition of n vertices into $k+1$ parts, we have that $C_k(\text{GIP}_{\mathscr{H}}) \le k+1$ (Exercise 9.9). Thus, for every constant $k \ge 2$ there is an explicit boolean function $f = \text{GIP}_{\mathscr{H}}$ such that

$$C_k^{\text{best}}(f) = \Omega(n) \quad \text{but} \quad C_{k+1}(f) = O(1).$$

Exercises

EX. 9.1. The *set cover* communication problem is as follows: we have k players each holding a collection A_i of subsets of $[n] = \{1,\dots,n\}$, and the players are looking for the smallest covering of $[n]$ using the sets in their collections. That is, the goal is to find the smallest number r of subsets $a_1,\dots,a_r$ of $[n]$ such that each a_j belongs to at least one A_i, and $a_1 \cup \cdots \cup a_r = [n]$.

Show that $O(kn^2)$ bits of communication are enough to construct a covering using at most $(\ln n + 1)$ times larger number of sets than an optimal covering algorithm would do.

Hint: Use a greedy protocol, like in the proof of Lemma 7.10.

EX. 9.2. For a fixed vector $x \in X$, there are many (how many?) spheres around it. How many colors do we need to leave none of them monochromatic?

EX. 9.3. Consider the function $f : X_1 \times X_2 \to \{0,1\}$ such that $f(x_1,x_2) = 1$ if and only if $x_1 = x_2$. Show that $\chi(f) = n$. *Hint*: If $\chi(f) < n$ then some color class contains two distinct vectors (x_1,x_1) and (x_2,x_2). What about the color of (x_1,x_2)?

Ex. 9.4. Three players want to compute the following boolean function $f(x,y,z)$ in $3m$ variables. Inputs x,y,z are vectors in $\{0,1\}^m$, and the function is defined by:

$$f(x,y,z)=\bigoplus_{i=1}^{m}\mathrm{Maj}(x_i,y_i,z_i)\,.$$

Prove that $C_3(f)\leq 3$. *Hint*: Show that the following protocol is correct. Each player counts the number of i's such that she can determine the majority of x_i,y_i,z_i by examining the bits available to her. She writes the parity of this number on the blackboard, and the final answer is the parity of the three written bits.

Ex. 9.5. Consider the following k-party communication game. Input is an $n\times k$ $(0,1)$ matrix A, and the ith player can see all A except its ith column. Suppose that the players a priori know that some string $v=(0,\ldots,0,1,\ldots,1)$ with the first 1 in position $t+1$, does not appear among the rows of A. Show that then the players can decide if the number of all-1 rows is even or odd by communicating only t bits.

Hint: Let y_i denote the number of rows of A of the form $(0,\ldots,0,1,\ldots,1)$, where the first 1 occurs in position i. For every $i=1,\ldots,t$, the ith player announces the parity of the number of rows of the form $(0,\ldots,0,*,1,\ldots,1)$, where the $*$ is at place i. Observe that this number is y_i+y_{i+1}. Subsequently, each player privately computes the mod 2 sum of all numbers announced. The result is (y_1+y_{t+1}) mod 2, where $y_{t+1}=0$.

Ex. 9.6. Prove that $C_k(\mathrm{GIP})=O(kn/2^k)$.

Hint: Use the previous protocol to show that (without any assumption) k-players can decide if the number of all-1 rows in a given $n\times k$ $(0,1)$ matrix is even or odd by communicating only $O(kn/2^k)$ bits. To do this, divide the matrix A into blocks with at most $2^{k-1}-1$ rows in each. For each block there will be a string v' of length $k-1$ such that neither $(0,v')$ nor $(1,v')$ occurs among the rows in that block. Using k bits the first player can make the string $(0,v')$ known to all players, and we are in the situation of the previous exercise.

Ex. 9.7. Consider the following multiparty game with the *referee*. As before, we have an $n\times k$ 0-1 matrix A, and the ith player can see all A except its ith column. The restriction is that now the players do not communicate with each other but simultaneously write their messages on the blackboard. Using only this information (and without seeing the matrix A), an additional player (the referee) must compute the string $P(A)=(x_1,\ldots,x_n)$, where x_i is the sum modulo 2 of the number of $1'$s in the ith row of A.

Let N be the maximal number of bits which any player is allowed to write on any input matrix. Prove that $N\geq n/k$.

Hint: For a matrix A, let $f(A)$ be the string $(p_1,\ldots,p_k)$, where $p_i\in\{0,1\}^N$ is the string written by the ith player on input A. For each possible answer $x=(x_1,\ldots,x_n)$ of the referee, fix a matrix A_x for which $P(A_x)=x$. Argue that $f(A_x)\neq f(A_y)$ for all $x\neq y$.

Ex. 9.8. Show that a set $T\subseteq X$ is a cylinder intersection if and only if, for every sphere S around a vector x, $S\subseteq T$ implies $x\in T$. *Hint*: For the "if" part consider the sets T_i of all vectors $(x_1,\ldots,x_i,\ldots,x_k)$ such that $(x_1,\ldots,x_i',\ldots,x_k)\in T$ for at least one $x_i'\in X_i$.

Ex. 9.9. Let $\mathscr{H}$ be a hypergraph on n vertices, and $2\leq k\leq n$ be a divisor of n Suppose that $|e|\leq k-1$ for all $e\in\mathscr{H}$. Show that then, for any balanced partition of the input into k parts, there is a k-party communication protocol evaluating $\mathrm{GIP}_{\mathscr{H}}$ using at most k bits of communication.

Hint: Given a partition of n vertices into k blocks, each $e\in\mathscr{H}$ must lie outside at least one of these blocks.

Ex. 9.10. Let us consider simultaneous messages n-party protocols for the parity function $f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, where the referee makes his life easier: he just outputs the *majority* of the answers of players. That is, we have n people, each with a random bit 0 or 1 on his (or her) forehead. Everybody can see everybody's else bit except his own. We are interested in how the group can coordinate their guesses so that the *majority* of people in the group guesses the correct answer. More precisely, each person casts a private vote (1 or 0); the outcome of the election is the value which the majority of voters cast. The voters are said to *win* the election when the outcome is equal to the parity of the n bits. Consider the following strategy for players:

> Each voter looks around at everybody else. If a voter sees as many 0's as 1's, she casts a vote 0. Otherwise, she assumes that the bit on her forehead is the same as the majority of the bits she sees; she then casts a vote consistent with this assumption.

Show that this strategy has a success probability $1 - \frac{1}{\theta(\sqrt{n})}$, that is, will correctly compute the parity for all 2^n but a fraction $1/\theta(\sqrt{n})$ of input vectors.

Bibliographic Notes

Theorem 9.3 is due to Nisan (2002), who also shows how the communication complexity of set packing is related to combinatorial auctions. The elegant proof of Lemma 9.1 is due Jaikumar Radhakrishnan and Venkatesh Srinivasan.

The "number on the forehead" model of multi-party communication games was first introduced by Chandra, Furst and Lipton (1983) who, for every $k \geq 2$, exhibited a function f with $C_k(f)$ grater than any constant. A much higher lower bound $C_k(f) = \Omega\left(4^{-k} \log_2 n\right)$ for the generalized inner product function (see Theorem 9.10) was later proved by Babai, Nisan, and Szegedy (1992). Grolmusz (1994) has shown that this lower bound for GIP is almost optimal (see Exercises 9.5 and 9.6). Similar lower bounds for other explicit functions were obtained by Chung (1990). The upper bound on the discrepancy, given in Theorem 9.7, greatly simplifies the analysis of k-party communication protocols. This result as well as its application to matrix product (Theorem 9.11) are due to Raz (2000). Results of Section 9.7 are due to Hayes (2001).

Part 3

Bounded Depth Circuits

CHAPTER 10

Depth-3 Circuits

We consider boolean circuits with unbounded fanin AND and OR gates. Inputs are variables $x_1, \ldots, x_n$ and their negations $\overline{x}_1, \ldots, \overline{x}_n$. A Π_3 circuit is a circuit of depth 3 whose gates are arranged in three layers: AND gate at the top of the circuit (this is the output gate), OR gates on the next (middle) layer, and AND gates on the bottom (next to the inputs) layer (see Fig. 1). A Σ_3 circuit is defined dually by interchanging the OR and AND gates. The *size* of such a circuit is the total number of gates in it.

Exponential lower bounds for depth-2 circuits (DNFs and CNFs) are easy to prove. So, for example, any DNF for the parity function $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ must have 2^{n-1} ANDs: every AND gate must have all n variables as inputs, for otherwise the DNF would make an error—accept two inputs of different parities. The situation with depth-3 circuits is much more complicated—this is the first nontrivial case.

10.1. Why depth 3 is interesting?

In last two decades several important methods of proving lower bounds for depth-3 circuits, and even for depth-d circuits with an arbitrary constant d, emerged. We will discuss these methods in this and the next chapter. For depth-d the obtained lower bounds have the form $2^{\Omega(n^{1/(d-1)})}$. This however does not solve the problem completely, since easy counting shows that boolean functions requiring much larger number of gates—namely, about $2^{(1-\varepsilon)n}$—exist.

To find an *explicit* boolean function f in n variables such that any depth-3 circuit for f requires $2^{\alpha n/\log\log n}$ gates for some $\alpha \to \infty$, is one of intriguing open problems. By a seminal result of Valiant (1977), this would give the first super-linear lower bound on the size of log-depth circuits with NOT and fanin-2 AND and OR gates, thus resolving one of the central problems in circuit complexity. We now show how does this happen. By a graph we will mean a directed graph.

A *labeling* of a graph is a mapping of the nodes into the integers. Such a labeling is *legal* if for each edge (u, v) the label of v is strictly greater than the label of u. The *depth* a graph is the largest number of nodes on directed path.

A canonical labeling is to assign each node the total number of edges on a longest directed path that terminates at that node. If the graph has depth d then this gives us a labeling using only d labels $0, 1, \ldots, d-1$. It is easy to verify that this is a valid labeling: if (u, v) is an edge then any path terminating in u can be prolonged to a path terminating in v.

OBSERVATION 10.1. The depth of a graph does not exceed the number of labels used by any legal labeling.

PROOF. All labels along a directed path must be distinct. □

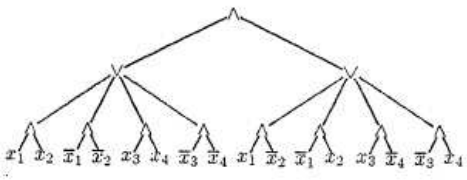


FIGURE 1. Depth-3 circuit for Parity $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ of $n = 4$ variables.

LEMMA 10.2. *Let $d = 2^k$ and $1 \le r \le k$ be integers. In any directed graph with s edges and depth d it is possible to remove rs/k edges so that the depth of the resulting graph does not exceed $d/2^r$.*

PROOF. Consider any directed graph with s edges and depth d, and consider the canonical labeling using labels $0, 1, \ldots, d-1$. For $i = 1, \ldots, \log_2 d = k$, let X_i be the set of all edges, the binary representations of labels of whose endpoints differ in the ith position (from the left) for the *first time*. If X_i is removed from the graph then we can relabel the nodes using integers $0, 1, \ldots, d/2 - 1$ by simply deleting the ith bit in the binary representation of the of labels. It is not difficult to see that this is a legal labeling (of a new graph): if an edge (u, v) survived, then the first difference between the binary representations of the old labels of u and v were not in the ith position. Consequently, if any $r \le k$ of the X_i's are removed, Observation 10.1 implies that a graph of depth at most $d/2^r$ remains. □

An important consequence is that any log-depth circuit of linear size can be reduced to a Σ_3 circuit of moderate fanin of middle layer gates and not too large fanin of the top gate.

LEMMA 10.3. *For every $\varepsilon > 0$, c_1 and c_2, there exists a constant K such that, if a boolean function f in n variables can be computed by a fanin-2 circuit of depth $c_1 \log n$ using $c_2 n$ gates, then f can be written as a sum of $\exp(Kn/\log\log n)$ CNFs each with at most $\exp(n^\varepsilon)$ clauses.*

Note that the top gate is now a *sum* gate (over the reals), not just an OR gate. Thus, what we obtain is a restricted version of a Σ_3 circuit: for every input, at most one AND gate on the middle layer is allowed to output 1.

PROOF. Take a circuit C of depth $c_1 \log n$ with $c_2 n$ fanin-2 gates. Hence, the circuit has at most $s \le 2c_2 n$ wires. Apply Lemma 10.2 with k about $\log(c_1 \log n)$ and r about $\log(c_1/\varepsilon)$ (a constant). This gives us a set S of $|S| \le sr/k = O(n/\log\log n)$ wires whose removal leaves us with a circuit of depth at most $d = 2^{-r} \cdot c_1 \log n = \varepsilon \log n$.

Take a set of new variables $y = (y_e \mid e \in S)$, one for each cut wire. For each such wire $e = (u, v) \in S$, let C_e be the subcircuit of C whose output gate is u. Each such subcircuit C_e depends on some x-variables and some y-variables. Moreover, each subcircuit C_e depends on at most $2^d = n^\varepsilon$ variables because each of these subcircuits has depth at most $\varepsilon \log_2 n$, and each gate has fanin at most 2. Hence, the test $y_e = C_e(x, y)$ can be written as a CNF $\phi_e(x, y)$ with at most $2^{2^d} = 2^{n^\varepsilon}$ clauses. Consider the CNF

$$\psi(x, y) = \phi_0(x, y) \wedge \bigwedge_{e \in S} \phi_e(x, y),$$

where ϕ_0 is the CNF of the last subcircuit, rooted in the output gate of the whole circuit. The CNF ϕ has $(|S|+1)2^{n^\varepsilon}$ clauses, and for every assignment $\alpha = (\alpha_e \mid e \in S)$ in $\{0,1\}^S$, we have that $\psi(x,\alpha) = 1$ iff $C(x) = 1$ and the computation of C on input x is consistent with the values assigned to cut wires by α. Since the computation of C on a given vector x cannot be consistent with two assignments $\alpha_1 \neq \alpha_2$, the function computed by our circuit C can be written as a sum

$$C(x) = \sum_{\alpha \in \{0,1\}^S} \psi(x,\alpha)$$

of $s = 2^{|S|}$ CNFs, each with at most $(|S|+1)2^{n^\varepsilon}$ clauses. □

The consequence, which makes depth-3 circuits interesting, is the following.

COROLLARY 10.4. *If a boolean function f in n variables requires Σ_3 circuits of size larger than $\exp(n/\log\log n)$, then f cannot be computed by a log-depth circuit using a linear number of fanin-2 gates.*

10.2. An easy lower bound for Parity

A binary vector is *odd* if it has an odd number of 1's; otherwise the vector is *even*. A parity function is a boolean function $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ which accepts all odd vectors and rejects all even vectors. Recall that a formula is a circuit in which all gates have fanout at most 1. The top fanin is the fanin of the output gate.

LEMMA 10.5. *Every Π_3 formula computing $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ with top fanin t requires at least $t2^{(n-1)/t}$ AND gates on the bottom layer.*

PROOF. Let s_i be the fanin of the ith OR gate on the middle layer. The ANDs at bottom layer can be labeled with (i,j) for $1 \leq i \leq t$ and $1 \leq j \leq s_i$ (see Fig. 1). Let $h_{i,j}$ denote the (i,j)-th AND. Then the circuit computes the function $\bigwedge_{i=1}^{t} \bigvee_{j=1}^{s_i} h_{i,j}$. By the distributive rule $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, this is an OR of ANDs of the form

$$H = h_{1,j_1} \wedge h_{2,j_2} \wedge \cdots \wedge h_{t,j_t}\,. \tag{10.1}$$

We call these "big" ANDs H the *monomials* produced by the circuit. We claim that: *each monomial H accepts at most one odd vector.* To show this, say that a variable x_i is *seen* by a gate, if either x_i or $\overline{x}_i$ is an input to this gate.

Case 1: Each of n variables is seen by at least one of $h_{1,j_1}, h_{2,j_2}, \ldots, h_{t,j_t}$. In this case, H is a (possibly inconsistent) product of *all* n variables, and hence, can accept at most one vector.

Case 2: Some variable x_i is seen by *none* of the gates $h_{1,j_1}, h_{2,j_2}, \ldots, h_{t,j_t}$. We claim that in this case $H^{-1}(1) = \emptyset$. Indeed, if the set $H^{-1}(1)$ of accepted inputs is not nonempty, that is, if the monomial H contains no variable together with its negation, then $H^{-1}(1)$ must contain a pair of two vectors that only differ in the ith position. But this is impossible, since one of these two vectors must be even, and the circuit would wrongly accept it.

Hence, we have $s_1 s_2 \cdots s_t$ monomials H, and each of them can accept at most one odd vector. Since we have 2^{n-1} odd vectors, this implies $s_1 s_2 \cdots s_t \geq 2^{n-1}$. Since our circuit is a formula, the total number of AND gates on the bottom layer is $s_1 + \cdots + s_t$. Using the fact that the arithmetic mean $(a+b)/2$ is greater than or equal to the geometric mean $(a \cdot b)^{1/2}$, we can conclude that

$$s_1 + \cdots + s_t \geq t(s_1 s_2 \cdots s_t)^{1/t} \geq t2^{(n-1)/t}\,.$$ □

10.3. The method of finite limits

The above argument only works for circuits with small top fanin, much smaller that n. We now describe another, less trivial argument which works for circuits with arbitrary top fanin.

Let $B \subseteq \{0,1\}^n$ be a set of vectors. A vector $y \notin B$ is a *k-limit* for a set B if for any k-element subset $S \subseteq \{1,\ldots,n\}$ of positions there exists a vector $x \in B$ such that $y \leq x$ and $x_i = y_i$ for all $i \in S$. In particular, if y is a k-limit for B then the fact that y does not belong to B cannot be detected by looking at k of fewer bits of y.

The following lemma reduces the lower bounds problem for depth-3 circuits to a purely combinatorial problem about finite limits. We say that a circuit C *separates* a pair $A, B \subseteq \{0,1\}^n$, $A \cap B = \emptyset$ if

$$C(x) = \begin{cases} 1 & \text{for } x \in A, \\ 0 & \text{for } x \in B. \end{cases}$$

We also say that a circuit has *bottom fanin* k if each gate, next to the inputs, has at most k *negated* inputs (the whole number of inputs to the gate may be n). By a Π_3^k circuit we will mean a Π_3 circuit of bottom fanin at most k.

LEMMA 10.6 (Limits and circuit size). *If every $1/\ell$ fraction of vectors in B has a k-limit in A, then every Π_3^k circuit separating (A,B) must have more than ℓ gates.*

PROOF. Suppose, for the sake of contradiction, that (A,B) can still be separated by a Π_3^k circuit of size ℓ. Since the last (top) gate is an AND gate, some of the OR gates g on the middle layer must separate the pair (A,B') for some $B' \subseteq B$ of size $|B'| \geq |B|/\ell$. By our assumption, the set A must contain a vector y which is a k-limit for the set B'. Hence,

$$g(y) = 1 \text{ and } g(x) = 0 \text{ for all } x \in B'.$$

To obtain the desired contradiction, we will show that then g, and hence, the whole circuit C, is forced to (incorrectly) reject the limit y.

Take an arbitrary AND gate h on the bottom layer feeding in g, and let S be the corresponding set of negated inputs to h. Since $|S| \leq k$ and since y is a k-limit for B', we know that y coincides on these inputs with some vector $x_S \in B'$. Since g is an OR gate and since g must reject all vectors in B', we also know that $h(x_S) = 0$. If some *negated* variable feeding in h computes 0 on x_S then it does the same on y (since y coincides with x_S on all positions in S), and hence, $h(y) = 0$. Otherwise, the 0 is produced on x_S by some *non-negated* variable. Since $y \leq x_S$, this variable must produce 0 also on y, and hence, $h(y) = 0$. Since this holds for every AND gate h feeding into g, this implies that also the gate g must (incorrectly) reject y, a contradiction. □

In oder to show that a given boolean function f cannot be computed by a Π_3 circuit with fewer than ℓ gates, we can now argue as follows.

a. Assume that f can be computed by such a circuit.
b. Assign some variables of f to constants in order to reduce the bottom fanin of the circuit till k.
c. Choose and appropriate subsets $A \subseteq f^{-1}(1)$ and $B \subseteq f^{-1}(0)$, and show that every subset $B' \subseteq B$ of size $|B'| \geq |B|/\ell$ has a k-limit vector $y \in A$.
d. Apply Lemma 10.6 to get a contradiction.

The bottom fanin can be reduced using the following simple lemma.

LEMMA 10.7. *Let $\mathscr{F}$ be a family of ℓ subsets of $[n]$ each of cardinality more than k. If*

$$\ell < \left(\frac{n}{e^{1/6}m}\right)^k, \tag{10.2}$$

then some subset T of $[n]$ of size $|T| = n - m$ intersects all members of $\mathscr{F}$.

PROOF. We construct the desired set T via the following "greedy" procedure. Since each set in $\mathscr{F}$ has more than k elements, and since we only have n elements in total, at least one element x_1 must belong to at least k/n fraction of sets in $\mathscr{F}$. Include such an element x_1 in T, remove all sets from $\mathscr{F}$ containing x_1 (hence, at most a $(1-k/n)$ fraction of sets in $\mathscr{F}$ remains), and repeat the procedure with the remaining sub-family of $\mathscr{F}$, etc. Our goal is to show that, if the initial family $\mathscr{F}$ had ℓ sets, and ℓ satisfies (10.2), then after $n-m$ steps all the sets of $\mathscr{F}$ will be removed.

The sub-family resulting after $n-m$ steps has at most $\ell \cdot \alpha$ sets, where

$$\begin{aligned}\alpha &= \left(1-\frac{k}{n}\right)\left(1-\frac{k}{n-1}\right)\cdots\left(1-\frac{k}{m+1}\right)\\ &\le e^{-\frac{k}{n}-\frac{k}{n-1}-\cdots-\frac{k}{m+1}} \le e^{-k(\ln n-\ln m-1/6)}\\ &= \left(\frac{n}{e^{1/6}m}\right)^{-k},\end{aligned}$$

where the last inequality follows from known estimates $H_n = \ln + \gamma_n$ on harmonic series $H_n = 1+1/2+/3+\cdots+1/n$ with $\frac{1}{2} < \gamma_n < \frac{2}{3}$. □

Now, having a Π_3 circuit of size ℓ satisfying (10.2), we can reduce its bottom fanin to k by just setting to 1 all variables in T. By Lemma 10.7, this will evaluate all bottom AND gates with *more* than k negated inputs to 0.

The next task—forcing a k-limit—depends on a boolean function we are dealing with. To demonstrate how this can be done, let us consider the Majority function $\mathrm{Maj}_n(x_1,\ldots,x_n)$ which accepts an input vector iff it contains at least so many 1's as 0's.

10.3.1. A lower bound for Majority. Let $\binom{[n]}{r}$ denote the r-th slice of the binary n-cube, that is, the set of all vectors in $\{0,1\}^n$ with precisely r ones.

LEMMA 10.8. *For every subset $B \subseteq \binom{[n]}{r}$ of size $|B| > k^r$ there is a k-limit y with fewer than r ones.*

PROOF. Induction on r. If $B \subseteq \binom{[n]}{1}$ and $|B| \ge k+1$ then $\mathbf{0} = (0,\ldots,0)$ is the desired k-limit of A. Suppose now that the lemma holds for all slices smaller that r and prove it for the r-th slice. So, take a set $B \subseteq \binom{[n]}{r}$ of size $|B| > k^r$. If $\mathbf{0}$ is a k-limit for B, then we are done.

Otherwise, by the definition of a k-limit, there must be a set of k coordinates such that every vector in B has at least one 1 among these coordinates. Hence, at least k^{-1} fraction of vectors in B must have a 1 in some, say ith, coordinate. Replace in all these vectors the ith 1 by 0, and let B' be the resulting set of vectors. Since $B' \subseteq \binom{[n]}{r-1}$ and $|B'| \ge |B|/k > k^{r-1}$, we have by the induction hypothesis that some vector y with fewer than $r-1$ ones is a k-limit for B'. The ith coordinate of y is 0. Replacing this coordinate by 1 we obtain a vector with at most $r-1$ ones and this vector is the desired k-limit for B. □

THEOREM 10.9. *Any depth-three circuit computing the majority function* Maj_n *has size at least* $2^{\Omega(\sqrt{n})}$.

PROOF. Let ℓ be the minimal size of a depth-three circuit computing $\neg\mathrm{Maj}_n$, the negation of majority, and hence, the minimal size of a depth-three circuit computing Maj_n itself. Since $\neg\mathrm{Maj}_n$ is self-dual (complementing the output and all inputs does not change the function), we can w.l.o.g. assume that we have a Π_3-circuit.

Let $k \le n$ and $r \le n/2$ be parameters (to be specified later). Set $m := n/2 + r$ and assume that the size ℓ of our circuit satisfies the inequality (10.2). Then, by Lemma 10.7, it is possible to set $n - m = n/2 - r$ of the variables to 1 so that the resulting circuit has bottom fanin at most k. The new circuit computes a boolean function $f : \{0,1\}^m \to \{0,1\}$ in m variables such that $f(x) = 1$ iff x has fewer than $n/2 - (n-m) = r$ ones. Hence, the new circuit separates the pair (A, B) of sets

$$A = \{\text{all vectors in } \{0,1\}^m \text{ with fewer than } r \text{ ones}\}$$

and

$$B = \{\text{all vectors in } \{0,1\}^m \text{ with precisely } r \text{ ones}\}.$$

Since the new circuit has size at most ℓ and its bottom fanin is at most k, Lemma 10.6 implies that no $1/\ell$ fraction of vectors in B can have a k-limit in A. Together with Lemma 10.8, this implies that $|B|/\ell = \binom{m}{r}/\ell$ cannot be larger than k^r. Hence,

$$\ell \ge \binom{m}{r} \cdot k^{-r} \ge \left(\frac{m}{kr}\right)^r. \tag{10.3}$$

By our assumption (10.2), this lower bound holds for any parameters k, r and $m = n/2 + r$ satisfying

$$\left(\frac{m}{kr}\right)^{r/k} < \frac{n}{e^{1/6}m}. \tag{10.4}$$

To ensure this, we can take, say, k about $\sqrt{n}$ and r about $\sqrt{n}/2$. Under this choice, (10.4) is fulfilled, and we obtain the desired lower bound

$$\ell \ge \left(\frac{m}{kr}\right)^r = 2^{\Omega(r)} = 2^{\Omega(\sqrt{n})}. \qquad \square$$

10.3.2. NP $\neq$ co-NP for depth-3 circuits. In this section we will exhibit a boolean function f in n variables such that f has a Σ_3 circuit of size $O(n)$ but its complement $\neg f$ requires Σ_3 circuits of size $2^{\Omega(\sqrt{n})}$. Note that we cannot take the majority function Maj_n for this purpose just because it is self-dual:

$$\neg\mathrm{Maj}_n(\neg x_1, \ldots, \neg x_n) = \mathrm{Maj}_n(x_1, \ldots, x_n).$$

Hence, by Theorem 10.9, both Maj_n and $\neg\mathrm{Maj}_n$ require Σ_3 circuits of exponential size. We therefore must use another function.

So, let $S_{s,m}$ be the boolean function with $n = 2sm$ variables defined by

$$S_{s,m}(x,y) = \bigvee_{i=1}^{s} \bigwedge_{j=1}^{m} (\overline{x}_{i,j} \vee \overline{y}_{i,j}). \tag{10.5}$$

This is an important function, known as *iterated disjointness* function. The function takes two sequences $x = (x_1, \ldots, x_s)$ and $y = (y_1, \ldots, y_s)$ of subsets of $[m] = \{1, \ldots, m\}$, and accepts the pair (x, y) iff $x_i \cap y_i = \emptyset$ for at least one $i \in [s]$.

It is clear (from its definition) that $S_{s,m}$ can be computed by a Σ_3 circuit of size $1 + s(m+1) = O(n)$. We shall show that, for $s = m = \sqrt{n}$, this function requires

Π_3-circuits of size $2^{\Omega(\sqrt{n})}$, implying that any Σ_3 circuit for its negation requires this size.

LEMMA 10.10. *If $f_{s,m}(x) = \bigvee_{i=1}^{s} \bigwedge_{j=1}^{m} \overline{x}_{i,j}$ is computed by a Π_3 circuit of size ℓ and bottom fanin k, then*

$$\ell \geq \left(\frac{m}{k}\right)^s .$$

PROOF. Any circuit for $f_{s,m}$ must separate the pair (A,B) where $A \subseteq \{0,1\}^{sm}$ is the set of all vectors with at most $s-1$ ones and B is the set of m^s vectors with exactly s ones killing all ANDs in f. Assume now that $\ell < \left(\frac{m}{k}\right)^s$. Then, by Lemma 10.6, no subset $B' \subseteq B$ of size $|B'| \geq |B|/\ell > m^s/\left(\frac{m}{k}\right)^s = k^s$ can have a k-limit in A, a contradiction with Lemma 10.8. □

LEMMA 10.11. *For any $k \leq sm$, any Π_3-circuit computing $S_{s,m}$ has size at least*

$$\min\left\{2^k, \left(\frac{m}{k}\right)^s\right\} .$$

PROOF. Take a Π_3-circuit computing $S_{s,m}(x,y)$, let ℓ be its size and assume that $\ell \leq 2^k$. We claim that then there exists a setting of constants to variables such that the resulting circuit has bottom fanin k and computes $f_{s,m}$. Together with Lemma 10.10, this claim implies that either $\ell \geq 2^k$ or $\ell \geq \left(\frac{m}{k}\right)^s$, and we are done. So it remains to prove the claim.

The most natural way is to randomly set one variable from each pair $x_{i,j}, y_{i,j}$ to 1. Any such setting will leave us with a circuit computing $f_{s,m}$. It remains therefore to show that at least one of such settings will leave no bottom AND gate with more than k negated inputs.

If a bottom AND gate contains both $x_{i,j}$ and $y_{i,j}$ negatively for some i,j then it is always reduced to 0. Otherwise such an AND gate with $> k$ negated inputs is *not* reduced to 0 with probability $\leq 2^{-(k+1)}$. Since we have at most $\ell \leq 2^k$ such AND gates, the probability that some of them will be not reduced to 0 does not exceed $\ell \cdot 2^{-(k+1)} \leq 1/2$. This, in particular, means that such a setting of constants exists. □

COROLLARY 10.12. *Any Π_3-circuit computing $S_{\sqrt{n},\sqrt{n}}$ has size at least $2^{\Omega(\sqrt{n})}$.*

PROOF. Take $s = m = \sqrt{n}$ and $k = \sqrt{n}/2$ in Lemma 10.11. □

REMARK 10.13. Recently, Razborov and Sherstov (2008) have shown that the iterated disjointness function (10.5) is hard in yet another respect: if $A = (a_{x,y})$ in an $n \times n$ ± 1 matrix with $n = m^3$ and $a_{x,y} = 1 - 2 \cdot S_{m,m^2}(x,y)$, then A has sign-rank $2^{\Omega(n^{1/3})}$. Recall that the *sign-rank* of a real matrix A with no zero entries is the least rank of a matrix $B = (b_{x,y})$ such that $a_{x,y} \cdot b_{x,y} > 0$ for all x, y. This result resolved an old problem about the power of probabilistic unbounded-error communication complexity.

The highest lower bounds for depth-3 circuits computing explicit boolean functions in n variables have the form $2^{\Omega(\sqrt{n})}$. We have seen how such lower bound can be derived for the majority function. To break this "square root barrier" is an important open problem. It is especially interesting in view of possible consequences for log-depth circuits (see Corollary 10.4).

RESEARCH PROBLEM 10.14. *Prove an explicit lower bound for Σ_3 circuits larger than $2^{\Omega(\sqrt{n})}$.*

To get such lower bounds, one could try to use the graph theoretic approach introduced in Section 1.8.

10.4. Graph theoretic lower bounds

Recall first what does it means that a boolean function g (or a circuit) "represents" a given graph G. The variables of g correspond to vertices of G, one for each vertex. The function g accepts/rejects subsets of vertices. We say that g *represents* the graph G if it accepts all edges and rejects all non-edges. On other subsets of vertices g may output arbitrary values.

The characteristic function of a bipartite $n \times n$ graph G with $n = 2^m$ is a boolean function $f_G(x, y)$ in $2m$ variables such that $f_G(x, y) = 1$ iff the vertices corresponding to vectors x and y are adjacent in G.

Define the *size* of a Σ_3 circuit as the maximum $\max\{s, r\}$, where s is the fanin of its top OR gate, and r is the maximum fanin of its AND gates on the middle layer. Let $\Sigma_3(G)$ denote the smallest size of a monotone Σ_3 representing the graph G. For a boolean function f, let $\Sigma_3^{\oplus}(f)$ denote the smallest size of a $\Sigma_3^{\oplus}$ circuit computing f.

Magnification Lemma (Lemma 1.12 in Section 1.8) immediately yields:

PROPOSITION 10.15. *For every bipartite graph G, $\Sigma_3(f_G) \geq \Sigma_3(G)$.*

This motivates the following problem.

RESEARCH PROBLEM 10.16. *Prove that an explicit bipartite $n \times n$ graph cannot be represented by a monotone Σ_3 circuit using fewer than $n^{1/k}$ gates, where $k = \log\log\log n$.*

By Corollary 10.4 and Proposition 10.15, this would re-solve an old problem in circuit complexity, namely, give an explicit boolean function which cannot be computed by a log-depth circuit using a linear number of fanin-2 gates.

Using counting arguments it can be shown that almost all bipartite $n \times n$ graphs require monotone Σ_3 circuits of size $\Omega(\sqrt{n})$ (Exercise 10.3). The problem therefore is to exhibit a specific graph.

Each monotone Σ_3 circuit for a graph G is just an OR of monotone CNFs

$$F = \left(\bigvee_{v \in S_1} x_v\right) \wedge \left(\bigvee_{v \in S_2} x_v\right) \wedge \cdots \wedge \left(\bigvee_{v \in S_r} x_v\right).$$

Such a CNF rejects a pair (u, v) of vertices iff at least one of the complements $I_i = \overline{S_i}$ covers this pair, that is, contains both endpoints u and v. Hence, F represents a graph H iff $I_1, \ldots, I_r$ are independent sets of H whose union covers all non-edges of H.

Thus, if $\mathrm{cnf}(H)$ denotes the minimum number of clauses in a monotone CNF representing the graph H, and if A is the adjacency matrix of H, then

$$\mathrm{cnf}(H) = \mathrm{Cov}(\overline{A}),$$

where $\overline{A}$ is the complement of A, and $\mathrm{Cov}(B)$ is the smallest number of all-1 submatrices of B covering all its ones.

We have already considered the cover number $\mathrm{Cov}(B)$ in Section 7.1.3 and showed (see Lemma 7.9) that, for every $(0,1)$ matrix B,

$$\mathrm{Cov}(B) = O(d \ln |B|),$$

where $|B|$ is the total number of ones in B, and d is the maximal number of zeroes in a line (row or column) of B. When translated to the language of graphs, this yields:

PROPOSITION 10.17. *For every bipartite $n \times n$ graph G of maximum degree d, we have that* $\mathrm{cnf}(G) = O(d \log n)$ *and* $\Sigma_3(G) = O(\sqrt{d} \log n)$.

Thus, all graphs of small degree can be represented by small monotone Σ_3 circuits. This also means that in order to solve Problem 10.16 above, we must consider graphs of large degree. In particular, good candidates must be dense enough, that is, have many edges.

It is conjectured that dense $K_{2,2}$-free graphs, that is bipartite graphs without 4-cycles, could be good candidates.

As we already mentioned in Section 3.3, explicit constructions of dense triangle-free graphs without 4-cycles are known. Such is, for example, the point-line incidence $n \times n$ graph G_n of a projective plane $PG(2,q)$ for a prime power q. Such a plane has $n = q^2 + q + 1$ points and n subsets of points (called lines). Every point lies in $q+1$ lines, every line has $q+1$ points, any two points lie on a unique line, and any two lines meet is a unique point. Now, if we put points on the left side and lines on the right, and joint a point x with a line L by an edge iff $x \in L$, then the resulting bipartite $n \times n$ graph will have $(q+1)n = \Theta(n^{3/2})$ edges and contain no 4-cycles.

RESEARCH PROBLEM 10.18. *Prove or disprove:* $\Sigma_3(G_n) \geq n^{\Omega(1)}$.

If the bound is true, this would clearly resolve Problem 10.16, and hence, yield the first super-linear lower bound for log-depth circuits.

In the next section we will prove the desired lower bounds for modified Σ_3 circuits, where all gates on the bottom level are Parity gates (not OR gates).

10.4.1. Depth-3 circuits with parity gates. Let us consider $\Sigma_3^{\oplus}$ circuits. These are Σ_3 circuits with the OR gates on the bottom (next to the inputs) layer replaced by Parity gates. Hence, at each AND gate on the middle layer a characteristic function of some affine subspace over $GF(2)$ is computed. The fanin of the top OR gate tells therefore how many affine subspaces, lying within $f^{-1}(1)$, do we need to cover the whole set $f^{-1}(1)$.

Let $\Sigma_3^{\oplus}(G)$ denote the smallest top fanin of a $\Sigma_3^{\oplus}$ representing the graph G. For a boolean function f, let $\Sigma_3^{\oplus}(f)$ denote the smallest top fanin of a $\Sigma_3^{\oplus}$ circuit computing f.

Our starting point is the following immediate consequence of the Magnification Lemma (Lemma 1.12):

PROPOSITION 10.19. *For every bipartite graph G, $\Sigma_3^{\oplus}(f_G) \geq \Sigma_3^{\oplus}(G)$.*

Hence, if $\Sigma_3^{\oplus}(G) \geq n^{\varepsilon}$, then $\Sigma_3^{\oplus}(f_G) \geq 2^{\varepsilon m}$; recall that f_G is a boolean function in $2m$ variables.

We are going to prove a general lower bound: any dense graph without large complete subgraphs requires large top fanin of $\Sigma_3^{\oplus}$ circuits. This immediately yields exponential lower bounds for many explicit functions.

A graph is *$K_{a,b}$-free* if it does not contains a complete $a \times b$ subgraph. For a graph G, by $|G|$ we will denote the number of edges in it.

THEOREM 10.20. *If an $n \times n$ graph G is $K_{a,b}$-free, then*

$$\Sigma_3^{\oplus}(G) \geq \frac{|G|}{(a+b)n}\,.$$

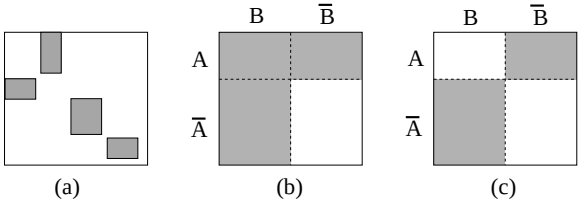


FIGURE 2. (a) An adjacency matrix of a fat matching, (b) the adjacency matrix of a graph represented by an OR gate $g = \bigvee_{v \in A \cup B} x_v$, and (c) the adjacency matrix of a graph represented by a Parity gate $g = \bigoplus_{v \in A \cup B} x_v$.

To prove the theorem, we first give a combinatorial characterization of the top fanin of $\Sigma_3^{\oplus}$ circuits, representing bipartite graphs, and then give a general lower bound on this characteristic.

A *fat matching* is a union of vertex-disjoint bipartite cliques (these cliques need not to cover all vertices). A *fat covering* of a graph G is a family of fat matchings such that each of these fat matchings is a subgraph of G and every edge of G is an edge of at least one member of the family.

Let $\text{fat}(G)$ denote the minimum number of fat matchings in a fat covering of G. Theorem 10.20 is a direct consequence of the following two lemmas.

LEMMA 10.21. *For every bipartite graph G,* $\text{fat}(G) = \Sigma_3^{\oplus}(G)$.

PROOF. Let U and V be the color classes of G, and let $g = \bigoplus_{v \in A \cup B} x_v$ with $A \subseteq U$ and $B \subseteq V$ be a gate on the bottom level of a $\Sigma_3^{\oplus}$ circuit representing G. Since g is a parity gate, it accepts a pair uv of vertices $u \in U$, $v \in V$ iff either $u \in A$ and $v \notin B$, or $u \notin A$ and $v \in B$. Thus, g represents a fat matching $(A \times \overline{B}) \cup (\overline{A} \times B)$ where $\overline{A} = U - A$ and $\overline{B} = V - B$ (see Fig 2(b)). Since the intersection of two fat matchings is again a fat matching (show this!), each AND gate on the middle level represents a fat matching. Hence, if the circuit has top fanin s, then the OR gate on the top represents a union of these s fat matchings, implying that $s \geq \text{fat}(G)$.

To show $\Sigma_3^{\oplus}(G) \leq \text{fat}(G)$, let $M = \bigcup_{i=1}^{r} A_i \times B_i$ be a fat matching. Set $A = \bigcup_{i=1}^{r} A_i$ and $B = \bigcup_{i=1}^{r} B_i$. We claim that the following AND of Parity gates represents M:

$$F = \left(\bigoplus_{u \in A} x_u \right) \left(\bigoplus_{v \in B} x_v \right) \left(\bigoplus_{w \in A_1 \cup \overline{B_1}} x_w \right) \cdots \left(\bigoplus_{w \in A_r \cup \overline{B_r}} x_w \right).$$

Indeed, if a pair $e = uv$ of vertices belongs to M, say, $u \in A_1$ and $v \in B_1$, then the first three sums accept uv because $u \in A_1$ and $v \notin \overline{B_1}$. Moreover, the mutual disjointness of the A_i as well as of the B_i implies that $u \notin A_i$ and $v \in B_1 \subseteq \overline{B_i}$ for all $i = 2, \ldots, r$. Hence, each of the last sums accepts the pair uv as well. To prove the other direction, suppose that a pair uv of vertices is accepted by F. The last r sums ensure that, for each $i = 1, \ldots, r$, one of the following must hold:

(a) $u \in A_i$ and $v \in B_i$;
(b) $u \notin A_i$ and $v \notin B_i$.

The first two sums of F ensure that (b) cannot happen for all i. Hence, (a) must happen for some i, implying that uv belongs to M.

Thus, every graph G can be represented by a $\Sigma_3^{\oplus}$ circuit of top fanin $\text{fat}(G)$. □

LEMMA 10.22. *Let G be a bipartite $n \times n$ graph. If G is $K_{a,b}$-free then*

$$\mathrm{fat}(G) \geq \frac{|G|}{(a+b)n}.$$

PROOF. Let $H = \bigcup_{i=1}^{t} A_i \times B_i$ be a fat matching, and suppose that $H \subseteq G$. By the definition of a fat matching, the sets $A_1, \ldots, A_t$, as well as the sets $B_1, \ldots, B_t$ are mutually disjoint. Moreover, since G contains no copy of $K_{a,b}$, we have that $|A_i| < a$ or $|B_i| < b$ for all i. Hence, if we set $I = \{i : |A_i| < a\}$, then

$$|H| = \sum_{i=1}^{t} |A_i \times B_i| = \sum_{i=1}^{t} |A_i| \cdot |B_i| \leq \sum_{i \in I} a \cdot |B_i| + \sum_{i \notin I} |A_i| \cdot b \leq (a+b)n.$$

Thus, no fat matching $H \subseteq G$ can cover more than $(a+b)n$ edges of G, implying that we need at least $|G|/(a+b)n$ fat matchings to cover all edges of G. □

There are many explicit bipartite graphs which are dense enough and do not have large complete bipartite subgraphs. By Theorem 10.20 and Proposition 10.19, each of these graphs immediately give us an explicit boolean function requiring an exponential (in the number of variables) lower bound on the top fanin of their $\Sigma_3^{\oplus}$ circuits.

To give an example consider the *disjointness function*. This is a boolean function $DISJ_{2m}$ in $2m$ variables such that

$$DISJ_{2m}(y_1, \ldots, y_m, z_1, \ldots, z_m) = 1 \text{ if and only if } \sum_{i=1}^{m} y_i z_i = 0\,.$$

THEOREM 10.23. *Every $\Sigma_3^{\oplus}$ circuit for $DISJ_{2m}$ has top fanin $2^{0.08m}$.*

PROOF. The graph G_f of the function $f = DISJ_{2m}$ is Kneser-type bipartite graph $K_m \subseteq U \times V$ where U and V consist of all $n = 2^m$ subsets of $[m] = \{1, \ldots, m\}$, and $uv \in K_m$ iff $u \cap v = \emptyset$. The graph K_m can contain a complete bipartite $a \times b$ subgraph $A \times B \neq \emptyset$ only if $a \leq 2^k$ and $b \leq 2^{m-k}$ for some $0 \leq k \leq m$, because then $\left(\bigcup_{u \in A} x_u\right) \cap \left(\bigcup_{v \in B} x_v\right) = \emptyset$. In particular, K_m can contain a copy of $K_{a,a}$ only if $a \leq 2^{m/2} = \sqrt{n}$. Since this graph has

$$|K_m| = \sum_{u \in U} d(u) = \sum_{u \in U} 2^{m-|u|} = \sum_{i=0}^{m} \binom{m}{i} 2^{m-i} = 3^m \geq n^{1.58}$$

edges, Theorem 10.20 yields that any $\Sigma_3^{\oplus}$ circuit representing K_m—and hence, any $\Sigma_3^{\oplus}$ circuit computing $DISJ_{2m}$—must have top fanin at least

$$\frac{|K_m|}{2an} \geq \frac{n^{1.58}}{n^{1.5}} = n^{0.08} = 2^{0.08m}\,.$$ □

We now consider a generalization of $\Sigma_3^{\oplus}$ circuits, where we allow to use an arbitrary *threshold* gate, instead of an OR gate, on the top. To analyze such circuits, we need the so-called "discriminator lemma" for threshold gates.

Let $\mathscr{B}$ be a family of subsets of a finite set X. A family $B_1, \ldots, B_t$ of members of $\mathscr{B}$ is *threshold cover* of a set $A \subseteq X$, it there exists a number $0 \leq k \leq t$ such that, for every $x \in X$, $x \in A$ if and only if x belongs to at least k of B_i. Let $\mathrm{thr}_{\mathscr{B}}(A)$ denote the minimum number t of members of $\mathscr{B}$ in a threshold cover of A.

To lower bound $\mathrm{thr}_{\mathscr{B}}(A)$ the following measure turned out to be very useful:

$$\Delta_{\mathscr{B}}(A) = \max_{B \in \mathscr{B}} \Delta_B(A)\,,$$

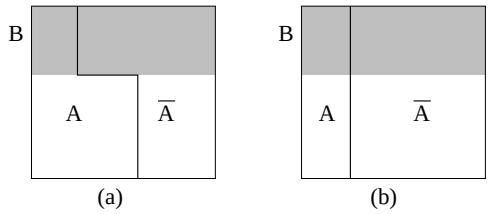


FIGURE 3. Schematic description of discriminators: $\Delta_B(A)$ is large in case (a), and is small in case (b).

where

$$\Delta_B(A) = \left| \frac{|A \cap B|}{|A|} - \frac{|\overline{A} \cap B|}{|\overline{A}|} \right| .$$

Small $\Delta_{\mathscr{B}}(A)$ means that every member B of $\mathscr{B}$ is splitted between the set A and its complement $\overline{A}$ rather balanced: the portion of $B \cap A$ in A is almost the same as the portion of $B \cap \overline{A}$ in $\overline{A}$. That is, the set A does not "discriminate" any member of $\mathscr{B}$.

LEMMA 10.24 (Discriminator Lemma).

$$\mathrm{thr}_{\mathscr{B}}(A) \geq \frac{1}{\Delta_{\mathscr{B}}(A)} .$$

PROOF. Let $B_1, \ldots, B_t \in \mathscr{B}$ be a threshold-k covering of A, i.e. $x \in A$ iff x belongs to at least k of B_i's. Our goal is to show that then $\Delta_{\mathscr{B}}(A) \geq 1/t$.

Since every element of A belongs to at least k of the sets $A \cap B_i$, the average size of these sets must be at least k. Since no element of $\overline{A}$ belongs to more than $k-1$ of the sets $\overline{A} \cap B_i$, the average size of these sets must be at most $k-1$. Hence,

$$1 \leq \frac{1}{|A|} \sum_{i=1}^{t} |A \cap B_i| - \frac{1}{|\overline{A}|} \sum_{i=1}^{t} |\overline{A} \cap B_i| \leq t \cdot \max_{1 \leq i \leq t} \left| \frac{|A \cap B_i|}{|A|} - \frac{|\overline{A} \cap B_i|}{|\overline{A}|} \right| . \qquad \square$$

The next fact which we need is one fact about Hadamard matrices.

An *Hadamard matrix* of oder n is an $n \times n$ matrix with entries ± 1 and with row vectors mutually orthogonal.

LEMMA 10.25 (Lindsey's Lemma). *The absolute value of the sum of all entries in any $a \times b$ submatrix of an $n \times n$ Hadamard matrix H does not exceed $\sqrt{abn}$.*

PROOF. By the definition of H, the matrix $M = \frac{1}{\sqrt{n}} H$ is unitary: $M^t M = I$. Since such matrices preserve the Euclidean norm, for every real vector v, we have $\|Mv\| = \|v\|$, and hence, $\|Hv\| = \sqrt{n}\|v\|$.

Now, if we denote by v_S the characteristic 0-1 vector of $S \subseteq \{1, \ldots, n\}$, with $v_S(i) = 1$ iff $i \in S$, then the absolute value of the sum of all entries in an $|S| \times |T|$ submatrix of H is the absolute value of the scalar product of vectors v_S and Hv_T. By the Cauchy–Schwarz inequality, this value does not exceed

$$\|v_S\| \cdot \|Hv_T\| = \sqrt{n}\|v_S\| \cdot \|v_T\| = \sqrt{n|S||T|} . \qquad \square$$

Now we are able to prove high lower bounds on the size of $\Sigma_3^{\oplus}$ with an arbitrary threshold gate on the top.

A graph associated with an Hadamard matrix M (or just an Hadamard graph) of oder n is a bipartite $n \times n$ graph where two vertices u and v are adjacent if and only if $M[u,v] = +1$.

THEOREM 10.26. *Any $\Sigma_3^{\oplus}$ circuit which has an arbitrary threshold gate on the top and represents an $n \times n$ Hadamard graph must have top fanin $\Omega(\sqrt{n})$.*

PROOF. Let A be an $n \times n$ Hadamard graph. Take an arbitrary $\Sigma_3^{\oplus}$ circuit which has an arbitrary threshold gate on the top and represents A. Let s be the fanin of this threshold gate, and let $\mathscr{B}$ be the set of all fat matchings. Then, by Lemma 10.21, $s \geq \mathrm{thr}_{\mathscr{B}}(A)$. To prove $s = \Omega(\sqrt{n})$ it is enough, by the Discriminator Lemma, to show that, for every fat matching $B = \bigcup_{i=1}^{t} S_i \times R_i$,

$$\left| \frac{|A \cap B|}{|A|} - \frac{|\overline{A} \cap B|}{|\overline{A}|} \right| = O(n^{-1/2}).$$

Since both the graph A and its bipartite complement [1] $\overline{A}$ have $\Theta(n^2)$ edges, it is enough to show that

$$\left| |A \cap B| - |\overline{A} \cap B| \right| \leq n^{3/2}.$$

By Lindsey's lemma, the absolute value of the difference

$$|A \cap (S_i \times R_i)| - |\overline{A} \cap (S_i \times R_i)|$$

does not exceed $\sqrt{s_i r_i n}$, where $s_i = |S_i|$ and $r_i = |R_i|$. Since both sums $\sum_{i=1}^{t} s_i$ and $\sum_{i=1}^{t} r_i$ are at most n, we obtain

$$\begin{aligned} \left| |A \cap B| - |\overline{A} \cap B| \right| &= \left| \sum_{i=1}^{t} |A \cap (S_i \times R_i)| - \sum_{i=1}^{t} |\overline{A} \cap (S_i \times R_i)| \right| \\ &\leq \sum_{i=1}^{t} \sqrt{s_i r_i n} \leq \sqrt{n} \sum_{i=1}^{t} \frac{s_i + r_i}{2} \leq n^{3/2}. \end{aligned}$$

□

Recall that the *inner product function* is a boolean function in $2m$ variables defined by

$$IP_{2m}(x_1, \ldots, x_m, y_1, \ldots, y_m) = \sum_{i=1}^{m} x_i y_i \bmod 2.$$

Since the graph G_f of $f = IP_{2m}$ is a Hadamard $n \times n$ graph with $n = 2^m$, Theorem 10.26 immediately yields

COROLLARY 10.27. *Any $\Sigma_3^{\oplus}$ circuit which has an arbitrary threshold gate on the top and computes IP_{2m} must have top fanin $\Omega(2^{m/2})$.*

10.4.2. Small depth-2 circuits for Ramsey graphs. Results above could wake an impression that Ramsey type graphs—that is graphs without large cliques in them and in their complements—could be good candidates of graphs requiring large depth-3 circuits. We will now show that this is not the case!

THEOREM 10.28. *There exist bipartite $m \times m$ graphs H such that*

a. both H and $\overline{H}$ are $K_{t,t}$-free for $t = 2\log_2 m$, but

b. H can be represented as a parity of $2\log_2 m$ OR gates.

[1] A *bipartite complement* $\overline{H}$ of a bipartite graph H is obtained by complementing its adjacency matrix.

PROOF. Let $\mathbb{F}_2 = GF(2)$ and r be a sufficiently large even integer. With every subset $S \subseteq \mathbb{F}_2^r$ we associate a bipartite graph $H_S \subseteq S \times S$ such that two vertices $u \in S$ and $v \in S$ are adjacent if and only if $\langle u, v\rangle = 1$, where $\langle u, v\rangle$ is the scalar product over $GF(2)$. Such graphs are known as *Sylvester graphs*. For $n = 2^r$, the graph H_S with $S = \mathbb{F}_2^r$ is denoted by H_n. We first show that H_n can be represented as a parity of $r = \log_2 n$ OR gates:

$$\bigoplus_{i=1}^{r} \bigvee_{v \in I_i} x_v \tag{10.6}$$

with $I_i = \{v \mid v(i) = 0\}$. Indeed, two vertices $u \in S$ and $v \in S$ are adjacent in H_n iff[2] $|v \wedge u|$ is odd iff $r - |u \wedge v|$ is odd iff the number of sets I_i containing at least one of u and v is odd iff the number of clauses $\bigvee_{v \in I_i} x_v$ accepting the pair uv is odd.

We are now going to show that H_n contains an induced $m \times m$ subgraph H_S with $m = \sqrt{n}$ satisfying the first claim of Theorem 10.28. The fact that H_S is an *induced* subgraph implies that (10.6) is also a representation of H_S: just set to 0 all variables x_v with $v \notin S$.

To prove that such a subgraph exists, we first establish one Ramsey type property of graphs H_S for arbitrary subsets $S \subseteq \mathbb{F}_2^r$.

LEMMA 10.29. *Suppose every vector space $V \subseteq \mathbb{F}_2^r$ of dimension $\lfloor (r+1)/2 \rfloor$ intersects S in less than t elements. Then neither H_S nor the bipartite complement $\overline{H}_S$ contains $K_{t,t}$.*

PROOF. The proof is based on the observation that any copy of $K_{t,t}$ in H_S would give us a pair of subsets X and Y of S of size t such that $\langle u, v\rangle = 1$ for all $u \in X$ and $v \in Y$. Viewing the vectors in X as the rows of the coefficient matrix and the vectors in Y as unknowns, we obtain that the sum $\dim(X') + \dim(Y')$ of the dimensions of vector spaces X' and Y', spanned by X and by Y, cannot exceed $r + 1$. Hence, at least one of these dimensions is at most $(r+1)/2$, implying that either $|X' \cap S| < t$ or $|Y' \cap S| < t$. However, this is impossible because both X' and Y' contain subsets X and Y of S of size t. □

It remains therefore to show that a subset $S \subseteq \mathbb{F}_2^r$ of size $|S| = 2^{r/2} = \sqrt{n}$ satisfying the condition of Lemma 10.29 exists. We show this by probabilistic arguments. For this, we use the following versions of Chernoff's inequalities: if X is the sum of n independent Bernoulli random variables with the success probability p, then

$$\Pr[|X| \le (1-c)pn] \le e^{-c^2 pn/2} \quad \text{for } 0 < c \le 1,$$

and

$$\Pr[|X| \ge cpn] \le 2^{-cpn} \quad \text{for } c > 2e.$$

Let $\mathbf{S} \subseteq \mathbb{F}_2^r$ be a random subset where each vector $u \in \mathbb{F}_2^r$ is included in $\mathbf{S}$ independently with probability $p = 2^{1-r/2} = 2/\sqrt{n}$. By Chernoff's inequality, $|\mathbf{S}| \ge pn/2 = 2^{r/2}$ with probability at least $1 - e^{-\Omega(pN)} = 1 - o(1)$.

Let now $V \subseteq \mathbb{F}_2^r$ be a subspace of $\mathbb{F}_2^r$ of dimension $\lfloor (r+1)/2 \rfloor = r/2$ (remember that r is even). Then $|V| = 2^{r/2} = \sqrt{n}$ and we may expect $p|V| = 2$ elements in $|\mathbf{S} \cap V|$. By Chernoff's inequality, $\Pr[|\mathbf{S} \cap V| \ge 2c] \le 2^{-2c}$ holds for any $c > 2e$. The number of vector spaces in $\mathbb{F}_2^r$ of dimension $r/2$ does not exceed $\binom{r}{r/2} \le 2^r/\sqrt{r}$. We can therefore take $c = r/2$ and conclude that the set $\mathbf{S}$ intersects some $r/2$-dimensional vector space V in $2c = r$ or more elements with probability at most $2^{r-(\log r)/2-r} = r^{-1/2} = o(1)$.

[2] $|v \wedge u|$ is the number of common 1's of vectors u and v.

Hence, with probability $1-o(1)$ the set $\boldsymbol{S}$ has cardinality at least $2^{r/2}$ and $|\boldsymbol{S}\cap V| < r$ for every $r/2$-dimensional vector space V. Fix such a set S' and take an arbitrary subset $S \subseteq S'$ of cardinality $|S| = 2^{r/2}$. By Lemma 10.29, neither H_S nor $\overline{H}_S$ contains a copy of $K_{r,r}$. □

10.5. Depth-3 threshold circuits

We now consider depth-3 circuits whose inputs are literals and gates are unbounded fanin threshold functions

$$Th_k^m(x_1,\ldots,x_m) = 1 \quad \text{iff} \quad x_1 + x_2 + \cdots + x_m \geq k\,.$$

These circuits are important by at least two reasons. The first reason is that threshold circuits are closely related to neural nets, an active area in computer science. The second reason (important in the context of circuit complexity) is that such circuits are unexpectedly powerful. Perhaps the most impressive result along these lines is due to Yao (1990) who showed that the whole class ACC is doable by depth-3 threshold circuits of

a. size $2^{(\log n)^{O(1)}}$ and
b. AND gates of fanin at most $(\log n)^{O(1)}$ at the bottom.

The class ACC consist of all boolean functions computable by constant-depth polynomial size circuits with NOT and unbounded fanin AND, OR and MOD_m gates for an arbitrary but fixed m. The function MOD_m computes 1 iff the number of 1's in the inputs vector is divisible by m.

Exponential lower bounds for ACC circuits are only known when m is a prime power (we will show this for $m = 3$ in the next chapter (see Section 11.4). But no such bound is known for a composite number m, say, for $m = 6$. This is why depth-3 threshold circuits with AND gates at the bottom are of particular interest. Below we will prove the largest known (superpolynomial) lower bound for such circuits.

Our starting point is the following theorem due to Hastad and Goldmann (1991), derived using a powerful result in multiparty communication complexity due to Babai, Nisan and Szegedy (1992); see Lemma 9.13 in Section 9.6 for the proof.

Recall that the generalized inner product function is defined by:

$$GIP_{n,s}(x) = \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{s} x_{ij}\,.$$

THEOREM 10.30 (Hastad–Goldmann). *Any depth-3 threshold circuits which computes $GIP_{n,s}$ and has bottom fanin at most $s-1$, must be of size* $\exp(\Omega(n/s4^s))$.

The consequence of this theorem is that the generalized inner product circuit requires depth-3 circuits of exponential size, as long as bottom fanin is smaller than $\log n$. We state this observation as

COROLLARY 10.31. *Any depth-3 threshold circuit which computes $GIP_{n,\log n}$ and has bottom fanin at most* $(\log n)/3$*, must be of size* $\exp(n^{\Omega(1)})$.

We are now going to use this result to prove a super-polynomial lower bound in the case when bottom gates are AND gates of arbitrary fanin. For this, consider now

the following boolean function

$$f_n(x) = \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{\log n} \bigoplus_{k=1}^{n} x_{ijk}\,.$$

THEOREM 10.32. *Any depth-3 threshold circuit which computes $f_n(x)$ and has unbounded fanin AND gates at the bottom, must be of size $n^{\Omega(\log n)}$.*

PROOF. Let C be a depth-3 threshold circuit computing $f_n(x)$. The strategy of the proof is to hit C with a random restriction in order to reduce the bottom fanin. Then we apply Corollary 10.31 to the resulting sub-circuit.

Set $p := (2\ln n)/n$. Let ϱ be the random restriction which assigns independently each variable to $*$ with probability p, and to $0,1$ with probabilities $(1-p)/2$. Given a boolean function g in n variables and a restriction ϱ, we will denote by $g\restriction_\varrho$ the function we get by doing the substitutions prescribed by ϱ.

Let K be a monomial, that is, a conjunction of literals. Denote by $|K|$ the number of literals in K. We are going to show that for each K we have

$$\Pr[|K\restriction_\varrho| \geq \tfrac{1}{3}\log n] \leq n^{-\Omega(\log n)}\,. \tag{10.7}$$

To show this, consider two cases.

Case 1: $|K| \leq (\log n)^2$. In this case we have

$$\Pr[|K\restriction_\varrho| \geq \tfrac{1}{3}\log n] \leq \binom{(\log n)^2}{\frac{1}{3}\log n} \cdot p^{\frac{1}{3}\log n} \leq O(p\log n)^{(\log n)/3} \leq n^{-\Omega(\log n)}\,.$$

Case 2: $|K| \geq (\log n)^2$. In this case we have

$$\Pr[|K\restriction_\varrho| \geq \tfrac{1}{3}\log n] \leq \Pr[K\restriction_\varrho \not\equiv 0] = \left(\frac{1+p}{2}\right)^{|K|} \leq n^{-\Omega(\log n)}\,.$$

Now, when we have (10.7), the reduction to Corollary 10.31 becomes easy. Namely, if our original circuit C would have size at most $n^{\varepsilon\log n}$ for a sufficiently small $\varepsilon > 0$ then, by (10.7), the probability that $C\restriction_\varrho$ has an AND gate on the bottom level of fanin larger than $\frac{1}{3}\log n$ would tend to 0. On the other hand, we have $n\log n$ sums $g(x) = \bigoplus_{k=1}^{n} x_{ijk}$ in f_n, and the probability that some of them will be evaluated by ϱ to a constant, is also at most

$$(1-p)^n n\log n \leq e^{-pn} n\log n = e^{-2\ln n} n\log n = \frac{\log n}{n} \to 0\,.$$

So, there exists an assignment ϱ such that both these events happen. That is, after this assignment ϱ we are left with a depth-3 threshold circuit C' which has bottom fanin at most $\frac{1}{3}\log n$ and computes a subfunction f'_n of f_n where none of the sums $g(x)$ is set to a constant. By setting (in necessary) some more variables to constant, we will obtain a circuit of bottom fanin at most $\frac{1}{3}\log n$ computing $GIP_{n,\log n}$. By Corollary 10.31, this is only possible if size(C'), and hence also size(C), is at least $\exp(n^{\Omega(1)})$, a contradiction with our assumption that size$(C) \leq n^{\varepsilon\log n}$. □

The reason why Theorem 10.32 does not imply large lower bounds for ACC circuits[3] is that Yao's reduction (mentioned above) requires much larger lower bounds, namely, bound of the form $\exp((\log n)^\alpha)$ for $\alpha \to \infty$.

[3]And could not imply since, by its definition, the function f_n itself can be computed using $n^2\log n$ AND and MOD_2 gates.

Exercises

EX. 10.1. For a bipartite graph G, let (as before) $\mathrm{cnf}(G)$ denote the smallest number of clauses in a monotone CNF representing G. Define the *intersection number* $\mathrm{int}(G)$ of G as the smallest number r for which it is possible to assign each vertex v a subset $S_v \subseteq \{1,\ldots,r\}$ such that u and v are adjacent in G iff $S_u \cap S_v = \emptyset$.

Prove that $\mathrm{cnf}(G) = \mathrm{int}(G)$.

Hint: Given a monotone CNF $C_1 \wedge \cdots \wedge C_r$, let $S_u = \{i \mid x_u \notin C_i\}$.

EX. 10.2. Show that a bipartite graph can be represented by a monotone Σ_3 circuits with top fanin s and middle fanin r iff it is possible to assign each vertex v an $s \times r$ $(0,1)$ matrix A_v such that u and v are adjacent in G iff the product-matrix $A_u \cdot A_v^\top$. (over the reals) has at least one 0 on the diagonal. *Hint*: Previous exercise.

EX. 10.3. Show that almost all bipartite $n \times n$ graphs require monotoneΣ_3 circuits of size $\Omega(\sqrt{n})$. *Hint*: Previous exercise.

EX. 10.4. A $\oplus$-decision tree for a boolean function $f(x_1,\ldots,x_m)$ is a binary tree whose internal nodes are labeled by subsets $S \subseteq [m]$ and whose leaves have labels from $\{0,1\}$. If a node has label S then the test performed at that node is to examine the parity $\bigoplus_{i\in S} x_i$. If the result is 0, one descends into the left subtree, whereas if the result is 1, one descends into the right subtree. The label of the leaf so reached is the value of the function (on that particular input). Let $DISJ_{2m}(x,y)$ be a boolean function in $2m$ variables defined by $DISJ_{2m}(x,y) = 1$ iff $x_i y_i = 0$ for all $i = 1,\ldots,m$.

Show that any $\oplus$-decision tree for $DISJ_{2m}$ requires $2^{\Omega(m)}$ leaves.

Hint: Transform the decision tree into a $\Sigma_3^\oplus$ circuit.

EX. 10.5. **Research problem.** Prove or disprove: there exists a bipartite $2^m \times 2^m$ graph G such that G can be represented by a monotone Σ_3 circuit of size $2^{\mathrm{polylog}(m)}$, but its bipartite complement $\overline{G}$ cannot be represented by a monotone Σ_3 circuit of such size.

Comment: Note that here G needs not be explicit—a mere existence would be enough! This would separate the second level of the communication complexity hierarchy introduced by Babai, Frankl and Simon (1986), and thus, solve an old problem in communication complexity.

Bibliographic Notes

Lemma 10.2 is proved in Valiant (1977). Lemma 10.5 was proved by Shi-Chun Tsai (2001). Results of Sections 10.3–10.3.2 were obtained by Hastad et. al (1995). Results of Section 10.4.1 were proved in [**81**]. Lemma 10.29 is due to Pudlák and Rödl (2004). Theorem 10.32 was proved by Razborov and Wigderson (1993).

CHAPTER 11

Large Depth Circuits

We now consider circuit of depth $d \geq 3$. As before, gates are unbounded fanin ORs and ANDs, and inputs are variables and their negations. We may assume that the underlying graph is layered so that: (i) inputs for gates on one layer are gates from the previous layer, and (ii) each layer consists of either OR gates or of AND gates. Hence, we have d alternating layers of OR and AND gates. Moreover, the first two (nearest to inputs) layers consist of CNFs (or of DNFs).

Lower bounds for such circuits are proved by reducing the depth one by one, until a circuit of depth-2 (or depth-1) remains. The key is the so-called Switching Lemma which allows to replace a CNF on the first two layers by a DNF, thus reducing the depth by 1. This is achieved by setting some variables to constants. If the total number of gates in a circuit is not large enough and the depth is constant, then we will end with a circuit computing a constant function, although a fair number of variables were not set to constants. For functions, like the Parity function, this yields the desired contradiction.

11.1. Switching lemma for non-monotone forms

Recall that a boolean function is a *t-CNF* function if it can be written as an AND of an arbitrary number of clauses, each being an OR of at most t literals. Dually, a boolean function is an *s-DNF* if it can be written as an OR of an arbitrary number of monomials, each being an AND of at most s literals.

Suppose we have a t-CNF function. Our goal is to find its dual representation as an s-DNF with s as small as possible. If we just multiply the clauses we can get very long monomials, much longer than s. So, the function itself may not be an s-DNF. We can try to assign constants 0 and 1 to some variables and "kill off" all long monomials (i.e., evaluate them to 0). If we set some variable x_i, say, to 1, then two things will happen: the literal $\overline{x}_i$ gets value 0 and disappears from all clauses, and all the clauses containing the literal x_i disappear (they get value 1).

Of course, if we set all variables to constants, then we are done – there will remain no monomials at all. The question becomes interesting if we must leave some fairly large number of variables not assigned. This question is answered by the following lemma.

Recall that a *restriction* is a map ϱ of the set of variables to the set $\{0, 1, *\}$. The restriction ϱ can be applied to a function $f = f(x_1, \ldots, x_n)$, then we get the function $f\restriction_\varrho$ (called a *subfunction* of f) where the variables are set according to ϱ, and $\varrho(x_i) = *$ means that x_i is left unassigned.

LEMMA 11.1 (Switching Lemma). *Let f be a t-CNF on n variables, and let ϱ be a random restriction leaving a fraction p of variables unassigned. Then*

$$\Pr[f\restriction_\varrho \textit{ is not an } s\textit{-DNF }] \leq (\gamma p t)^s\,, \tag{11.1}$$

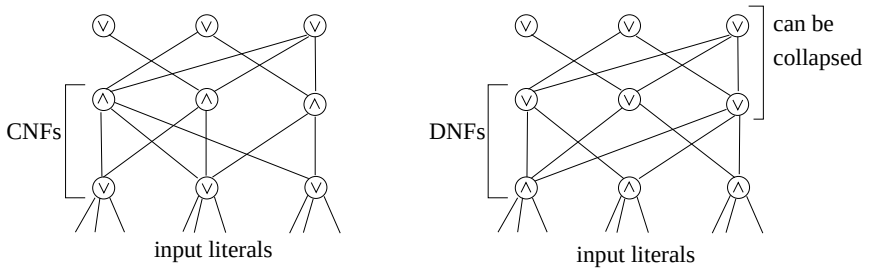


FIGURE 1. After the Switching Lemma is applied, levels 2 and 3 can be collapsed into one level.

where γ is an absolute constant.

We will prove this lemma in the next section. Now we apply it to show that the parity function cannot be computed by constant depth circuits of polynomial size.

THEOREM 11.2. *Any depth-d circuit with unbounded fanin AND and OR gates computing a parity of n variables requires $2^{\Omega(n^{1/(d-1)})}$ gates.*

PROOF. Let C be a depth-d circuit for parity of size S. Our first goal is to reduce the fanin of gates on the first (next to the inputs) layer. Suppose that they are OR gates; a symmetric argument applies if they are AND gates.

We think of each such gate as a 1-DNF. We apply the Switching Lemma with $t = 1$, $s = 2\log_2 S$ and $p = 1/(2\gamma)$, and deduce that after a random restriction each of the these 1-DNFs becomes an s-CNF (in fact, a single clause of length $\leq s$) with probability at least

$$1 - (\gamma p t)^s = 1 - 2^s = 1 - S^{-2}.$$

Since we have at most S of the these 1-DNFs, this in particular implies that there is a restriction that makes all these 1-DNFs expressible as an OR of at most s input literals. We apply such a restriction, and what we obtain is a circuit of depth d such that each bottom gate has fanin at most

$$b := 2\log_2 S$$

and the circuit still computes parity of $n' = n/(2\gamma)$ variables.

We now apply the Switching Lemma to the first two bottom layers with

$$p = 1/(2\gamma b)$$

and both s and t equal to b. We get that, for each AND gate on layer 2, after the restriction the gate can be replaced by an s-DNF with probability at least $1 - 2^{-b} = 1 - S^{-2}$. Hence, there is a restriction for which this is true for all the at most S gates at layer 2. We apply this restriction, replace each layer-2 gate with a s-DNF, and and use associativity to collapse the OR gate of each DNF into an OR gates of the second layer of the original circuit. This way we collapse layer 2 with layer 3 (see Fig. 1).

Now we have a circuit of depth $d - 1$ that computes parity of

$$pn' = \frac{n}{4\gamma^2 b}$$

variables, and such that every bottom gate has fanin at most b.

If we repeat the same argument another $d-3$ times, we will eventually end up with a circuit of depth 2 such that the fanin of the bottom gates is at most $b = 2\log_2 S$ and the circuit computes parity of

$$m = \frac{n}{(4\gamma^2 b)^{d-2}} = \frac{n}{(4\gamma^2 \log_2 S)^{d-2}}$$

variables. Since in any DNF (or CNF) computing parity of m variables, each monomial (clause) must have length m, this implies that

$$2\log_2 S = b \geq m = \frac{n}{(4\gamma^2 \log_2 S))^{d-2}},$$

from which the desired lower bound $S = 2^{\Omega(n^{1/(d-1)})}$ follows. □

Note that the only property of the parity function, we used in the proof, is that the function cannot be made constant by setting fewer than $n-1$ variables to constants. Hence, we in fact have a more general result:

THEOREM 11.3. *If a boolean function f in n variables cannot be made constant by setting all but $\Omega(n^{1/d})$ variables to constants, then any depth-$(d+1)$ circuit with unbounded fanin AND and OR gates computing f requires $2^{\Omega(n^{1/d})}$ gates.*

11.2. Razborov's proof of switching lemma

We denote by $\mathscr{R}^\ell$ the set of all restrictions assigning exactly ℓ stars. Hence

$$|\mathscr{R}^\ell| = \binom{n}{\ell} 2^{n-\ell}.$$

A *minterm* of f is a restriction ϱ such that $f\restriction_\varrho \equiv 1$ and which is minimal in the sense that un-specifying every single value $\varrho(i) \in \{0,1\}$ already violates this property. The *length* of a minterm is the number $n - |\varrho^{-1}(*)|$ of assigned variables.

Let $\min(f)$ be the length of the longest minterm of f, and let

$$\mathrm{Bad}^\ell(s,t) := \left\{\varrho \in \mathscr{R}^\ell \,:\, \min(f\restriction_\varrho) > s\right\}.$$

In particular, $\mathrm{Bad}^\ell(s,t)$ contains all the restrictions $\varrho \in \mathscr{R}^\ell$ for which $f\restriction_\varrho$ is not an s-DNF.

LEMMA 11.4. *Let f be a t-CNF on n variables. Then, for any $1 \leq s \leq \ell \leq n$,*

$$|\mathrm{Bad}^\ell(s,t)| \leq |\mathscr{R}^{\ell-s}| \cdot (4t)^s. \tag{11.2}$$

Before we merge into the proof of Lemma 11.4, let us show that it indeed implies Switching Lemma. To show this, take a random restriction ϱ in $\mathscr{R}^\ell$ for $\ell = pn$. Then, by Lemma 11.4, the probability that $f\restriction_\varrho$ is *not* an s-Or-And function, is at most

$$\frac{|\mathrm{Bad}^\ell(s,t)|}{|\mathscr{R}^\ell|} \leq \frac{\binom{n}{\ell-s}2^{n-\ell+s}(4t)^s}{\binom{n}{\ell}2^{n-\ell}} \leq \left(\frac{8t\ell}{n-\ell}\right)^s = \left(\frac{8tp}{(1-p)}\right)^s$$

which is at most[1] $(16pt)^s$ as long as $p \leq 2/3$.

We now turn to the proof of Lemma 11.4. A general idea is to apply the following:

[1] More precise calculations yield $(7pt)^s$ but we will not care about this.

Coding Principle: In order to prove that some set A is not very large try to construct an injective mapping $\mathrm{Code} : A \to B$ of A to some set B which is a priori known to be small, and *give a way how to retrieve* the element $a \in A$ from its code $\mathrm{Code}(a)$. Then $|A| \leq |B|$.

PROOF OF LEMMA 11.4. Let F be a t-CNF formula. Fix an order of its clauses and fix an order of literals in each clause. Suppose that ϱ is a bad restriction, i.e., $\varrho \in \mathrm{Bad}^{\ell}(s,t)$. Then there must be a minterm π of $F\restriction_{\varrho}$ whose length is at least s. We truncate π so that it has length exactly s.

Our goal is to show, how using the minterm π and the formula F plus a "small" additional information, to reconstruct the restriction ϱ.

Consider the first clause C_1 of F that is not set to 1 by ϱ; hence, ϱ does not set any literal of C_1 to 1 and does not set all literals of C_1 to 0. Let π_1 be the portion of π that assigns values to variables in C_1 (actually, to variables in $C_1\restriction_{\varrho}$ since π is a minterm of $F\restriction_{\varrho}$, not of the whole formula F). Let also $\overline{\pi}_1$ be the uniquely determined restriction which has the same domain as π_1 and does *not* set the clause $C_1\restriction_{\varrho}$ to 1. That is, $\overline{\pi}_1$ evaluates all the literals "touched" by π_1 to 0.

Define the string $\boldsymbol{a}_1 \in \{0,1\}^t$ based on the fixed ordering of the variables in clause C_1 by letting the j-th component of $\boldsymbol{a}_1$ be 1 if and only if the j-th variable in C_1 is set by π_1 (and hence, also by $\overline{\pi}_1$). Note that since $C_1\restriction_{\varrho}$ is not an empty clause there is at least one 1 in $\boldsymbol{a}_1$. Here is a typical example:

C_1	=	x_3	$\vee$	$\overline{x}_4$	$\vee$	x_6	$\vee$	x_7	$\vee$	x_{12}
π_1	=	$*$		1		$*$		1		0
$\overline{\pi}_1$	=	$*$		1		$*$		0		0
$\boldsymbol{a}_1$	=	0		1		0		1		1

The main property of the string $\boldsymbol{a}_1$ is that *knowing* C_1 *and* $\boldsymbol{a}_1$ *we can reconstruct* $\overline{\pi}_1$.

Now, if $\pi_1 \neq \pi$, we repeat the above argument with $\pi - \pi_1$ in place of π, $\varrho\pi_1$ in place of ϱ and find a clause C_2 which is the first clause of F not set to 1 by $\varrho\pi_1$. Based on this we generate π_2, $\overline{\pi}_2$ and $\boldsymbol{a}_2$ as before. Continuing this way we get a sequence of clauses $C_1, C_2, \ldots$. Each C_i contains some variable that was not in C_j for $j < i$, so we must stop after we have identified at most s clauses. Say we have identified m clauses. Hence, $\pi = \pi_1\pi_2\ldots\pi_m$.

Let $\boldsymbol{b} \in \{0,1\}^s$ be a vector that indicates for each variable set by π (which are the same as those set by $\overline{\pi}$) whether it is set to the same value as $\overline{\pi}$ sets it. (Recall that π_i must set at least one literal of C_i to 1 and may set some of them to 0, whereas $\overline{\pi}_i$ sets all these literals to 0.) We encode the restriction ϱ by a string

$$\mathrm{Code}(\varrho) := \left\langle \varrho\overline{\pi}_1\overline{\pi}_2\ldots\overline{\pi}_m,\ \boldsymbol{a}_1,\ldots,\boldsymbol{a}_m,\ \boldsymbol{b}\right\rangle.$$

Our goal is to show that the mapping $\varrho \mapsto \mathrm{Code}(\varrho)$ is injective. For this, it is enough to show how to reconstruct ϱ uniquely, given $\mathrm{Code}(\varrho)$.

First note that it is easy to reconstruct $\overline{\pi}_1$. Identify the first clause of F that is not set to 1 by $\varrho\overline{\pi}_1\overline{\pi}_2\ldots\overline{\pi}_m$. Since none of the $\overline{\pi}_i$ sets a clause to 1, this must be clause C_1. Now use $\boldsymbol{a}_1$ to identify the variables of C_1 that are set by $\overline{\pi}_1$, and use $\boldsymbol{b}$ to identify how π_1 would set these variables. Thus we have reconstructed both sub-restriction π_1 and $\overline{\pi}_1$. Knowing these sub-restrictions and the entire restriction $\varrho\overline{\pi}_1\overline{\pi}_2\ldots\overline{\pi}_m$ we can construct the restriction $\varrho\pi_1\overline{\pi}_2\ldots\overline{\pi}_m$.

Now we can identify C_2: it is the first clause of F that is not set to 1 by $\varrho\pi_1\overline{\pi}_2\dots\overline{\pi}_m$. Then we use $\boldsymbol{a}_2$ to identify the variables of C_2 set by $\overline{\pi}_2$, and use $\boldsymbol{b}$ to identify how π_2 would set these variables.

Continuing this way, we can reconstruct the restriction $\overline{\pi}_1\overline{\pi}_2\dots\overline{\pi}_m$ and thus the original restriction ϱ.

To finish the proof of Lemma 11.4, it is enough to upper bound the range of Code(ϱ). First, observe that restrictions $\varrho\overline{\pi}_1\overline{\pi}_2\dots\overline{\pi}_m$ belong to $\mathscr{R}^{\ell-s}$. Hence, the number of such restrictions does not exceed $|\mathscr{R}^{\ell-s}|$. The number of strings $\boldsymbol{b}\in\{0,1\}^s$ is clearly at most 2^s. Finally, each $(\boldsymbol{a}_1,\dots,\boldsymbol{a}_m)$ is a string in $\{0,1\}^{mt}$ with the property that each $\boldsymbol{a}_j$ has at least one 1 and the total number of 1's in all $\boldsymbol{a}_j$ is s. The number of such strings $(\boldsymbol{a}_1,\dots,\boldsymbol{a}_m)$ with k_i ones in $\boldsymbol{a}_i$ is

$$\prod_{i=1}^{m}\binom{t}{k_i}\le\prod_{i=1}^{m}t^{k_i}=t^{\sum_{i=1}^m k_i}=t^s\,.$$

The number of integer solutions $k_1,\dots,k_m\ge 1$ of $k_1+\dots+k_m=s$ is $\binom{s-1}{m-1}\le 2^s$ (show this!). Thus, the range of Code(ϱ), and hence, the number $|\mathrm{Bad}^\ell(s,t)|$ of restrictions $\varrho\in\mathscr{R}^\ell$ for which $\min(f\restriction_\varrho)>s$, does not exceed $|\mathscr{R}^{\ell-s}|\times(4t)^s$, as desired. □

11.3. Circuits with parity gates

We already know that Parity function cannot be computed by constant depth circuits using a polynomial number of unbounded fanin AND and OR gates. Let us therefore extend the model and allow Parity functions be also used as gates. What functions are then difficult to compute? We will show that such is the Majority function Maj_n which accepts an input vector of length n iff it has at least as many 1's as 0's. The general idea is similar as in the case of monotone circuits, but this time with an algebraic "flavour." The proof consists of two steps:

a. prove that the majority function is hard to approximate by such polynomials;
b. show that functions, computable by small circuits, can be approximated by low degree polynomials.

We first establish the first goal (a). In fact we will apply this argument not to Majority function itself but rather to a closely related function, the *k-threshold function* Th^n_k. This function is 1 when at least k of the inputs are 1. Note that each such function is a subfunction of the Majority function in $2n$ variables: just set some $n-k$ variables to 1 and some k of the remaining variables to 0. It is therefore enough to prove a hight lower bound on Th^n_k for at least one threshold value $1\le k\le n$. We will consider $k=\lceil(n+h+1)/2\rceil$ for an appropriate h.

To achieve the first goal (a), we have to show that any polynomial of low degree over $GF(2)$ has to differ from k-threshold function on a large fraction of inputs. Recall that the degree of a multivariate polynomial over $\mathbb{F}_2=GF(2)$ is the length of (the number of variables in) its longest monomial.

LEMMA 11.5. *Let $n/2\le k\le n$. Every polynomial $p(x)$ of degree at most $2k-n-1$ differs from the k-threshold function on at least $\binom{n}{k}$ inputs:*

$$\#\{x\mid p(x)\ne Th^n_k(x)\}\ge\binom{n}{k}\,.$$

PROOF. Let g be a polynomial of degree $d \le 2k-n-1$ over $\mathbb{F}_2$ and let U denote the set of all vectors where it differs from Th_k^n. Let A denote the set of all 0-1 vectors of length n containing exactly k 1's.

Consider the 0-1 matrix $M = (m_{a,u})$ whose rows are indexed by the members of A, columns are indexed by the members of U, and $m_{a,u} = 1$ if and only if $a \ge u$. For two vectors a and b we denote by $a \wedge b$ the coordinate-wise And of these vectors. Our goal is to prove that the columns of M span the whole linear space; since the dimension of this space is $|A| = \binom{n}{k}$, this will mean that we must have $|U| \ge \binom{n}{k}$ columns.

The fact that the columns of M span the whole linear space follows directly from the following claim saying that every unit vector lies in the span:

CLAIM 11.6. If $a \in A$ and $U_a = \{u \in U \mid m_{a,u} = 1\}$, then

$$\sum_{u \in U_a} m_{b,u} = \begin{cases} 1 & \text{if } b = a; \\ 0 & \text{if } b \ne a. \end{cases}$$

To prove the claim, observe that by the definition of U_a, we have (all sums are over $\mathbb{F}_2$):

$$\sum_{u \in U_a} m_{b,u} = \sum_{\substack{u \in U \\ u \le a \wedge b}} 1 = \sum_{x \le a \wedge b} \Big(Th_k^n(x) + g(x) \Big) = \sum_{x \le a \wedge b} Th_k^n(x) + \sum_{x \le a \wedge b} g(x).$$

The second term of this last expression is 0, since $a \wedge b$ has at least $d+1$ 1's. The first term is also 0 except if $a = b$.

This completes the proof of the claim, and thus, the proof of the lemma. □

Our next goal (b) is to show that, if a boolean f function can be computed by a small-depth circuit will a small number of AND, OR and Parity gates, then f can be approximated well enough by a low degree polynomial. This is done in a bottom-up manner. Input variables themselves are polynomials of degree 1, and need not be approximated. Also, since the degree is not increased by computing the sum, parity gates do not have to be approximated as well. Hence, it remains to show how to approximate AND and OR gates. For this, the following simple fact will be useful.

PROPOSITION 11.7. *Let $y \in \{0,1\}^m$ and let $S \subseteq \{1,\ldots,m\}$ be a random subset. If $y \ne \mathbf{0}$ then*

$$\Pr[\textstyle\sum_{i \in S} y_i \text{ is even}] = \frac{1}{2}.$$

PROOF. Say that a vector x is orthogonal to a vector y if their scalar product $\langle x, y \rangle = \sum_i x_i y_i \bmod 2$ over $GF(2)$ is equal to 0. By letting x to be the characteristic vector of S, it is enough to show that every vector $y \ne \mathbf{0}$ is orthogonal to exactly half of all vectors in $\{0,1\}^m$.

To show this, take an i for which $y_i = 1$ and split the m-cube $\{0,1\}^m$ into 2^{m-1} pairs x, x' that differ only in the ith coordinate. Since the vector y is orthogonal to exactly one vector from each pair, it follws that y is orthogonal to exactly a half of all vectors in $\{0,1\}^m$. □

If p and q are polynomials representing two functions, then $p \cdot q$ is the polynomial corresponding to their AND, and $(p \oplus 1)(q \oplus 1) \oplus 1$ is the polynomial corresponding to their OR. But since we have *unbounded* degree gates, the degree of AND and OR gates can greatly increase. We will therefore approximate these gates so that approximating

polynomial will have fairly low degree. In this section, all polynomials are polynomials over the field $GF(2)$.

LEMMA 11.8 (Approximation Lemma). *Let f be an OR of polynomials $g_1, \ldots, g_m$ of degree $\leq h$. Then, for every integer $r \geq 1$, there exists a polynomial p of degree $\leq rh$ such that*

$$\#\{x \mid p(x) \neq f(x)\} \leq 2^{n-r}.$$

By the duality, the same hold also for conjunctions (products) of polynomials.

PROOF. To construct the desired polynomial, approximating $f = \vee_{i=1}^{m} g_i$, randomly select r subsets $S_1, \ldots, S_r$ of $\{1, \ldots, m\}$, where each i is included in S_j with probability $1/2$. Let

$$f_j = \bigoplus_{i \in S_j} g_i$$

and consider

$$f' = \bigvee_{j=1}^{r} f_j = \bigvee_{j=1}^{r} \bigoplus_{i \in S_j} g_i .$$

We claim that the probability that f' satisfies the requirements of the lemma is non-zero. Since f' is an OR of r polynomials of degree at most h, f' itself can be written as a polynomial of degree at most rh using the rule $f \vee g = (f \oplus 1) \wedge (g \oplus 1) \oplus 1$.

Hence, it remains to show that f' differs from f in at most 2^{n-r} inputs. To show this, take an arbitrary input $a \in \{0,1\}^n$. We claim that the probability that $f'(a) \neq f(a)$ is at most 2^{-r}. To see this consider two cases.

If $g_i(a) = 0$ for every i, then both $f(a) = 0$ and $f'(a) = 0$. On the other hand, if there exists an i for which $g_i(a) = 1$, then $f(a) = 1$ and (by Proposition 11.7 applied with $y_i := g_i(a)$) for each j, $f_j(a) = 0$ independently with probability at most $1/2$. Therefore, $f'(a) = 0$ with probability at most 2^{-r}, and the expected number of inputs on which $f' \neq f$ is at most 2^{n-r}. Hence for at least one particular choice of the sets S_j, the polynomial f' differs from f on at most 2^{n-r} inputs. □

LEMMA 11.9. *If a boolean function f in n variables can be computed by an unbounded fanin circuit of depth d over the basis $\{\wedge, \vee, \oplus\}$, then for any integer $r \geq 1$, there exists a polynomial g of degree at most r^d such that g differs from f on at most $\ell \cdot 2^{n-r}$ inputs.*

PROOF. Apply Lemma 11.8 to approximate the OR and AND gates in the circuit. The functions computed by the gates at the ith level will be approximated by polynomials of degree at most r^i. Since we have only d levels, the function f computed at the top gate will be approximated by a polynomial f' of degree at most r^d. Since, by Lemma 11.8, at each of ℓ gates we have introduced at most 2^{n-r} errors, f' can differ from f on at most $\ell 2^{n-r}$ inputs. □

THEOREM 11.10. *Every unbounded fanin depth-d circuit over $\{\wedge, \vee, \oplus\}$ computing* Maj_n *requires $2^{\Omega(n^{1/2d})}$ gates.*

PROOF. By the remark above, it is enough to prove such a lower bound for a depth-d circuit computing a k-threshold function Th_k^n for some $n/2 \leq k \leq n$ (to be specified later). Take such a circuit of size ℓ computing Th_k^n. Lemmas 11.5 and 11.9 imply that

$$\ell \geq \binom{n}{k} 2^{r-n}.$$

Taking $r = \lfloor n^{1/(2d)} \rfloor$ and $k = \lceil (n + r^d + 1)/2 \rceil = \lceil (n + \sqrt{n} + 1)/2 \rceil$, the right hand side turns to $2^{\Omega(r)}$, and we are done. □

11.4. An algebraic lower bound for parity

A modular function MOD_p is the boolean function which is 1 iff the number of 1's in the input vector is divisible by p. Hence, MOD_2 is the parity function. We have seen that MOD_2 cannot be computed by a constant depth circuit with polynomial number of NOT, AND and OR gates. But what if, besides NOT, AND and OR gates, we allow some modular gates MOD_p with $p \geq 3$ be used as gates—can then MOD_2 be easier computed?

It turns out that the use of gates MOD_p, where $p \geq 3$ is a prime power, does not help to compute the Parity MOD_2 more efficiently.[2] We will show this for the special case $p = 3$.

Under a *modular circuit* we will understand an unbounded fanin circuit with AND, OR, NOT and MOD_3 gates. The general proof idea will be again the same: show that functions, computable by small circuits with MOD_3 gates, can be approximated by low degree polynomials over $GF(3)$, and prove that the parity function is hard to approximate by such polynomials.

DEFINITION 11.11. A *b-approximator* is a polynomial on the input variables $x_1, \ldots, x_n$ of degree at most b over $GF(3)$, the three element field $\{-1, 0, 1\}$, where on inputs from $\{0, 1\}$ it takes values from $\{0, 1\}$.

The following lemma states that, if a boolean function can be computed by a modular circuit of small depth using a small number of gates, then this function can be approximated well enough by a small degree polynomial over $GF(3)$.

LEMMA 11.12. *If a boolean function f can be computed by a modular depth-d circuit of size ℓ then, for every integer $r \geq 1$, there is a $(2r)^d$-approximator which differs from f on at most $\ell \cdot 2^{n-r}$ inputs.*

PROOF. Take such a circuit C computing f. We inductively assign to each gate of the circuit a particular approximator working up from inputs to the output. Each assignment introduces some error which is the number of output deviations between it and the result of applying the true operator at that gate to the approximators of the gates feeding into it, looking only at inputs drown from $\{0, 1\}^n$.

Approximators of input variables are variables themselves. If the gate g is a NOT gate and the unique gate feeding into it has b-approximator f, then we assign the b-approximator $1 - f$ to g. This approximator introduces no errors. If g is a MOD_3 gate and its inputs have b-approximators $f_1, \ldots, f_k$ then we assign the $2b$-approximator[3] $(\sum_{i=1}^{k} f_i)^2$ to g. Since $0^2 = 0$ and $(-1)^2 = 1$ this introduces no new errors, as well.

It remains to consider the case when g is an OR gate (the case of an AND gate is similar). So, let $g(x)$ be the function computed at an OR gate whose inputs have b-approximators $f_1, \ldots, f_m$.

CLAIM 11.13. For every integer $r \geq 1$, there exists a $2rb$-approximator p of g such that the number of inputs $x \in \{0, 1\}^n$ on which $p(x)$ differs from $\mathrm{OR}(f_1, \ldots, f_m)$ does not exceed 2^{n-r}.

[2] For other values of p, the smallest being $p = 6$, no explicit boolean function requiring a super-polynomial number of gates is known (Research Problem!).

[3] The only reason to take a square is to ensure that the resulting function takes boolean values 0 and 1.

PROOF. For every integer $r \geq 1$, we construct a $2rb$-approximator for $g(x)$ as follows. Randomly select r subsets $S_1, \ldots, S_r$ of $\{1, \ldots, m\}$, where each i is included in S_j with probability $1/2$. Let

$$f_i'(x) := \Big(\sum_{j \in S_i} f_j(x)\Big)^2$$

and consider a random polynomial

$$p(x) := 1 - \prod_{i=1}^{r} (1 - f_i'(x)).$$

The degree of $p(x)$ is at most $2rb$, as the degree of each f_i' is at most $2b$. Moreover, if $\mathrm{OR}(f_1, \ldots, f_m)$ outputs 0, then $p(x) = 0$. So, take an input x on which $\mathrm{OR}(f_1, \ldots, f_m)$ outputs 1, that is $f_i(x) = 1$ for at least one i. By Proposition 11.7, applied with $y_1 := f_1(x), \ldots, y_m := f_m(x)$, we obtain that

$$\Pr[f_i'(x) = 0] \leq \frac{1}{2}$$

for each $i = 1, \ldots, r$. Hence,

$$\Pr[p(x) = 0] = \Pr[f_i'(x) = 0 \text{ for all } i = 1, \ldots, r] \leq 2^{-r}.$$

By an averaging argument there must be a collection of the sets S_i so that the number of input settings, on which the OR of $f_1(x), \ldots, f_m(x)$ is 1 and $p(x) = 0$, is at most 2^{n-r}. □

Now we can finish the proof of Lemma 11.12 as follows. The inputs of our circuit computing f are assigned the corresponding 1-approximators. By Claim 11.13, each layer increases the degree of the approximators by a factor of at most $2r$, and each assignment of approximators contributes at most 2^{n-r} error. Since we only have d layers and only ℓ gates in total, the last gate will receive a $(2r)^d$-approximator which differs from f on at most $\ell \cdot 2^{n-r}$ input vectors. □

To apply Lemma 11.12 to the parity function, we have to show that $\mathrm{parity}(x)$ cannot be approximated well enough by small degree polynomials over $GF(3)$.

LEMMA 11.14. *Any $\sqrt{n}$-approximator must differ from* $\mathrm{parity}(x)$ *on at least* $0.15 \cdot 2^n$ *input vectors.*

PROOF. For this proof, we represent boolean values by 1 and -1 rather than 0 and 1. Namely, we replace each boolean variable x_i by a new variable $y_i = 1 - 2x_i$. Hence, $y_i = 1$ if $x_i = 0$, and $y_i = -1$ if $x_i = 1$. The parity function then turns to the product of the y_i:

$$\bigoplus_{i=1}^{n} x_i = 1 \quad \text{iff} \quad \prod_{i=1}^{n} y_i = -1.$$

Suppose that $p(y)$ is a polynomial over $GF(3)$ of degree at most $\sqrt{n}$. We need to show that this polynomial differs from $\prod_{i=1}^{n} y_i$ on at least 0.15 fraction of the vectors in $\{1, -1\}^n$.

Let $A = \{y \mid p(y) = \prod_{i=1}^{n} y_i\}$. We wish to show that A is small (has size at most $0.85 \cdot 2^n$) and will do this by upper bounding the number $|F|$ functions in the set F of all functions $f : A \to \{-1, 0, 1\}$: since $|F| = 3^{|A|}$ we may bound the size of A by showing that $|F|$ is small.

We claim that every function in F can be represented as a multilinear polynomial over $GF(3)$ of degree at most $(n+\sqrt{n})/2$. Suppose $f \in F$, and $M = \prod_{i\in S} y_i$ is a monomial in the representation of f. If $|S| > (n+\sqrt{n})/2$, we replace the monomial by

$$M' = \prod_{i\notin S} y_i \cdot p(y).$$

Since

$$\prod_{i\notin S} y_i \cdot \prod_{i=1}^{n} y_i = \prod_{i\in S} y_i \cdot \prod_{i\notin S} y_i^2 = \prod_{i\in S} y_i,$$

we have that $M'(y) = M(y)$ for all $y \in A$, and

$$\text{degree}(M') \le \frac{n-\sqrt{n}}{2} + \sqrt{n} = \frac{n+\sqrt{n}}{2}.$$

Thus, every function in F can be represented as a multilinear polynomial over $GF(3)$ of degree at most $(n+\sqrt{n})/2$.

The number of multilinear monomials of degree at most $(n+\sqrt{n})/2$ is

$$N = \sum_{i=0}^{\frac{n+\sqrt{n}}{2}} \binom{n}{i} \le 0.85 \cdot 2^n$$

for large n. Since, $|F| \le 3^N$, we conclude that

$$|A| = \log_3 |F| \le \log_3 N \le 0.85 \cdot 2^n. \qquad \square$$

Combining the two lemmas above we obtain the following

THEOREM 11.15. *Any modular depth-d circuit computing the parity function requires* $2^{\Omega(n^{1/2d})}$ *gates.*

PROOF. Let ℓ be the minimum size of a modular depth-d circuit computing parity(x). Taking $r = n^{1/2d}/2$ in Lemma 11.12, we obtain that then there must exist a $\sqrt{n}$-approximator $p(x)$ such that

$$\#\{x \mid p(x) \ne \text{parity}(x)\} \le \ell \cdot 2^{n-n^{1/2d}/2}.$$

But Lemma 11.14 implies that

$$\#\{x \mid p(x) \ne \text{parity}(x)\} \ge 0.15 \cdot 2^n,$$

and the desired lower bound on ℓ follows. $\square$

11.5. Rigid matrices require large circuits

We now consider boolean circuits computing $(0,1)$ matrices of some fixed in advance dimension. Inputs are *rectangular matrices*, that is, matrices of rank 1. Each of these matrices can be described by a Cartesian product $I \times J$ corresponding to its all-1 submatrix. Boolean operation on matrices are computed component-wise. Thus, each such circuit computes some matrix. As before, the *depth* of a circuit is the length of a longest path from an input to an output gate. The *size* is the number of gates.

Let $C_d(M)$ denote the smallest size of an unbounded fanin circuit of depth d over the basis $\{\&, \vee, \neg, \oplus\}$ computing the matrix M.

What matrices require large constant-depth circuits? We will show that such are matrices of high "rigidity".

The *rigidity* $\mathscr{R}_M(r)$ of a $(0,1)$ matrix M over $GF(2)$ is the smallest number of entries of A that must be changed in order to reduce its rank over $GF(2)$ until r. That is,

$$\mathscr{R}_M(r) = \min\{|B| : \mathrm{rk}(M \oplus B) \le r\},$$

where $|B|$ is the total number of 1s in B.

THEOREM 11.16. *Let M be an $n \times n$ $(0,1)$ matrix, and $d \ge 2$ an integer. If*

$$\mathscr{R}_M(r) \ge \frac{n^2}{\exp((\ln r)^{1/d})} \tag{11.3}$$

then

$$C_{d-1}(M) \ge 2^{\Omega((\ln r)^{1/d})}. \tag{11.4}$$

PROOF. We will again use the approximation method. This time we will approximate matrices, computed at intermediate gates of the circuit, by matrices of small rank. Set

$$\ell := \lfloor 2(\ln r)^{1/d} \rfloor.$$

Note that for $r = O(1)$ the theorem is obvious, so we assume that r and ℓ are large enough.

Suppose we have an unbounded fanin circuit over $\{\&, \vee, \neg, \oplus\}$ of depth at most d and size s computing the matrix M. We have to show that (11.3) implies

$$s \ge 2^{\Omega((\ln r)^{1/d})}. \tag{11.5}$$

At each gate v of the circuit some $(0,1)$ matrix A is computed. We inductively assign to each gate on the ith layer an *approximator*, which is a $(0,1)$ matrix $\widetilde{A}$ of rank

$$\mathrm{rk}(\widetilde{A}) \le (s+1)^{O(\ell^i)}. \tag{11.6}$$

As before, the assignments are done inductively, first to the inputs, then working up to the output. Each assignment introduces some *errors* which are the positions the approximator $\widetilde{A}$ differs from the matrix obtained by applying the true operator at that gate to the approximators of the gates feeding into it. Our goal is to assign approximators in such a way that:

$$\text{At each gate at most } n^2/2^\ell \text{ errors are introduced.} \tag{11.7}$$

We first show that (11.6) and (11.7) already imply the theorem. To see this, let $\widetilde{M}$ be the approximator of the matrix computed at the last gate.

If $\mathrm{rk}(\widetilde{M}) \ge r$ then (11.6) implies that $r \le (s+1)^{O(\ell^{d-1})}$, and since $\ell^{d-1} = \Omega((\ln r)^{1-1/d}))$ (by the choice of ℓ), the desired lower bound (11.5) on the size s follows.

If $\mathrm{rk}(\widetilde{M}) \le r$ then our assumption (11.3) implies that

$$|M - \widetilde{M}| \ge \mathscr{R}_M(r) \ge \frac{n^2}{\exp((\ln r)^{1/d})}.$$

On the other hand, (11.7) implies that

$$|M - \widetilde{M}| \le s \cdot n^2/2^\ell = s \cdot \frac{n^2}{\exp(2(\ln r)^{1/d})}.$$

Comparing these two estimates, we again obtain the desired lower bound (11.5) on the size s. It remains, therefore, to show how to assign approximators satisfying (11.6) and (11.7).

Approximators of input matrices are matrices themselves. Recall that these matrices have rank ≤ 1.

If the gate v is a NOT gate and the unique gate feeding into it has an approximator $\widetilde{A}$, then we assign the approximator $\neg\widetilde{A}$ to v. Since $\mathrm{rk}(\neg\widetilde{A}) \leq \mathrm{rk}(\widetilde{A}) + 1$, the rank condition (11.6) is fulfilled.

If the gate v is a parity gate, then let the approximator of v be just the sum modulo 2 of the approximators of all its $m \leq s$ inputs gates. The rank condition (11.6) is fulfilled by the subadditivity of rank:

$$\mathrm{rk}(\widetilde{A}) \leq m \cdot (s+1)^{O(\ell^{i-1})} \leq (s+1)^{O(\ell^i)}.$$

So far we have introduced no errors at all. The source of errors are, however, AND and OR gates. For these gates we use the following lemma whose proof is similar to that of the Approximation Lemma for multivariate polynomials (Lemma 11.8).

LEMMA 11.17 (Approximation Lemma for Matrices). *Let $\ell \geq 1$ be an integer. If $A = \bigvee_{i=1}^{h} A_i$ is an OR of $n \times n$ $(0,1)$ matrices, each of rank at most r, then there is a $(0,1)$ matrix C such that*

$$\mathrm{rk}(C) \leq 1 + (1+hr)^{\ell} \quad \textit{and} \quad |A \oplus C| \leq n^2/2^{\ell}.$$

PROOF. Let $\mathscr{L}$ be the linear space of $(0,1)$ matrices over $GF(2)$ generated by $A_1, \ldots, A_h$. Then $\mathrm{rk}(B) \leq hr$ for every $B \in \mathscr{L}$. Take a matrix $B = (b_{ij})$ in $\mathscr{L}$ at random. That is, $B = \bigoplus_{i=1}^{h} \lambda_i A_i$, where $\Pr[\lambda_i = 0] = \Pr[\lambda_i = 1] = 1/2$ for all coefficients λ_i uniformly and independently. Let $A = (a_{ij})$. Each time when $a_{ij} = 1$, the (i,j)-th entry of at least one of the matrices $A_1, \ldots, A_m$ is 1, and the corresponding scalar product $b_{ij} = \bigoplus_{i=1}^{h} \lambda_i \cdot A_i[i,j]$ equals 0 with probability $1/2$. That is, $\Pr[b_{ij} = 0 | a_{ij} = 1] = 1/2$.

Hence, if we let $C = (c_{ij})$ to be an OR of ℓ independent copies of B, then $\Pr[c_{ij} = 0] = 1$ if $a_{ij} = 0$, and $\Pr[c_{ij} = 0] \leq 2^{-\ell}$ if $a_{ij} = 1$. That is, the expected number of positions, where C deviates from A, does not exceed $n^2 \cdot 2^{-\ell}$.

It therefore exists a matrix C of the form $C = \bigvee_{k=1}^{\ell} B_k$ such that $|A \oplus C| \leq n^2/2^{\ell}$ and $\mathrm{rk}(B_i) \leq hr$ for each i. Using the rule $x \vee y = (x \oplus 1) \wedge (y \oplus 1) \oplus 1$, this OR can be written as an all-1 matrix plus an AND of ℓ matrices, each of which has rank at most $1 + hr$. Since the AND of matrices is a component-wise product of their entries, and component-wise product is bilinear in the space of rows of matrices, this implies that $\mathrm{rk}(A \wedge B) \leq \mathrm{rk}(A) \cdot \mathrm{rk}(B)$. Since we have an AND of ℓ matrices each of rank at most $1 + hr$, this give the desired upper bound $\mathrm{rk}(C) \leq 1 + (1+hr)^{\ell}$ on the rank of C. □

Now, if v is an OR gate at the ith layer of our circuit, and if it has h inputs then Lemma 11.17, applied with $r = (s+1)^{O(\ell^{i-1})}$, yields the desired approximator for v satisfying (11.6). The case of an AND gate reduces to that of OR gates by DeMorgan rules. □

Theorem 11.16 has several interesting consequence. Let us mention two of them.

1. Babai, Frankl and Simon (1986) introduced the communication complexity analogon PH^{cc} of the complexity class PH, and proved that PH^{cc} coincides with the class of $n \times n$ $(0,1)$ matrices M whose constant depth circuit complexity over the basis $\{\&, \vee\}$ does not exceed $\leq \exp\left((\ln \ln n)^{O(1)}\right)$. Theorem 11.16 immediately implies that, if

$$\mathscr{R}_M(r) \geq \frac{n^2}{\exp(\ln r)^{o(1)}} \quad \text{for} \quad r \geq \exp((\ln \ln n)^{\omega(1)}),$$

then $M \notin PH^{cc}$. That is, the class PH^{cc} does not contain highly rigid matrices.

2. Razborov (1988) used probabilistic arguments to show that unbounded fanin circuits over $\{\&, \oplus, 1\}$ of small depth can efficiently compute some combinatorially "complicated" matrices, sharing many extremal properties of random matrices. Together with Theorem 11.16 this implies that also matrices of low rigidity may share many properties of random matrices.

Exercises

Ex. 11.1. Let $X = \{x_1, \ldots, x_n\}$ be a set of boolean variables. Consider the following random restriction $\varrho : X \to \{0, 1, *\}$: for each $i = 1, \ldots, n$ set $\varrho(x_i) = *$ with probability $p = 1/\sqrt{n}$, and set x_i to 0 or 1 with equal probability $(1-p)/2$. Assume that n is large enough.

a. Let F be an OR of literals, and $c > 0$ a constant. Show that $F\restriction_\varrho$ will depend on more than c variables with probability at most $n^{-c/3}$.

Hint: Consider two cases depending on whether: (i) the clause is "large", that is, contains more than $m := c \log_2 n$ literals, or (ii) is "small", that is, contains at most m literals. Show that in the first case $F\restriction_\varrho$ will be non-constant with probability at most $((1+p)/2)^m$, whereas in the second case $F\restriction_\varrho$ will contain at least c variables with probability at most $\binom{m}{c}p^c$. Show that both these bounds are at most $n^{-c/3}$ if n is large enough.

b. Weaker version of the Switching Lemma. Prove that for every integer constants $c, k \geq 1$ there is a constant $b = b(c,k)$ with the following property: If F is a c-CNF on n variables, then

$$\Pr[F\restriction_\varrho \text{ depends on} \geq b \text{ variables}] \leq n^{-k}.$$

Hint: Argue by induction on k. Use the previous exercise for the base case $b(1,k) = 3k$. For the induction step, take a maximal set of clauses in F whose sets of variables are pairwise disjoint, and let Y be the union of these variable sets. Hence, each clause of F has at least one variable in Y. Consider two cases depending on whether $|Y| \geq k2^c \log n$ or not. If $|Y| \geq k2^c \log n$, then use the disjointness of clauses determining Y to show that then $F\restriction_\varrho$ becomes constant with probability at least $1 - n^{-k}$. In the case when $|Y| \leq k2^c \log n$ show that, for every i, the probability that more than i variables in Y will remain unassigned is at most $n^{-i/3}$ (cf. the previous exercise). Take $i = 4k$ (why?), set these $4k$ free variables of Y to constants in all possible ways to obtain a $(c-1)$-CNF F', and apply induction hypothesis to F'.

Bibliographic Notes

The version of the Switching Lemma (Lemma 11.1) is due to Hastad (1986, 1989). Somewhat weaker versions of this lemma were earlier proved by Ajtai (1983), Furst, Saxe, and Sipser (1984), and Yao (1985). All these proofs used probabilistic arguments. The novel (non-probabilistic) proof given in Section 11.2 is due to Razborov (1995). The results of Section 11.3 are also due to Razborov (1987), but in their presentation we followed the exposition of Lovász, Shmoys and Tardos (1995). Theorem 11.15 is a special case of a more general result proved by Smolensky (1997). Theorem 11.16 is due to Razborov (1989b).

CHAPTER 12

Depth-2 Circuits With Arbitrary Gates

In this chapter we consider unbonded fanin circuits of depth 2. If we would only allow AND, OR and NOT gates, then each such circuit would be just a DNF or a CNF, and large (exponential) lower bounds here are easy to show: already the Parity function has then exponential complexity. But what if we allow arbitrary(!) boolean functions be used as gates?

Of course, then every single boolean function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit with just one gate—the function f itself. The problem, however, becomes non-trivial if instead of one function, we want to *simultaneously* compute n boolean functions $f_1, \ldots, f_n$ on the same set of n variables $x_1, \ldots, x_n$, that is, to compute an *n-operator* $f : \{0,1\}^n \to \{0,1\}^n$. Note that in this case the phenomenon which causes complexity of circuits is *information transfer* instead of *information processing* as in the case of single functions.

As before, a circuit computing a given n-operator can be imagined as a directed acyclic graph with n input nodes corresponding to the variables $x_1, \ldots, x_n$, n output nodes corresponding to the boolean functions $f_1, \ldots, f_n$ to be computed, and each non-input node computing an arbitrary boolean function of its inputs. The size of the circuit is then the total number of wires in it.

Note that also for operators we cannot expect larger than n^2 lower bounds: every operator can be computed using at most n^2 wires, even in depth 1. In this chapter we will concentrate on general circuits of depth 2—the first non-trivial case. Superlinear lower bounds of the form $\Omega(n \log n)$ for such circuits were proved using graph theoretic arguments—so-called "superconcentrators"—and algebraic arguments—the matrix rigidity. The advantage of these arguments is that they actually say more: they give us a structural information about how the circuits for a given operator look like. The disadvantage is purely numerical: these arguments cannot (provably!) lead to larger than $\Omega(n \log^2 n)$ lower bounds on the number of wires.

Larger lower bounds $\Omega(n^{3/2})$ were recently proved using a much simper information theoretic argument, and we present it below. The argument itself is reminiscent of Nechiporuk's argument for formulas: an operator requires many wires if the number of its sub-operators is large.

12.1. Lower bounds for depth-2 circuits

Using counting arguments, it can be shown (Exercise 12.1) that *most* operators $f : \{0,1\}^n \to \{0,1\}^n$ require about n^2 wires in any circuit. But where are these "hard" operators? The disadvantage of any counting or probabilistic argument is that it usually gives no hint on what the hard objects actually are. In particular, what is the complexity of often used operators like cyclic convolution (corresponding to product of polynomials) or matrix product? That is, what we need are lower bounds for *explicit* operators.

In this section we prove such lower bounds in the class of depth-2 circuits. We will assume that there are no direct wires from input to output nodes: this can be easily achieved by adding n new nodes of fanin 2 on the middle layer labeled by input variables. The increase of the size by an additive factor of n will not hurt us, because the bounds we are going to prove will be super-linear in n.

Let $\text{size}_2(f)$ denote the smallest number of wires in a depth-2 circuit with arbitrary gates computing the operator f. In this section we will first show that it is the entropy of an operator f that forces large number of wires in depth-2 circuits. We will then use this to show that the operator f corresponding to the product two $n \times n$ boolean matrices over $GF(2)$ has $\text{size}_2(f) = \Theta(n^3)$.

12.1.1. Entropy and the number of wires. An operator $f : \{0,1\}^n \to \{0,1\}^m$ maps binary strings of length n to binary strings of length m. Each such operator can be looked at as a sequence $f = (f_1, \ldots, f_m)$ of m (not necessarily distinct) boolean functions $f_i : \{0,1\}^n \to \{0,1\}$, each on the same set of n variables. The *range* of f is the set

$$\text{Range}(f) = \{f(a) \mid a \in \{0,1\}^n\} \subseteq \{0,1\}^m$$

of distinct values taken by f. Define the *entropy*, $\mathsf{E}(f)$, of an operator f as the logarithms to the basis 2 of the number of distinct values taken by f. That is,

$$\mathsf{E}(f) = \log_2 |\text{Range}(f)| .$$

REMARK 12.1. It is clear that, for any operator $f = (f_1, \ldots, f_m) : \{0,1\}^n \to \{0,1\}^m$, we have that $\mathsf{E}(f) \leq \min\{n, m\}$, just because $|\text{Range}(f)| \leq \min\{2^n, 2^m\}$. For our purposes, however, more important will be the following three properties of entropy:

a. $\mathsf{E}(f) \leq |\{f_1, \ldots, f_m\}|$. That is, $\mathsf{E}(f)$ cannot exceed the number of *distinct* boolean functions in f. This holds because only different functions can produce different values.
b. $\mathsf{E}(f) \geq r$ if we have r distinct single variables among the functions $f_1, \ldots, f_m$, just because then f must take at least 2^r distinct values.
c. $\mathsf{E}(f) \leq \mathsf{E}(g)$ if every function f_i of f can be computed as some boolean function applied to the functions of operator g. Indeed, in this case $g(a) = g(b)$ implies $f(a) = f(b)$. Hence, f cannot take more distinct values than g.

The next important concept is that of a "suboperator". Given a set $I \subseteq [n]$ of inputs and a set $J \subseteq [m]$ of outputs, define the *suboperator* $f_{I,J}$ of f as an operator

$$f_{I,J} = (f_j^i \mid i \in I,\ j \in J)$$

consisting of $|I| \cdot |J|$ (not necessarily distinct) boolean functions f_j^i with $i \in I$ and $j \in J$, where f_j^i is a subfunction of f_j obtained by setting the ith variable to 1 and all remaining variables in I to 0. Thus, $f_{I,J}$ maps binary strings of length $n - |I|$ ($|I|$ variables are fixed) to binary strings of length $|I| \cdot |J|$ (so many functions f_j^i we have).

Take now an arbitrary depth-2 circuit computing a given operator f. Let I be a subset of input nodes and J a subset of output nodes. Let also $\text{Wires}(I, J)$ denote the number of wires leaving I plus the number of wires entering J.

The following lemma is our main technical tool relating the number of wires to the entropy of the computed operator.

LEMMA 12.2. *For any subset I of inputs and any subset J of outputs, we have:*

$$\text{Wires}(I, J) \geq \mathsf{E}(f_{I,J}) .$$

PROOF. Let V be the set of all nodes on the middle layer from which there is a wire to a node in J. (Note that V only depends on the choice of J.) For $v \in V$, let g_v be a boolean function computed at this node, and let $I(v)$ be the set of inputs $i \in I$ from which there is a wire to v. For $i \in I$, let g_v^i be the subfunction of g_v obtained by setting $x_i = 1$ and $x_j = 0$ for all $j \in I - \{i\}$. Let also g_v^0 be obtained from g_v by setting *all* variables x_i with $i \in I$ to 0. Consider the operator $h = (g_v^i \mid v \in V, i \in I)$. A simple (but crucial) observation is:

> If $i \notin I(v)$, then the function g_v cannot depend on the ith input variable x_i (since then we have no wire from x_i to g_v), implying that $g_v^i = g_v^0$.

Hence, for each node $v \in V$, the function g_v constitutes at most $1 + |I(v)|$ distinct functions to the operator h: the function g_v^0 and at most $|I(v)|$ distinct functions g_v^i with $i \in I(v)$. Since we have only $|V|$ possible functions g_v, the total number of distinct boolean functions in h and, by Remark 12.1(1), the entropy $\mathsf{E}(h)$ of h, cannot not exceed $|V| + \sum_{v \in V} |I(v)|$.

On the other hand, we have that $\text{Wires}(I,J) \geq |V| + \sum_{v \in V} |I(v)|$. Indeed, $\sum_{v \in V} |I(v)|$ is the number of wires going from I to V and, since (by the definition of V) from every node in V there must be at least one wire to a node in J, $|V|$ is at most the total number of wires from V to J.

To finish the proof, observe that all output functions f_j with $j \in J$ must be computable from the set of functions g_v with $v \in V$: only these functions have an influence (have a wire) to the outputs in J. Hence, the suboperator $f_{I,J}$ must be computable from h, as well. Remark 12.1(3) implies that

$$\mathsf{E}(f_{I,J}) \leq \mathsf{E}(h) \leq |V| + \sum_{v \in V} |I(v)|,$$

as desired. □

REMARK 12.3. Note that our lower bound on the number of wires going from V to J is very "pessimistic": we lower bound this number just by the number $|V|$ of the starting nodes of these wires, as if these nodes had fanout 1. Here, apparently, is some space for an improvement.

THEOREM 12.4. *Let f be an operator. Then, for every partition $I_1, \ldots, I_p$ of inputs, and every partition $J_1, \ldots, J_p$ of outputs, we have that*

$$\text{size}_2(f) \geq \mathsf{E}(f_{I_1,J_1}) + \mathsf{E}(f_{I_2,J_2}) + \cdots + \mathsf{E}(f_{I_p,J_p}).$$

PROOF. No wire can leave two input nodes, and no wire can enter two output nodes. Hence, the result follows directly from Theorem 12.2. □

12.1.2. Application: Matrix product. Let $n = m^2$. The operator $f = \text{mult}_n(X,Y)$ of matrix product takes two m-by-m matrices X and Y as inputs, and produces their product $Z = X \cdot Y$. Since Z is just a sequence of m^2 scalar products in $2m$ variables (row of X times a column of Y), all these scalar products can be computed by depth-1 circuit using $2m \cdot m^2 = 2n^{3/2}$ wires.

THEOREM 12.5. *Any depth-2 circuit for $\text{mult}_n(X,Y)$ requires at least $n^{3/2}$ wires.*

PROOF. Observe that if we take I to be the ith row of X and J to be the ith row of Z, then the suboperator $f_{I,J}$ contains all $m^2 = n$ single variables of Y among its boolean functions. Indeed, if we set $x_{ij} = 1$ and all other entries of X to 0, then the product $E_{ij} \cdot Y$ of Y with the resulting $(0,1)$ matrix E_{ij} is just the jth row of Y (see

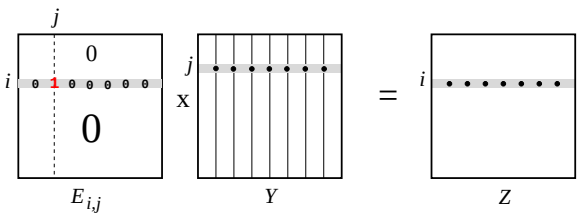


FIGURE 1. As j ranges from 1 to m, the values of $E_{i,j} \cdot Y$ range through all m single variables $y_{i,1}, \ldots, y_{i,m}$ of the ith row of Y.

Fig. 1). When doing this for $j = 1, \ldots, m$, we obtain all m^2 variables $Y = \{y_{ij}\}$ among the functions in $f_{I,J}$. By Remark 12.1(2), we then have that $\mathsf{E}(f_{I,J}) \geq n$.

Since we have $m = n^{1/2}$ rows, we have m sets $I_1, \ldots, I_m$ of inputs and m sets $J_1, \ldots, J_m$ of outputs. Since the I_i's as well as J_i's are disjoint, Theorem 12.4 implies that every depth-2 circuit computing $f(X,Y) = X \cdot Y$ must have at least $\sum_{i=1}^{m} \mathsf{E}(f_{I_i,J_i}) \geq mn = n^{3/2}$ wires. □

REMARK 12.6. Note that the entropy of the matrix product operator $f(X,Y) = X \cdot Y$ is large only for this special "row-wise" partition of inputs and outputs. In particular, $\mathsf{E}(f_{I_i,J_j}) \leq |J_j| = m = \sqrt{n}$ for $i \neq j$, because in this case the assignments of constants to the ith row of X does not affect the results computed at the jth row of Z, which are m scalar products of m columns of Y with the jth row of X.

There are, however, "more complicated" operators, whose entropy remains large under *any* partitions of inputs and outputs. Such is, for example, the operator of *cyclic convolution* $f = \mathrm{conv}_n(\boldsymbol{x}, \boldsymbol{y})$. This operator takes vectors $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ as inputs and outputs the vector $\boldsymbol{z} = (z_0, \ldots, z_{n-1})$, where

$$z_j = \sum_{k=i+j \pmod n} x_i y_k \pmod 2.$$

This operator can also represented as a matrix-vector product. Namely, associate with a vector of variables $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ the following $n \times n$ matrix of variables:

$$\mathrm{Circ}(\boldsymbol{x}) = \begin{pmatrix} x_0 & x_{n-1} & \cdots & x_2 & x_1 \\ x_1 & x_0 & \cdots & x_3 & x_2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n-2} & x_{n-3} & \cdots & x_0 & x_{n-1} \\ x_{n-1} & x_{n-2} & \cdots & x_1 & x_0 \end{pmatrix}.$$

Then $\mathrm{conv}_n(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{Circ}(\boldsymbol{x}) \cdot \boldsymbol{y}$ over $GF(2)$.

EXERCISE 12.7. Let $n = k \cdot l$. Show that, if we partite input vector $\boldsymbol{x}$ into $p = n/k = l$ consecutive intervals $I_1, \ldots, I_p$ of length $n/p = k$, then there exists a partition of the output vector $\boldsymbol{z}$ into $q = n/p = k$ disjoint sets $J_1, \ldots, J_q$ such that $\mathsf{E}(\mathrm{conv}_n(\boldsymbol{x}, \boldsymbol{y})_{I_i,J_j}) \geq n$ for *any* i and j. *Hint*: Consider residue classes modulo p.

12.2. Depth-2 circuits for linear operators

Every m-by-n $(0,1)$-matrix A defines a linear transformation $f_A(\boldsymbol{x}) = A\boldsymbol{x}$ over $GF(2)$, where $\boldsymbol{x} \in GF(2)^n$ is an input. If A is an $n \times n$ matrix, then we call f_A a *linear n-operator*. Each such operator is just a set of n linear forms over $GF(2)$.

We are interested in computing such forms by general circuits using as few wires as possible. We are particularly interested in circuits of depth 2, the first nontrivial case. Recall that such a circuit is a directed acyclic graph of depth 2 with n input nodes $x_1, \ldots, x_n$, n output nodes $y_1, \ldots, y_n$ and every non-input node v assigned a gate g_v computing an *arbitrary* boolean function of its inputs; there is no bound on the fanin or on the fanout of the nodes. A circuit is *linear* if all gates are linear functions over $GF(2)$, that is, parities or their negations.

PROPOSITION 12.8. *Some linear n-operators require at least $\Omega(n^2/\log n)$ wires in any linear circuits, and any such operator can be computed by a linear depth-2 circuits using $O(n^2/\log n)$ wires.*

PROOF. Simple counting argument shows that at most $(nL)^{O(L)}$ different linear n-operators can be computed by linear circuits with at most L wires. Since the total number of such operators is 2^{n^2}, the lower bound $L = \Omega(n^2/\log n)$ follows.

For the proof of the upper bound, we use a well-known fact, due to Erdös, that, if a graph G has n vertices and e edges, and k satisfies

$$n\binom{e/n}{k} \geq k \cdot \binom{n}{k},$$

then G contains a complete $k \times k$ subgraph. By removing these subgraphs one-by-one, we will decompose the graph into its *edge-disjoint* complete bipartite subgraphs so that the sum of sizes (=number of vertices) of these subgraphs does not exceed $O(n^2/\ln n)$. This decomposition then can be used to construct a linear depth-2 circuit for the matrix A with the same number of wires: the set of input nodes and output nodes, incident to fixed node on the middle layer, forms a bipartite graph. Since, due to the linearity, this circuit must also compute the whole operator f_A, we are done. □

Nothing similar, however, is known for general (non-linear) circuits computing linear operators. Here even the case of depth-2 circuits remains unclear. The largest lower bound for general depth-2 circuits computing a linear n-operator has the form $\Omega(n \log n)$. This bound was proved by Pudlák (1994) and is achieved by a triangular matrix A.

To show the difference between linear and general circuits for linear operators, we relax the problem and only require that a circuit correctly computes the operator on the standard basis $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$, where $\boldsymbol{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ with precisely one 1 in the jth position.

Namely, say that a circuit *represents* a boolean matrix $A = (a_{ij})$ if it correctly computes the linear operator $A\boldsymbol{x}$ over $GF(2)$ on all n unit vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$; on other input vectors $\boldsymbol{x}$ the circuit can output arbitrary values. Hence, if $f = (f_1, \ldots, f_n)$ is the operator computed by a circuit representing A, then the only requirement is that $f_i(\boldsymbol{e}_j) = a_{ij}$ must hold for all i and j.

It is clear that any circuit computing the whole operator $f_A x) = A\boldsymbol{x}$ must also represent the matrix. Moreover, in the class of *linear* circuits we also have an inverse: a linear circuit represents a matrix A if and only if it computes the entire linear transformation $A\boldsymbol{x}$. This holds, because the behavior of a linear circuit on all 2^n input vectors $\boldsymbol{x}$

is completely determined by its behavior on n unit vectors: just write each input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ as the sum $\boldsymbol{x} = x_1\boldsymbol{e}_1 \oplus \cdots \oplus x_n\boldsymbol{e}_n$ and use the linearity of gates.

We will now show that in the class of *general* circuits the situation is entirely different. Recall that, by Proposition 12.8 and the remark in the previous paragraph, in the class of linear circuits some matrices A require $\Omega(n^2/\log n)$ wires to represent them.

THEOREM 12.9. *Every $n \times n$ $(0,1)$ matrix A can be represented by a depth-2 circuit with $O(n \log n)$ wires.*

PROOF. We construct the desired depth-2 circuit representing $A = (a_{ij})$ as follows. Let m be the smallest even integer such that $\binom{m}{m/2} \geq n$; hence $m = O(\ln n)$. Take m middle nodes $V = \{v_1, \ldots, v_m\}$. To each input variable x_j assign its *own* subset $S_j \subseteq V$ of $|S_j| = m/2$ middle nodes; hence, $S_{j_1} \subseteq S_{j_2}$ iff $j_1 = j_2$. Join x_j with all nodes in S_j. Finally, connect each $v \in V$ with all output nodes. The total number of wires is then $n(m/2) + nm = O(n \ln n)$.

Now we assign gates to the nodes. If v is a node on the middle layer connected to inputs $x_{j_1}, \ldots, x_{j_k}$, then assign to v the gate $g_v = x_{j_1} \oplus \cdots \oplus x_{j_k}$. To the ith output node we assign the gate

$$\phi_i = a_{i1}h_1 \oplus a_{i2}h_2 \oplus \cdots \oplus a_{in}h_n\,, \text{ where } h_k = \prod_{v \in S_k} g_v\,.$$

Then

$$\begin{aligned} h_k(\boldsymbol{e}_j) = 1 &\text{ iff } g_v(\boldsymbol{e}_j) = 1 \text{ for all } v \in S_k \\ &\text{iff } x_j \text{ is connected to all nodes in } S_k \\ &\text{iff } S_k \subseteq S_j \\ &\text{iff } k = j. \end{aligned}$$

Hence, $h_j(\boldsymbol{e}_j) = 1$ and $h_k(\boldsymbol{e}_j) = 0$ for all $k \neq j$. Thus, if $f_i(\boldsymbol{x})$ is the function computed at the ith output gate then, for all $j = 1, \ldots, n$, we have that

$$f_i(\boldsymbol{e}_j) = \phi_i(\boldsymbol{e}_j) = a_{i1} \cdot 0 \oplus \cdots \oplus a_{ij} \cdot 1 \oplus \cdots \oplus a_{in} \cdot 0 = a_{ij}\,,$$

as desired. □

We now show that the upper bound $n \log n$ in Theorem 12.9 is almost optimal. We will use the following so-called "sunflower lemma" of Erdös and Rado (1960) which has found many applications in circuit complexity. [1]

A *sunflower* with k petals is a family $S_1, \ldots, S_k$ of k finite sets, each two of which share precisely the same set of common elements, called the *core* of the sunflower (see Fig. 2). That is, there is a set C (the core of the sunflower) such that

$$S_i \cap S_j = C \qquad \text{for all } 1 \leq i < j \leq k.$$

In other words, each element belongs either to *none*, or to *exactly one*, to *all* of the S_i. In particular, all sets $S_i - C$ are mutually disjoint.

LEMMA 12.10 (Sunflower Lemma). *Every family of more than $s!(k-1)^s$ sets, each of which has cardinality at most s, contains a sunflower with k petals.*

[1] Interestingly, without knowing this lemma, Razborov in his seminal result on monotone circuits rediscovered and essentially used this phenomenon—large structures have some regular patterns.

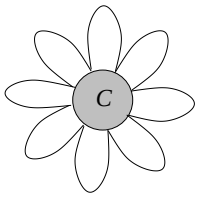


FIGURE 2. A sunflower with 8 petals

PROOF. We proceed by induction on s. For $s = 1$, we have more than $k-1$ points (disjoint 1-element sets), so any k of them form a sunflower with k petals (and an empty core). Now let $s \geq 2$, and $\mathscr{F}$ be a family of $|\mathscr{F}| > s!(k-1)^s$ sets, each of cardinality at most s. Take a maximal family $\mathscr{A} = \{A_1, \ldots, A_t\}$ of pairwise disjoint members of $\mathscr{F}$.

If $t \geq k$, these sets form a sunflower with $t \geq k$ petals (and empty core), and we are done.

Assume that $t \leq k-1$, and let $B = A_1 \cup \cdots \cup A_t$. Then $|B| \leq s(k-1)$. By the maximality of $\mathscr{A}$, the set B intersects every member of $\mathscr{F}$. By the pigeonhole principle, some point $x \in B$ must be contained in at least

$$\frac{|\mathscr{F}|}{|B|} > \frac{s!(k-1)^s}{s(k-1)} = (s-1)!(k-1)^{s-1}$$

members of $\mathscr{F}$. Let us delete x from these sets and consider the family

$$\mathscr{F}_x := \{S - \{x\} \mid S \in \mathscr{F},\, x \in S\}.$$

By the induction hypothesis, this family contains a sunflower with k petals. Adding x to the members of this sunflower, we get the desired sunflower in the original family $\mathscr{F}$. □

For a matrix A, let $\mathrm{dist}(A)$ denote the smallest Hamming distance between the columns of A.

THEOREM 12.11. *Every depth-2 circuit representing an n by n $(0,1)$-matrix requires at least*

$$\Omega\Big(\mathrm{dist}(A) \cdot \frac{\ln n}{\ln \ln n}\Big)$$

wires.

PROOF. Fix a minimal depth-2 circuit with arbitrary gates representing a given matrix A. Without loss of generality, we may assume that there are no direct wires from inputs to outputs: this can be easily achieved by adding at most n new wires. Let $x_1, \ldots, x_n$ be its input nodes, and $S_1, \ldots, S_n$ be sets of their neighbors on the middle layer. Let $f_1, \ldots, f_n$ be the functions computed at the output nodes. Since the circuit represents A, we must have that $f_i(\boldsymbol{e}_j) = a_{ij}$ for all $1 \leq i, j \leq n$.

Let L_1 be the number of wires leaving the input nodes, and L_2 the number of wires entering the output nodes. Hence, $L_1 = \sum_{i=1}^{n} |S_i|$, and $L_1 + L_2$ is the total number of wires. Set

$$m := c \ln n / \ln \ln n \tag{12.1}$$

for a sufficiently small absolute constant $c > 0$. If we have $L_1 > mn$ wires leaving the input nodes, then we are done. So, assume that $L_1 \leq mn$. Our goal is to show that then we must have $L_2 \geq m \cdot \text{dist}(A)$ wires entering the output nodes.

Our assumption $\sum_{i=1}^{n} |S_i| \leq mn$ implies that at least $n/2$ of the sets S_i must be of size at most $s = 2m$. Hence, if the constant c in (12.1) is small enough then, by Sunflower Lemma, these sets must contain a sunflower with $k = 2m$ petals. Having such a sunflower with a core C, we can pair its members arbitrarily $(S_{p_1}, S_{q_1}), \ldots, (S_{p_m}, S_{q_m})$; hence, $S_{p_i} \cap S_{q_i} = C$ for all $i = 1, \ldots, m$. Important for us will only be that the symmetric differences

$$S_{p_i} \oplus S_{q_i} = (S_{p_i} - S_{q_i}) \cup (S_{p_i} - S_{q_i}) = (S_{p_i} \cup S_{q_i}) - C$$

of these pairs of sets are mutually disjoint. Hence, we have m mutually disjoint subsets $S_{p_i} \oplus S_{q_i}$ of nodes on the middle layer, and we only have to show that each of these sets has at least $d = \text{dist}(A)$ outgoing wires: then $L_2 \geq m \cdot \text{dist}(A)$.

Fix one of the pairs (S_p, S_q). Since the circuit represents the matrix A, the value $f(\boldsymbol{e}_j)$ of the computed operator $f = (f_1, \ldots, f_n)$ on the jth unit vector must be the jth column of A. Since the Hamming distance between the pth and the qth columns of A must be at least d, there must exist a set I of $|I| \geq \text{dist}(A)$ rows such that

$$f_i(\boldsymbol{e}_p) \neq f_i(\boldsymbol{e}_q) \text{ for all } i \in I. \tag{12.2}$$

CLAIM 12.12. Every output f_i with $i \in I$ must be adjacent to at least one node in $S_p \oplus S_q$.

PROOF. Let V be the set of all nodes on the middle layer. For a node $v \in V$, let $g_v(x_1, \ldots, x_n)$ be the boolean function computed at this node. Claim 12.12 is a direct consequence of the following two observations about the behavior of the gates g_v on unit vectors. Let $\mathbf{0}$ denote the all-0 vector.

OBSERVATION 12.13. For every input node $j \in \{1, \ldots, n\}$, we have that $g_v(\boldsymbol{e}_j) = g_v(\mathbf{0})$ iff the wire (j, v) is not present.

PROOF. ($\Leftarrow$): If the wire (j, v) *is not* present, then g_v cannot depend on jth input variable x_j, and this is the only variable set to 1 by $\boldsymbol{e}_j$.

($\Rightarrow$): Suppose that the wire (j, v) *is* present. To show that then $g_v(\boldsymbol{e}_j) \neq g_v(\mathbf{0})$, assume that $g_v(\boldsymbol{e}_j) = g_v(\mathbf{0})$. Then we can remove the wire (j, v) and replace g_v by a new boolean function g'_v obtained from g_v by fixing the jth variable x_j of g_v to 0. By our assumption $g_v(\boldsymbol{e}_j) = g_v(\mathbf{0})$, we have that $g'_v(\boldsymbol{e}_j) = g_v(\mathbf{0}) = g_v(\boldsymbol{e}_j)$, as $\boldsymbol{e}_j$ has only one 1 in position j and the jth variable x_j is already set to 0 in g'_v. For the remaining unit vectors $\boldsymbol{e}_k$ with $k \neq j$, we also have that $g'_v(\boldsymbol{e}_k) = g_v(\boldsymbol{e}_k)$, just because the jth position of $\boldsymbol{e}_k$ is 0. Hence, we have removed one wire (j, v), and the resulting circuit still represents A. This contradicts the minimality of our original circuit. □

OBSERVATION 12.14. For all $v \notin S_p \oplus S_q$, we have that $g_v(\boldsymbol{e}_p) = g_v(\boldsymbol{e}_q)$.

PROOF. If $v \notin S_p \cup S_q$, then neither the wire (p, v) nor the wire (q, v) is present. Observation 12.13 implies that then $g_v(\boldsymbol{e}_p) = g_v(\mathbf{0}) = g(\boldsymbol{e}_q)$.

If $v \in S_p \cap S_q$, then both wires (p, v) and (q, v) must be present. Observation 12.13 implies that then $g_v(\boldsymbol{e}_p) \neq g_v(\mathbf{0})$ as well as $g_v(\boldsymbol{e}_q) \neq g_v(\mathbf{0})$. Hence, in this case we also have that $g_v(\boldsymbol{e}_p) = g_v(\boldsymbol{e}_q)$, just because g_v can take only two values. □

To finish the proof of Claim 12.12, take the boolean function f_i computed at the ith output gate with $i \in I$. The value of f_i only depends on the values of gates g_v

computed at the nodes on the middle layer. Hence, if there would be no wire from a node in $S_p \oplus S_q$ to the ith output f_i, then Observation 12.14 would imply that all gates on the middle layer, connected to f_i, would output the *same* values on input vectors $\boldsymbol{e}_p$ and $\boldsymbol{e}_q$. But this would imply $f_i(\boldsymbol{e}_p) = f_i(\boldsymbol{e}_q)$, a contradiction with (12.2).

This completes the proof of Claim 12.12. □

By Claim 12.12, for each of m pairs (S_{p_i}, S_{q_i}) of subsets of nodes on the middle layer, there must be at least $|I| \geq \text{dist}(A)$ wires going from the vertices in $S_{p_i} \oplus S_{q_i}$ to the output layer. Since the sets $S_{p_i} \oplus S_{q_i}$, $i = 1, \ldots, m$, are mutually disjoint, the total number of wires going from the middle layer to the output layer must be at least $m \cdot \text{dist}(A)$, as desired.

This completes the proof of Theorem 12.11. □

There are explicit $n \times n$ $(0,1)$ matrices H_n (so-called Sylvester–Hadamard matrices) such that $\text{dist}(H_n) \geq n/2$ but, still, the entire linear transformation $\boldsymbol{y} = H_n \boldsymbol{x}$ can be computed by a *linear* depth-2 circuit with $n \log_2 n$ wires (Exercise 12.3). Thus, the lower bound in Theorem 12.11 is almost tight. But only "almost."

RESEARCH PROBLEM 12.15. *Can the factor* $1/\ln\ln n$ *in Theorem 12.11 be removed?*

12.3. Relation to circuits of logarithmic depth

A depth-2 circuit of *width* w has n boolean variables $x_1, \ldots, x_n$ as input nodes, w arbitrary boolean functions $h_1, \ldots, h_w$ as gates on the middle layer, and arbitrary boolean functions $g_1, \ldots, g_n$ as gates on the output layer. Direct input-output wires, connecting input variables with output gates, are now allowed! Such a circuit computes an operator $f = (f_1, \ldots, f_n) : GF(2)^n \to GF(2)^n$ if, for every $i = 1, \ldots, n$,

$$f_i(\boldsymbol{x}) = g_i(\boldsymbol{x}, h_1(\boldsymbol{x}), \ldots, h_w(\boldsymbol{x})).$$

The *degree* of such a circuit is the maximum, over all output gates g_i, of the number of wires going directly from input variables $x_1, \ldots, x_n$ to the gate g_i. That is, we ignore the wires incident with the gates on the middle layer. Let $\deg_w(f)$ denote the smallest degree of a depth-2 circuit of width w computing f.

It is clear that $\deg_n(f) = 0$: just put the functions $f_1, \ldots, f_n$ on the middle layer. Hence, this parameter is only nontrivial for $w < n$. Especially interesting is the case when $w = O(n/\ln\ln n)$:

LEMMA 12.16. *If* $\deg_w(f) = n^{\Omega(1)}$ *for* $w = O(n/\ln\ln n)$*, then* f *cannot be computed by a circuit of depth* $O(\ln n)$ *using* $O(n)$ *fanin-2 gates.*

PROOF. Suppose that $f = (f_1, \ldots, f_n)$ can be computed by a circuit Φ of depth $O(\ln n)$ using $O(n)$ fanin-2 gates. By Lemma 10.2, for an arbitrary small constant $\varepsilon > 0$, any such circuit can be reduced to a circuit of depth at most $\varepsilon \log n$ by removing at most $w = O(n/\log\log n)$ edges. Put on the middle layer all the w boolean functions computed at these edges, and connect each middle node with all inputs as well as with all outputs. Since a subcircuit of Φ computing each f_i has depth at most $\varepsilon \log n$, each such subcircuit can depend on at most $2^{\varepsilon \log n} = n^{\varepsilon}$ input variables. Hence, the obtained depth-2 circuit has degree at most n^{ε}. Since this holds for arbitrary small constant $\varepsilon > 0$, we are done. □

The highest known lower bound for an explicit operator f, proved by Pudlák, Rödl and Sgall (1997) has the form $\deg_w(f) = \Omega((n/w)\ln(n/w))$, and is too weak to have a consequence for log-depth circuits.

A natural question therefore was to improve the lower bound on the degree at least for *linear* circuits, that is, for depth-2 circuits whose middle gates as well as output gates are linear boolean functions over some field $\mathbb{F}$. Such circuits compute linear operators $f_A(\boldsymbol{x}) = A\boldsymbol{x}$ for some matrix A over $\mathbb{F}$. By Lemma 12.16, this would give a super-linear lower bound for log-depth circuits over $\{\oplus, 1\}$. (Even over this basis no super-linear lower bound is known so far!)

This last question attracted attention of many researchers because of its relation to a purely algebraic characteristic of the underlying matrix A—its rigidity. Recall that the *rigidity* $\mathscr{R}_A(r)$ of a matrix A is the smallest number of entries of A that must be changed in order to reduce its rank over $\mathbb{F}$ until r. That is,

$$\mathscr{R}_A(r) = \min\{|B| : \mathrm{rk}(A-B) \le r\}.$$

It is not difficult to show that any linear depth-2 circuit Φ of width r computing $A\boldsymbol{x}$ must have degree at least $\mathscr{R}_A(r)/n$: If we set all direct input-output wires to 0, then the resulting degree-0 circuit will compute some linear transformation $A'\boldsymbol{x}$. The operator $\boldsymbol{y} = A'\boldsymbol{x}$ takes $2^{\mathrm{rk}(A')}$ different values. Hence, the operator $H : GF(2)^n \to GF(2)^r$ computed by w boolean functions on the middle layer of Φ must take at least so many different values, as well. This implies that the width r must be large enough to fulfill $2^r \ge 2^{\mathrm{rk}(A')}$, from which $\mathrm{rk}(A') \le r$ follows. On the other hand, A' differs from A in at most dn entries, where d is the degree of the original circuit Φ. Hence, $\mathscr{R}_A(r) \le dn$ from which $d \ge \mathscr{R}_A(r)/n$ follows. Thus an explicit $n \times n$ $(0,1)$ matrix A with

$$\mathscr{R}_A(r) \ge n^{1+\varepsilon} \quad \text{for} \quad r = O(n/\ln\ln m)$$

would give us a linear operator $f_A(\boldsymbol{x}) = A\boldsymbol{x}$ which cannot be computed by log-depth circuit over $\{\oplus, 1\}$ using a linear number of parity gates. To prove such a lower bound even over this basis remains an open problem!

Motivated by its connection to proving lower bounds for log-depth circuits, matrix rigidity (over different fields) was considered by many authors.

It is clear that $\mathscr{R}_A(r) \le (n-r)^2$ for any $n \times n$ matrix A: just take an arbitrary $r \times r$ submatrix A' of A and set to 0 all entries outside A. Valiant (1977) has proved that $n \times n$ matrices A with $\mathscr{R}_A(r) = (n-r)^2$ exist if the underlying field is infinite. For finite fields the lower bound is only slightly worse.

LEMMA 12.17. *There exist $n \times n$ matrices A over $GF(2)$ such that*

$$\mathscr{R}_A(r) \ge \frac{(n-r)^2 - 2n - \log_2 n}{\log_2(2n^2)} \quad \textit{for all} \quad r < n - \sqrt{2n + \log_2 n}. \tag{12.3}$$

PROOF. Direct counting. Recall that the rigidity $\mathscr{R}_A(r)$ of A over $GF(2)$ is the smallest number $|B|$ of nonzero entries in a $(0,1)$ matrix B such that $\mathrm{rk}(A \oplus B) \le r$. For $|B| = s$ there are at most $\binom{n^2}{s} \le n^{2s}$ possibilities to chose s nonzero entries of B, and at most $\binom{n}{r}^2 \le 2^{2n}$ possibilities to chose a nonsingular $r \times r$ minor of $A \oplus B$. Assuming that s is strictly smaller than the lower bound on $\mathscr{R}_A(r)$, given in (12.3), it can be easily verified that the number of possible matrices A is upper bounded by $2^{n^2}/n$, which is smaller than the total number 2^{n^2} of such matrices. □

The problem, however, is to exhibit an explicit matrix A of large rigidity. The problem is particularly difficult if we require A be a $(0,1)$ matrix or at least a matrix with relatively few different entries.

Explicit $n \times n$ ± 1 matrices A of rigidity $\mathscr{R}_A(r) = \Omega(n^2/r)$ over the reals is easy to present.

Let $n = 2^m$. The $n \times n$ Sylvester ± 1-matrix $S_n = (s_{ij})$ by labeling the rows and columns by m-bit vectors $x, y \in GF(2)^m$ and letting $s_{ij} = (-1)^{\langle x,y\rangle}$. Hence,

$$S_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}, \qquad S_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix} \text{ and } S_{2n} = \begin{bmatrix} S_n & S_n \\ S_n & \overline{S}_n \end{bmatrix},$$

where $\overline{S}_n$ is the matrix obtained from S_n by flipping all $+1$'s to -1's and all -1's to $+1$'s. The rigidity of these matrices over the reals is $n^2/4r$.

THEOREM 12.18. *If $r \le n/2$ is a power of 2 then $\mathscr{R}_{S_n}(r) \ge n^2/4r$.*

PROOF. Divide S_n uniformly into $(n/2r)^2$ submatrices of size $2r \times 2r$. One can easily verify that these submatrices each have full rank over the reals. So we need to change at least r elements of each submatrix to reduce each of their ranks to r, a necessary condition to reducing the rank of S_n to r. The total number of changes is then at least $r \cdot (n/2r)^2 = n^2/4r$. □

This proof works for any matrix whose submatrices have full rank. Consider the $n \times n$ matrix $B = (b_{ij})$ where $b_{ij} = 1$ if $i \equiv j \mod 2r$, and $b_{ij} = 0$ otherwise. By the same proof $\mathscr{R}_B(r) \ge n^2/4r$ even though the rank of B is only $2r$.

The same argument also yields a lower bound $n^2/2r$ on the rigidity of the following $(0,1)$ matrix D_n over $GF(2)$. The matrix D_n is defined for all n of the form $n = 2^m$ by the following recursion:

$$D_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \qquad D_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \text{ and } D_{2n} = \begin{bmatrix} D_n & D_n \\ D_n & \mathbf{0} \end{bmatrix}.$$

These bounds on rigidity are, however, still too weak to have consequences for log-depth circuits over $\{\oplus, 1\}$.

Exercises

EX. 12.1. Consider circuits of arbitrary depth with all boolean functions allowed as gates. Prove that operators $f : \{0,1\}^n \to \{0,1\}^n$ requiring $\Omega(n^2)$ gates in such circuits exist.

Hint: Let $\mu(L)$ be the number of different n-operators $f : \{0,1\}^n \to \{0,1\}^n$ computable by boolean circuits with at most L wires. Since we have 2^{n2^n} *different* n-operators, each of which requires a *different* circuit, it is enough to show that L must be large in order to fulfill the inequality $\mu(L) \ge 2^{n2^n}$. Hence, what we need is a good enough upper bound on $\mu(L)$ in terms of n and L.

The first instinct—just count, as Shannon did—will not work (directly). If the ith gate has d_i inputs, then there is a huge number $2^{2^{d_i}}$ of possibilities to assign a boolean function φ_i to this gate. If d_i is larger than $n + \log_2 n$, then the number of these possibilities alone exceeds the total number $(2^{2^n})^n = 2^{n2^n}$ of all n-operators! Shannon hasn't faced this situation, since $d_i \le 2$ in his model.

This unpleasant situation can, however, be avoided by an amazingly simple idea: just turn the power of the circuit against itself in order to ensure that $d_i \le n$ must hold in any optimal circuit.

Show that we can assume that we have $m \le n^2$ gates, and that no gate in circuit has fanin larger than n. Use this to show that, if $d_1, \ldots, d_m$ is a sequence of fanins of the gates in an optimal circuit with $\le L$ wires, then

$$\log_2 \mu(L) \le \sum_{i=1}^{m} 2^{d_i} + O(n^2 \log n).$$

Show that at most $n/2$ gates can have fanin d_i larger than $2L/n$, and use this to give an estimate

$$\sum_{i=1}^{m} 2^{d_i} \le O(n^2 4^{L/n}) + 2^{n-1}\,.$$

EX. 12.2. Recall that the rank $\mathrm{rk}(A)$ of an $n \times n$ matrix A over some field is the smallest number r such that A can be written as a product $A = B \cdot C$ of an $n \times r$ matrix B and an $r \times n$ matrix C. We now introduce a "weighted" version of rank and show that it coincides with the number of wires in linear depth-2 circuits computing $A\boldsymbol{x}$. For a $(0,1)$ matrix A, let $|A|$ be the number of 1's in A. Define

$$\mathrm{Rk}(A) = \min\{|B| + |C| : A = B \cdot C\}\,.$$

That is, now we are interested not in the dimension of the matrices B and C but rather in the total number of 1's in them.
Prove that $\mathrm{Rk}(A)$ is precisely the smallest number of wires in a linear depth-2 circuit computing $f_A(\boldsymbol{x}) = A\boldsymbol{x}$.

Hint: Take as B and C the adjacency matrices of the bipartite graphs formed by the first and the second level of wires.

EX. 12.3. Let $n = 2^r$ and consider a bipartite $n \times n$ $(0,1)$ matrix H_n whose rows and columns are indexed by vectors in $GF(2)^r$, and $H_n[x,y] = 1$ iff $\langle x, y\rangle = 1$, where $\langle x, y\rangle$ is the scalar product over $GF(2)$. Such matrices are known as $(0,1)$-*Sylvester matrices*.

a. Show that H_n can be defined inductively by

$$H_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \qquad H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \text{ and } H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & \overline{H}_n \end{bmatrix},$$

where $\overline{H}_n$ is the matrix obtained from H_n by flipping all its entries.

b. Show that $\boldsymbol{y} = H_n\boldsymbol{x}$ can be computed by a linear depth-2 circuit using $O(n \log n)$ wires. *Hint*: Exercise 12.2.

c. Show that $\mathrm{dist}(H_n) \ge n/2$.

EX. 12.4. **Research problem.** For an $n \times n$ $(0,1)$ matrix A, let $\mathrm{size}_2(A)$ denote the minimum number of wires in a depth-2 circuit with arbitrary boolean functions as gates computing the linear transformation $\boldsymbol{y} = A\boldsymbol{x}$ for all vectors $\boldsymbol{x} \in GF(2)^n$.

Does matrices A with $\mathrm{size}_2(A) = \Omega(n^2/\log n)$ exist?

EX. 12.5. Let A be an $n \times n$ $(0,1)$ matrix, and $\alpha = \alpha(n) \ge 2$. Prove that then

$$\mathrm{size}_2(A\boldsymbol{x}) \ge \frac{1}{\alpha}\sqrt{n \cdot \mathscr{R}_A\left(\frac{n}{\alpha}\right) - \frac{n^3}{\alpha}}\,,$$

where $\mathscr{R}_A(r) = \min\left\{|B| : \mathrm{rk}(A \oplus B) \le r\right\}$ is the rigidity of A over $GF(2)$.

Hint: Take a depth-2 circuit computing $A\boldsymbol{x}$ and let L be the number of wires in it. Say that a node v is *large* if $d(v) > \alpha L/n$, and *small* otherwise. Show that the total number of large nodes in the circuit must be smaller than n/α. Set all large outputs y_i to constant 0, and remove all wires incident to such inputs. The resulting circuit computes a linear operator $A'\boldsymbol{x}$ for an $n' \times n$ submatrix $A' \le A$ of A obtained by setting to 0's all its entries in the rows corresponding to large outputs y_i of the original circuit. What is $|A - A'|$? Remove now all small nodes on the middle layer. The resulting circuit still computes some linear operator $B\boldsymbol{x}$. How is B obtained from A'? How large is $|A' - B|$? In a resulting circuit for $B\boldsymbol{x}$, every path from an input node to an output node must go through some large node on the middle layer. But there are at most n/α such

nodes. Show that therefore, $\mathrm{rk}(B) \le n/\alpha$. Now use the definition of rigidity to give the desired lower bound on the number L of wires in the original circuit for $A\boldsymbol{x}$.

Ex. 12.6. Use Lemma 12.17 and the previous exercise to show that some boolean $n \times n$ matrices A require

$$\mathrm{size}_2(A\boldsymbol{x}) = \Omega\left(\left(\frac{n}{\log n}\right)^{3/2}\right).$$

Ex. 12.7. **Research Problem.** Prove or disprove: If a linear operator $f_A(\boldsymbol{x}) = A\boldsymbol{x}$ can be computed by a depth-2 circuit of degree d and width w, then f_A can also be computed by a *linear* depth-2 circuit of degree $O(d)$ and width $O(w)$.

Comment: Some partial results towards the positive answer are given in [**87**].

Ex. 12.8. Show that the only "sorrow" with the previous problem is the possible non-linearity of the gates on the last (output) layer: If a depth-2 circuit Φ computes a linear operator $f_A(\boldsymbol{x}) = A\boldsymbol{x}$ and has linear gates on the output layer, then Φ can be transformed into an equivalent *linear* circuit of the same size and width.

Hint: Replace the operator $h : \{0,1\}^n \to \{0,1\}^w$, computed at the middle layer, by a linear operator $H'(\boldsymbol{x}) := \sum_{i=1}^n x_i H(\boldsymbol{e}_i) \mod 2$.

Ex. 12.9. A boolean function f if *symmetric* if there is a set L of natural numbers (called the *type* of f) such that f accepts a binary vector $\boldsymbol{x}$ iff the number of 1's in $\boldsymbol{x}$ belongs to L. A *symmetric* depth-2 circuit is a depth-2 circuit with parity gates on the middle layer, and symmetric boolean functions of the same type on the output layer. Let $\mathrm{sym}_L(A)$ denote the smallest number of nodes on the middle layer of a symmetric depth-2 circuit of type L representing a $(0,1)$ matrix $A = (a_{ij})$. Let also $\mathrm{sym}(A)$ be the minimum of $\mathrm{sym}_L(A)$ over all types $L \subseteq \{0,1,\ldots\}$.

(a) Show that $\mathrm{sym}_L(A) =$ smallest number r for which it is possible to assign each row/column i a subset $S_i \subseteq \{1,\ldots,r\}$ such that $a_{ij} = 1$ if and only if $|S_i \cap S_j| \in L$.
(b) Show that $\mathrm{sym}(A) = \Omega(n)$ for almost all $n \times n$ matrices A.

Ex. 12.10. Let $n = 2^m$, and consider an $n \times n$ Sylvester matrix H_n. Its rows and columns are labeled by vectors in $GF(2)^m$, and the entries of H_n are the scalar products of these vectors over $GF(2)$. A type $L \subseteq \{0,1,\ldots\}$ is a threshold-k type if $L = \{k, k+1, \ldots\}$. Prove that

$$\mathrm{sym}_L(H_n) = \Omega(\sqrt{n}) \quad \text{for any threshold type } L\,.$$

Hint: Let $r = \mathrm{sym}_L(H_n)$, and consider an assignment $i \mapsto S_i$ of subsets $S \subseteq \{1,\ldots,r\}$ to rows/columns of $H_n = (h_{ij})$ such that $h_{ij} = 1$ iff $|S_i \cap S_j| \ge k$. Take $E = \{(i,j) \mid h_{ij} = 1\}$ and consider the family $\mathscr{F} = \{F_1,\ldots,F_r\}$ with $F_k = \{(i,j) \mid k \in S_i \cap S_j\}$. Show that $r \ge \mathrm{thr}_{\mathscr{F}}(E)$, where $\mathrm{thr}_{\mathscr{F}}(E)$ is the threshold cover number of E dealt with in the Discriminator Lemma (Lemma 10.24). Then use Lindsey's Lemma (Lemma 10.25) to show that $\mathrm{Disc}_{\mathscr{F}}(E) = O(n^{-1/2})$.

Ex. 12.11. **Research Problem.** Exhibit an explicit $n \times n$ $(0,1)$ matrix A such that $\mathrm{sym}(A) \ge 2^{(\log\log n)^{\alpha}}$ for some $\alpha(n) \to \infty$.

Comment: By results of Yao (1990) and Beigel and Tarui (1994) (combined with the Magnification Lemma for graphs), this would yield a super-polynomial lower bound for ACC circuits, and thus, resolve an old problem in circuit complexity. Actually, as shown by Green et al. (1995), it would be enough to prove such a lower bound on sym_L for a special kind of types L consisting

of all natural numbers whose binary representations have bit 1 in the middle. Such types (called also *middle-bit* types) consist of disjoint intervals of consecutive numbers.

EX. 12.12. **Research Problem.** Say that $L \subseteq \{0,1,\ldots\}$ is an *interval type* if $L = \{a, a+1, \ldots, b\}$ for some nonnegative integers $a \leq b$. Let $\mathrm{sym}_{\mathrm{int}}(A)$ denote the minimum of $\mathrm{sym}_L(A)$ over all interval types L.
Exhibit an explicit $n \times n$ $(0,1)$ matrix A such that $\mathrm{sym}_{\mathrm{int}}(A)$ is larger than $2^{(\log\log n)^c}$ for any constant c.

Comment: This would be a major step towards resolving the previous problem.

EX. 12.13. Let F be a depth-2 circuit computing a linear operator $f_A(\boldsymbol{x}) = A\boldsymbol{x}$ where A is an $n \times n$ $(0,1)$ matrix. Assume that all output gates are linear boolean functions; the gates on the middle layer may be arbitrary boolean functions. Assume also that there are no direct wires from inputs to outputs.
Show that F can be transformed into a *linear* depth-2 circuit with the same number of wires computing f_A.

Hint: Let h be the operator computed by the gates on the middle layer. Write each vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ as the linear combination $\boldsymbol{x} = \sum_{i=1}^{n} x_i \boldsymbol{e}_i$ of unit vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$, and replace the operator h by a *linear* operator $h'(\boldsymbol{x}) := \sum_{i=1}^{n} x_i h(\boldsymbol{e}_i) \bmod 2$. Show that the obtained linear circuit computes f_A and that the number of wires has not increased.

Bibliographic Notes

The idea to count subfunctions was first used by Cherukhin (2005) to prove an $n^{3/2}$ lower bound for the operator of cyclic convolution. Lemma 12.2 and Theorem 12.5 are from [**83**]. The proof idea of Theorem 12.11 is essentially the same as that of a similar lower bound proved by Alon, Karchmer and Wigderson (1990) for *linear* depth-2 circuits. Lower bounds of the form $\mathscr{R}_A(r) = \Omega(n^2/r)$ for the matrix rigidity were proved by many authors using different arguments. The short proof above was given by Midrijanis (2005). In fact, similar arguments (all submatrices have large rank) were earlier used by Dmitri Grigoriev and Noam Nisan.

Part 4

Decision Trees and Branching Programs

CHAPTER 13

Decision Trees

A decision tree is an algorithm for computing a function of an unknown input. Each vertex of the tree is labeled by a variable and the branches from that node are labeled by the possible values of the variable. The leaves are labeled by the output of the function. The process starts at the root, knowing nothing, works down the tree, choosing to learn the values of some of the variables based on those already known and eventually reaches a decision. The decision tree complexity of a function is the minimum depth of a decision tree that computes that function.

To be a bit more precise, let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. A deterministic *decision tree* for f is a binary tree whose internal nodes have labels from $x_1, \ldots, x_n$ and whose leaves have labels from $\{0,1\}$. If a node has label x_i then the test performed at that node is to examine the ith bit of the input. If the result is 0, one descends into the left subtree, whereas if the result is 1, one descends into the right subtree. The label of the leaf so reached is the value of the function (on that particular input).

13.1. P = NP ∩ co-NP for the tree depth

The *depth* of a decision tree is the number of edges in a longest path from the root to a leaf, or equivalently, the maximum number of bits tested on such a path.

Let Depth(f) denote the minimum depth of a decision tree computing f.

Given an input $a = (a_1, \ldots, a_n)$ from $\{0,1\}^n$, we would like to know whether $f(a) = 1$ or $f(a) = 0$. How many bits of a must we see in order to answer this question? It is clear that seeing Depth(f) bits is always enough: just look at those bits of a which are tested along the (unique!) path from the root to a leaf.

In a deterministic decision tree all the tests are made in a prescribed order independent of individual inputs. Can we do better if we relax this and allow for each input a to choose its own smallest set of bits to be tested? This question leads to a notion of "nondeterministic" decision tree.

A *nondeterministic decision tree* for a boolean function $f(x_1, \ldots, x_n)$ is a (not necessarily binary) tree each whose edge is labeled by a literal (a variable or a negated variable). One literal can label several edges leaving one and the same node. Such a tree T computes f in a nondeterministic manner: $T(a) = 1$ iff there exists a path from a root to a leaf such that all literals along this path are consistent with the input a, that is, are evaluated to 1 by this input. Let $D_1(f)$ denote the smallest depth of a nondeterministic tree computing f, and define the dual measure by $D_0(f) = D_1(\neg f)$. It is not difficult to verify that

$$D_1(f) = \min\{k \mid f \text{ can be written as a } k\text{-DNF}\}$$

and

$$D_0(f) = \min\{k \mid f \text{ can be written as a } k\text{-CNF}\}.$$

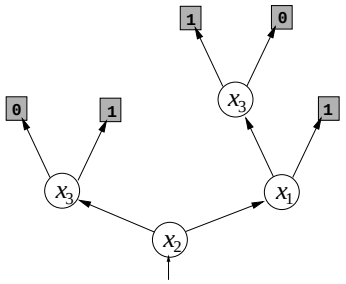


FIGURE 1. A decision tree of depth 3; on input $(0,1,0)$ it outputs 1.

It is clear that $\max\{D_0(f), D_1(f)\} \le \mathrm{Depth}(f)$, that is, for every input a, seeing its $\mathrm{Depth}(f)$ bits is enough to determine the value $f(a)$, be it 0 or 1. Is this upper bound optimal? The following example shows that this may be not the case: there are boolean functions f for which

$$\max\{D_0(f), D_1(f)\} \le \sqrt{\mathrm{Depth}(f)}.$$

Such is, for example, the monotone boolean function $f(X)$ on $n = m^2$ boolean variables defined by:

$$f = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} x_{ij}.$$

For this function we have $D_0(f) = D_1(f) = m$ but $\mathrm{Depth}(f) = m^2$ (see Exercise 13.1), implying that $\mathrm{Depth}(f) = D_0(f) \cdot D_1(f)$.

It turns out that the example given above is, in fact, the worst case: if a boolean function f can be written as an s-CNF as well as a t-DNF then $\mathrm{Depth}(f) \le s \cdot t$.

THEOREM 13.1. *For every boolean function f,*

$$\mathrm{Depth}(f) \le D_0(f) \cdot D_1(f).$$

PROOF. Induction on the number of variables n. If $n = 1$ then the inequality is trivial.

Let (say) $f(0,\dots,0) = 0$; then some set Y of $k \le D_0(f)$ variables can be chosen such that by fixing their value to 0, the function is 0 independently of the other variables. We can assume w.l.o.g. that the set

$$Y = \{x_1, \dots, x_k\}$$

of the first k variables has this property.

Take a complete deterministic decision tree T_0 of depth k on these k variables. Each of its leaves corresponds to a unique input $a = (a_1, \dots, a_k) \in \{0,1\}^k$ reaching this leaf. Replace such a leaf by a minimal depth deterministic decision tree T_a for the sub-function

$$f_a := f(a_1, \dots, a_k, x_{k+1}, \dots, x_n).$$

Obviously, $D_0(f_a) \le D_0(f)$ and $D_1(f_a) \le D_1(f)$. We claim that the latter inequality can be strengthened:

$$D_1(f_a) \le D_1(f) - 1 \tag{13.1}$$

To prove this, take an arbitrary input $(a_{k+1}, \dots, a_n)$ of f_a which is accepted by f_a. Together with the bits $(a_1, \dots, a_k)$, this gives an input of the whole function f with

$f(a_1,\ldots,a_n)=1$. According to the definition of the quantity $D_1(f)$, there must be a set $Z=\{x_{i_1},\ldots,x_{i_m}\}$ of $m\le D_1(f)$ variables such that fixing them to the corresponding values $x_{i_1}=a_{i_1},\ldots,x_{i_m}=a_{i_m}$, the value of f becomes 1 independently of the other variables. A simple (but crucial) observation is that

$$Y\cap Z\neq\emptyset. \tag{13.2}$$

Indeed, if $Y\cap Z=\emptyset$ then the value of $f(0,\ldots,0,a_{k+1},\ldots,a_n)$ should be 0 because fixing the variables in Y to 0 forces f to be 0, but should be 1, because fixing the variables in Z to the corresponding values of a_i forces f to be 1, a contradiction.

By (13.2), only $|Z-Y|\le m-1$ of the bits of $(a_{k+1},\ldots,a_n)$ must be fixed to force the sub-function f_a to obtain the constant function 1. This completes the proof of (13.1).

Applying the induction hypothesis to each of the sub-functions f_a with $a\in\{0,1\}^k$, we obtain

$$\mathrm{Depth}(f_a)\le D_0(f_a)\cdot D_1(f_a)\le D_0(f)(D_1(f)-1).$$

Altogether,

$$\mathrm{Depth}(f)\le k+\max_a \mathrm{Depth}(f_a)\le D_0(f)+D_0(f)(D_1(f)-1)=D_0(f)D_1(f). \qquad \square$$

13.1.1. Block sensitivity. Let $f:\{0,1\}^n\to\{0,1\}$ be a boolean function, and $a\in\{0,1\}^n$. A *certificate* of a is a subset $S\subseteq\{1,\ldots,n\}$ such that $f(b)=f(a)$ for all vectors $b\in\{0,1\}^n$ such that $b_i=a_i$ for all $i\in S$. That is, the value $f(a)$ can be determined by looking at only bits of a in the set S. By $C(f,a)$ we denote the minimum size of a certificate for a. The *certificate complexity* of f is $C(f)=\max_a C(f,a)$.

It is not difficult to see that $C(f)=\max\{D_1(f),D_0(f)\}$. Theorem 13.1 gives the following relation between the decision tree depth of boolean functions and their certificate complexity:

$$C(f)\le\mathrm{Depth}(f)\le C(f)^2.$$

A similar relation also exists between certificate complexity and another important measure of boolean functions – their "block sensitivity."

For a vector $a\in\{0,1\}^n$ and a subset of indices $S\subseteq\{1,\ldots,n\}$, let $a[S]$ denote the vector a, with *all* bits in S flipped. That is, $a[S]$ differs from a exactly on the bits in S. For example, if $a=(0,1,1,0,1)$ and $S=\{1,3,4\}$, then $a[S]=(1,1,0,1,1)$. We say that f is *sensitive* to S on a if

$$f(a[S])\neq f(a).$$

The *block sensitivity* of f on a, denoted $B(f,a)$, is the largest number t for which there exist t disjoint sets $S_1,\ldots,S_t$ such that f is sensitive on a to each of these sets, i.e., $f(a[S_i])\neq f(a)$ for all $i=1,\ldots,t$. The *block sensitivity* of a boolean function f is $B(f)=\max_a B(f,a)$.

THEOREM 13.2. *For every boolean function f,*

$$B(f)\le C(f)\le B(f)^2.$$

PROOF. The upper bound $B(f)\le C(f)$ follows from the fact that for any input a, any certificate of a must include at least one variable from each set which f is sensitive to on this input a.

The lower bound: $B(f)\ge\sqrt{C(f)}$. Take an input $a\in\{0,1\}^n$ achieving certificate complexity, i.e., every certificate for a has size at least $C(f)$. Let S_1 be a minimal set of indices for which $f(a[S_1])\neq f(a)$, let S_2 be another minimal set disjoint from S_1,

and such that $f(a[S_2]) \neq f(a)$, etc. Continue until no such set exists. We have, say, t disjoint sets $S_1, \ldots, S_t$ to each of which the function f is sensitive on a.

The union $I = S_1 \cup \cdots \cup S_t$ must be a certificate of a since otherwise we could pick yet another set S_{t+1} for which $f(a[S_{t+1}]) \neq f(a)$. Thus

$$\sum_{i=1}^{t} |S_i| = |I| \geq C(f).$$

If $t \geq \sqrt{C(f)}$ then we are done since $B(f) \geq t$. If not, then,

$$|S| \geq |I|/t \geq C(f)/t \geq \sqrt{C(f)}$$

for at least one set $S \in \{S_1, \ldots, S_t\}$. Since S is a minimal set for which $f(a[S]) \neq f(a)$, this means that $f(a[S - \{i\}]) = f(a)$ for every $i \in S$. Thus, on the vector $b := a[S]$, the function f is sensitive to each single coordinate $i \in S$; hence

$$B(f) \geq B(f, b) \geq |S| \geq \sqrt{C(f)}. \qquad \square$$

13.2. Depth lower bounds

To prove that some boolean function f requires decision trees of large depth, it is useful to imagine the situation as a game between Alice and Bob. This time, however, the players are not cooperative: Alice acts as an "adversary." Bob knows the function $f : \{0,1\}^n \to \{0,1\}$ but does not know the actual input vector $x \in \{0,1\}^n$. He can ask Alice what the ith bit of x is. Then what the jth bit is, and so on. He stops when he definitely knows the answer "$f(x) = 0$" or "$f(x) = 1$." Alice's goal is to inductively construct (depending on what bits Bob has already asked about) an input x on which Bob is forced to make many queries. That is, Alice tries to construct an "evasive" path forcing Bob to make his tree deep.

We demonstrate this on symmetric functions. Recall that a boolean function is *symmetric* if every permutation of its variables leaves its value unchanged. That is, a boolean function is symmetric if and only if its value depends only on *how many* of its variables (not on *which* of them) are 0 or 1.

A boolean function f in n variables is called *evasive* if it has maximal possible depth, that is, if $\mathrm{Depth}(f) = n$.

LEMMA 13.3. *Every non-constant symmetric boolean function is evasive.*

PROOF. Let $f : \{0,1\}^n \to \{0,1\}$ be a symmetric boolean function in question. Since f is not constant, there is a k with $1 \leq k \leq n$ such that if $k-1$ variables have value 1 then the function has value 0 but if k variables are 1 then the function's value is 1 (or the other way round).

Using this, we can propose the following strategy for Alice. She thinks of a 0-1 sequence of length n and Bob can ask the values of each bit. Alice answers 1 on the first $k-1$ questions and 0 on every following question. Thus, after $n-1$ questions, Bob cannot know whether the number of 1's is $k-1$ or k, that is, he cannot know the value of the function. $\square$

Every boolean function f in n variables splits the n-cube $\{0,1\}^n$ into two disjoint blocks $f^{-1}(0)$ and $f^{-1}(1)$. Since the number 2^n of vectors in the n-cube is even, the sizes of these blocks must be both even or both must be odd. It turns out that all boolean function with odd block size are evasive.

LEMMA 13.4. *If $|f^{-1}(0)|$ is odd then f is evasive.*

PROOF. Consider an arbitrary deterministic decision tree that computes the function f. Let v be an arbitrary node in this tree. If the depth of v is d, then exactly 2^{n-d} of the possible inputs lead to v. In particular, any node whose depth is at most $n-1$ is reached by an even number of possible inputs. On the other hand, each input reaches exactly one leaf. Thus, if $|f^{-1}(0)|$ is odd, there must be a leaf that is reached by a single input $\boldsymbol{x}$ with $f(\boldsymbol{x}) = 0$; this leaf has depth n. □

Symmetric functions are very special; the following class is significantly more general. Call a boolean function in n variables *weakly symmetric* if for all pairs x_i, x_j of variables, there is a permutation of the variables that takes x_i into x_j but does not change the value of the function. For example, the function

$$(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee \cdots \vee (x_{n-1} \wedge x_n) \vee (x_n \wedge x_1)$$

is weakly symmetric but not symmetric (check this!).

THEOREM 13.5. *Let n be a prime power. If $f : \{0,1\}^n \to \{0,1\}$ is weakly symmetric, and $f(\mathbf{0}) \neq f(\mathbf{1})$, then f is evasive.*

PROOF. Every permutation $\pi : [n] \to [n]$ on input coordinates induces a permutation $\widehat{\pi} : \{0,1\}^n \to \{0,1\}^n$ on the set of possible input vectors:

$$\widehat{\pi}(x_1, \ldots, x_n) = (x_{\pi(1)}, \ldots, x_{\pi(n)}) .$$

Let Γ be the set of all permutation π that leave the value of the function unchanged, that is,

$$\Gamma = \{\pi \mid f(\widehat{\pi}(x)) = f(x)\} .$$

It can be easily verified that Γ forms a group. Moreover, since the function f is weakly symmetric, this group is *transitive*, that is, for any pair of ground elements i and j, there is a permutation $\pi \in \Gamma$ such that $\pi(i) = j$.

We define the *orbit* of a vector $x \in \{0,1\}^n$ to be the set of images of x under permutations in Γ:

$$\text{orbit}(x) = \{\widehat{\pi}(x) \mid \pi \in \Gamma\} .$$

CLAIM 13.6. For any vector x except $\mathbf{0}$ or $\mathbf{1}$, the size $|\text{orbit}(x)|$ is divisible by n.

PROOF. Since $x \neq \mathbf{0}$ and $x \neq \mathbf{1}$, the orbit of x has more than one element. Let $|x|$ denote the number of 1's in x. Then

$$\sum_{y \in \text{orbit}(x)} |y| = \sum_{y \in \text{orbit}(x)} \sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \sum_{y \in \text{orbit}(x)} y_i .$$

Since Γ is transitive, for every i, there must be a permutation $\pi \in \Gamma$ such that $\pi(i) = 1$. Thus the last summand does not actually depend on i, implying that

$$\sum_{y \in \text{orbit}(x)} |y| = n \cdot \sum_{y \in \text{orbit}(x)} y_1 .$$

Since all vectors in the orbit have the same number of 1s, we have

$$\sum_{y \in \text{orbit}(x)} |y| = |\text{orbit}(x)| \cdot |x| .$$

Thus, $|\text{orbit}(x)| \cdot |x|$ is divisible by n. On the other hand, $0 < |x| < n$ implies that $|x|$ is not divisible by n. Since n is prime power, Euclid's theorem implies that $|\text{orbit}(x)|$ must be divisible by n. □

By Lemma 13.4, the function f is evasive if

$$S := \sum_{x \in f^{-1}(0)} (-1)^{|x|} \neq 0\,.$$

If $f(x) = 0$, then the orbit of x contributes

$$\sum_{y \in \text{orbit}(x)} (-1)^{|y|} = |\text{orbit}(x)| \cdot (-1)^{|x|}$$

to this sum, since all vectors in orbit(x) have the same number of 1s. By Claim 13.6, this is a multiple of n, except for the cases $x = \mathbf{0}$ and $x = \mathbf{1}$. Since exactly one of the vectors $\mathbf{0}$ and $\mathbf{1}$ is in $f^{-1}(0)$, the sum S is either one more or one less than a multiple of n. In either case, $S \neq 0$, so f must be evasive. □

13.3. Decision trees for graph properties

We will now consider decision problems for graphs, like "is a given graph connected?" or "does a given graph has a Hamiltonian cycle?" We will only consider properties of graphs that are "label independent:"

$(*)$ If a graph has this property then every graph isomorphic to it has that property.

In order to represent such a property as a boolean function, we have first fix a labeling of vertices, say, $1, 2, \ldots, v$. Then we introduce a boolean variable x_{ij} for each pair $i \neq j$ of vertices with value 1 if i and j are adjacent and 0 if they are not. Thus, each vector $x \in \{0,1\}^n$ with $n = \binom{v}{2}$ gives us a (labeled) graph on v vertices. In particular, the connectivity property for graphs corresponds then to a boolean function f such that $f(x) = 1$ iff the graph encoded by x is connected.

Due to $(*)$, the boolean function f corresponding to a graph property is weakly symmetric: for every two pairs of vertices, say, $\{i, j\}$ and $\{k, l\}$, there is a permutation of the vertices taking i into k and j into l. This permutation also induces a permutation on the set of point pairs that takes the first pair into the second one and does not change the value of f. In other words, a graph property f must be invariant under relabelings of the vertices, or equivalently, under any permutation of the edges that is induced by a permutation of the vertices.

A graph property is *monotone* if either (1) every subgraph of a graph with the property also has the property, or (2) no subgraph of a graph without the property has the property. For example, both properties "to have a k-clique" and "to be disconnected" are monotone: the first is preserved by adding edges, the second is preserved by removing edges. A graph property is *trivial* if either every graph has it or no one has it.

It turns out that *every* monotone non-trivial property of graphs on v vertices requires decision trees of depth $\Omega(v^2)$. This is almost maximal since depth $\binom{v}{2}$ is always enough: just take a complete tree whose paths correspond to all $2^{\binom{v}{2}}$ possible graphs.

Theorem 13.5 is not immediately useful for graph properties, since the number $n = \binom{v}{2}$ of possible edges is only a power of a prime when $v = 2$ or $v = 3$. Still, when properly applied, the theorem works at least in the case when the number of vertices v is a power of two. For this we will make use of the following simple principle, which we already used several times without explicitly mentioning it:

The Little Birdie Principle:
Extra information cannot increase the complexity of the problem.

Thus, in order to prove that it is hard to compute a function f on all inputs, it is enough to show that it is hard to compute f on some subset of inputs.

THEOREM 13.7. *Let $v = 2^m$, and let f be a nontrivial monotone graph property of v-vertex graphs. Then* $\mathrm{Depth}(f) \geq v^2/4$.

PROOF. Let $n = \binom{v}{2}$ and let $f : \{0,1\}^n \to \{0,1\}$ be a nontrivial monotone graph property. Let G_k be the graph consisting of $v/2^k$ disjoint copies of the clique K_{2^k} on 2^k vertices. In particular, G_0 is the empty graph on v vertices (no edges at all), and G_m is the complete graph K_v on v vertices. Since the property is non-trivial, we can assume w.l.o.g. that $f(G_0) = 1$ and $f(G_m) = 0$. By monotonicity, there is a unique index k such that $f(G_k) = 1$ but $f(G_{k+1}) = 0$.

Split the vertices into two equal sized parts, and suppose a little bird tells us that the induced subgraph on each of the two parts consists of $v/2^{k+1}$ disjoint copies of K_{2^k}. Now only $v^2/4$ of the $\binom{v}{2}$ bits actually matter, namely pairs i, j with i and j from different parts of the bipartition. Let $f' : \{0,1\}^{v^2/4} \to \{0,1\}$ denote the induced (bipartite graph) property on these bits; the remaining variables of f are fixed according to the little bird's information.

Since v is a power of two, $v^2/4$ is a power of a prime. Moreover, $f'(\mathbf{0}) \neq f'(\mathbf{1})$, since $f'(\mathbf{0}) = f(G_k) = 1$ and $f'(\mathbf{1}) = f(G_{k+1} + \text{ some edges}) = 0$. Finally, f' is invariant under a transitive automorphism group induced by the vertex permutations that leave the little birdie information fixed, that is, permute the vertices with the two parts of the bipartition. This implies that f' is weakly symmetric, and Theorem 13.5 yields $\mathrm{Depth}(f) \geq \mathrm{Depth}(f') = v^2/4$, as desired. □

13.4. $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ for the tree size

The *size* of a decision tree is the number of all its leaves. Let $\mathrm{Size}(f)$ denote the minimum size of a *deterministic* decision tree computing f. The minimum size of a *nondeterministic* decision tree for f is denoted by $\mathrm{dnf}(f)$. Note that $\mathrm{dnf}(f)$ is just the minimal number of monomials in a DNF of f. That is, $\mathrm{dnf}(f)$ is the minimal number t such that f can be written as an Or of t monomials.

We already know that $\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$ for decision trees if we consider their *depth* as complexity measure. In this section we will show that the situation changes drastically if we consider their size instead of the depth: in this case we have $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$. Namely, there are explicit boolean functions f such that *both* f and its negation $\neg f$ have nondeterministic decision trees of small size, whereas the size of any deterministic decision tree for f is super-polynomial.

Let f be a boolean function, and suppose we know that $\mathrm{dnf}(f)$ is small. Is then the decision tree also small? The following examples show that it may be *not* the case:

$$f = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} x_{ij}\,.$$

It can be shown that $\mathrm{Size}(f) \geq 2^{\mathrm{dnf}(f)}$ (Exercise 13.4). Well, this function has very small DNF (of size m) but the DNF of its negation

$$\neg f = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} \neg x_{ij}$$

is huge—it has m^m monomials. It is therefore natural to ask what happens if *both* the function f *and* its negation $\neg f$ have small DNFs? Put otherwise, does $\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$ for decision trees if we consider the *size* as their complexity measure? Below we answer this question negatively.

The sum $N(f) := \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$ will be called the *weight* of f. It is clear that $N(f) \leq \mathrm{Size}(f)$ (just because every decision tree represents *both* the function and its negation). It was long unknown if $\mathrm{Size}(f)$ is polynomial in $N(f)$, i.e. if $\mathrm{Size}(f) \leq N(f)^c$ for some absolute constant c.

It was however known that the decision tree size of any boolean function is *quasi-polynomial* in its weight. In the next section we will show that this upper bound is almost optimal: there are explicit functions f for which $\mathrm{Size}(f) = 2^{\Omega(\log^2 N)}$. Here and throughout, $\log x$ stands for $\log_2 x$.

We have seen (Theorem 13.1) that, if a boolean function f as well as its negation $\neg f$ can be written as a DNF, all whose monomial have length at most m, then f has a deterministic decision tree of depth at most m^2. For the *size* of trees we have the following analogon:

THEOREM 13.8 (Upper bound). *Let f be a boolean function in n variables and $N = \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$ be its weight. Then*

$$\mathrm{Size}(f) \leq \left(\tfrac{n}{\log^2 N}\right)^{O(\log^2 N)} \leq n^{O(\log^2 N)}.$$

PROOF. The idea is to apply the following simple "greedy" strategy: given DNFs for f and $\neg f$, let the decision tree *always* test the "most popular" literal first.

Assume, we have DNFs for both f and $\neg f$, and let N be the total number of monomials in these two DNFs. Since the disjunction of these two DNFs is a tautology (i.e., outputs 1 on all inputs), there must exist a monomial of length at most $\log_2 N$, just because monomial of length k accepts only 2^{n-k} of the inputs.

Select one of such monomials and denote its length by k. The selected monomial belongs to one of the two DNFs. By the cross-intersection property of monomials (see Exercise 13.3), every monomial in the other DNF contains at least one literal which is contradictory to at least one literal in the selected monomial. Hence, there is a literal in the selected monomial, which is contradictory to at least a $1/k$-portion of the monomials in the other DNF. Thus, if we evaluate this literal to 1, then all these monomials will get the value 0 and so will disappear from the DNF.

Test this variable first and apply this strategy recursively to both restrictions which arise. By the observation we just made, for each node v, at least one of its two successors is such that at least one of the two DNFs in it decreases by a factor of $1 - 1/k$. Let us call the corresponding outgoing edge(s) *decreasing*. Now, if v is a node (not a leaf) such that the path from the source to v contains s decreasing edges, at least one of the two initial DNFs was decreased at least $s/2$ times, and each time it was decreased by a factor of $1 - 1/k \geq 1 - 1/\log_2 N$. If s would be at least $2\log^2 N$ then at least one of the DNFs at v would have only

$$N\left(1 - \frac{1}{\log_2 N}\right)^{s/2} < N \cdot e^{-s/(2\log_2 N)} \leq N \cdot e^{-\log_2 N} = N^{1-\log_2 e} < 1$$

monomials, which is impossible (because v is not a leaf). Thus, *every* path to a leaf has at most n edges, and among them at most $s := 2\log^2 N$ can be decreasing. Recall that for *every* node at least one of the out-going edges was decreasing. Assume w.l.o.g. that every node has *exactly* one decreasing edge (if there were two, we simply ignore

one of them). Mark decreasing edges by 1 and the remaining edges by 0. Then every leaf corresponds to a 0-1 string of length at most n with at most s ones. The number of such strings (and hence, the total number of leaves) does not exceed $L(n, s/2)$, where $L(n,t)$ denotes the maximal possible number of leaves in a decision tree of depth n such that *every* path from the root to a leaf has at most t 1-edges.

It remains to estimate $L(n,t)$ for $t = s/2$. Clearly, we have the following recurrence:

$$L(n,t) \le L(n-1,t) + L(n-1,t-1) \text{ with } L(0,t) = L(n,0) = 1. \tag{13.3}$$

By induction on n and t, it can be shown that

$$L(n,t) \le \sum_{i=0}^{t} \binom{n}{i} \le \left(\frac{ne}{t}\right)^t.$$

Indeed, using the identity $\binom{n-1}{k} + \binom{n-1}{k-1} = \binom{n}{k}$, the induction hypothesis together with the recurrence (13.3) yields:

$$\begin{aligned} L(n,t) &\le L(n-1,t) + L(n-1,t-1) \le \sum_{i=0}^{t} \binom{n-1}{i} + \sum_{i=0}^{t-1} \binom{n-1}{i} \\ &= 1 + \sum_{i=1}^{t} \binom{n-1}{i} + \sum_{i=1}^{t-1} \binom{n-1}{i-1} = 1 + \sum_{i=1}^{t} \binom{n}{i} = \sum_{i=0}^{t} \binom{n}{i}. \end{aligned}$$

Thus,

$$\mathrm{Size}(f) \le L(n, s/2) = L(n, \log^2 N) \le \left(\frac{n}{\log^2 N}\right)^{O(\log^2 N)}. \qquad \square$$

RESEARCH PROBLEM 13.9. *Is it possible to improve the upper bound* $\mathrm{Size}(f) \le n^{O(\log^2 N)}$ *in Theorem 13.8 to* $\mathrm{Size}(f) \le 2^{O(\log^2 N)}$*?*

In the next section we will exhibit explicit boolean functions f requiring deterministic decision trees of size $N^{\Omega(\sqrt{\log N})}$ (iterated majority function) and even $N^{\Omega(\log N)}$ (iterated NAND function), where $N = \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$ its the weight of f.

THEOREM 13.10. *There are explicit boolean functions f such that both f and $\neg f$ have DNFs of size N, but any deterministic decision tree for f has size $N^{\Omega(\log N)}$.*

That is, for the size of decision trees we have that $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$. The rest of this section is devoted to the proof of this theorem. For this purpose we will use an argument which has many applications in engineering. The argument is based on harmonic analysis of boolean functions, and is known as the "spectral argument."

13.4.1. Spectral lower bound for decision tree size. Roughly speaking, the main idea of what is known as "spectral argument" is to estimate how far is a given boolean function apart from the parity function. For this it will be convenient to switch to $(-1,+1)$-notation, i.e., to consider boolean functions as mappings from $\{-1,+1\}^n$ to $\{-1,+1\}$, where the correspondence $1 \to -1$ and $0 \to +1$ is assumed. Explicitly, this correspondence is given by the transformation $x' = 1 - 2x$ which transforms the value $x \in \{0,1\}$ into the value $x' \in \{+1,-1\}$; hence, $x' = (-1)^x$. For a boolean function $f : \{0,1\}^n \to \{0,1\}$, its transformed ± 1 version f' is then

$$f'(x_1,\dots,x_n) = 1 - 2 \cdot f\left(\frac{1-x_1}{2}, \frac{1-x_2}{2}, \dots, \frac{1-x_n}{2}\right).$$

Indeed, if the value of the ith variable x_i in f' is $+1$, then the value of the ith variable of f is $(1-1)/2 = 0$, and if x_i has value -1 in f' then it has value $(1+1)/2 = 1$ in f.

Given a function $f : \{-1,1\}^n \to \{-1,1\}$, we can interpret the domain $\{-1,1\}^n$ as 2^n points lying in $\mathbb{R}^n$, and think of f as giving a ± 1 labeling to each of these points. There is a familiar method for interpolating such data points with a polynomial.

EXAMPLE 13.11. Suppose $n = 3$ and f is the Majority function Maj_3. So, in the ± 1 notation we have that $\mathrm{Maj}_3(1,1,1) = 1$, $\mathrm{Maj}_3(1,1,-1) = 1$, ..., $\mathrm{Maj}_3(-1,-1,-1) = -1$. Denoting $x = (x_1, x_2, x_3)$, we can write

$$\begin{aligned}\mathrm{Maj}_3(x) = &\left(\frac{1+x_1}{2}\right)\left(\frac{1+x_2}{2}\right)\left(\frac{1+x_3}{2}\right)\cdot(+1)\\ &+\left(\frac{1+x_1}{2}\right)\left(\frac{1+x_2}{2}\right)\left(\frac{1-x_3}{2}\right)\cdot(+1)\\ &+\quad\cdots\\ &+\left(\frac{1-x_1}{2}\right)\left(\frac{1-x_2}{2}\right)\left(\frac{1-x_3}{2}\right)\cdot(-1).\end{aligned}$$

If we actually expand out all of the products, tremendous cancellation occurs and we get

$$\mathrm{Maj}_3(x) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3\,. \tag{13.4}$$

We could do a similar interpolate-expand-simplify procedure even for a function $f : \{0,1\}^n \to \mathbb{R}$, just by multiplying each x-interpolator by the desired value $f(x)$. Note that after expanding and simplifying, the resulting polynomial will always be *multilinear*, that is, have no variables squared, cubed, etc. In general, a multilinear polynomial over variables $x_1, \ldots, x_n$ has 2^n terms, one for each monomial

$$\chi_S := \prod_{i \in S} x_i\,,$$

where $S \subseteq [n] := \{1, \ldots, n\}$; for $S = \emptyset$ this monomial is constant 1. Hence, every function $f : \{-1,+1\}^n \to \mathbb{R}$ can be expressed (in fact, even uniquely) as a multilinear polynomial

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i = \sum_{S \subseteq [n]} c_S \cdot \chi_S(x), \tag{13.5}$$

where each c_S is a real number. This expression is of f as a linear combination of the monomials χ_S is also known as *Fourier transform* or *Fourier expansion* of f.

We can also treat the functions $f : \{-1,+1\}^n \to \mathbb{R}$ as elements of 2^n-dimensional vector space with an inner product defined by

$$\langle f, g \rangle := 2^{-n} \sum_{x \in \{-1,+1\}^n} f(x)g(x)\,.$$

A convenient way to look at this inner product as a mean value. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be a random vector uniformly distributed in $\{-1,1\}^n$. We can think of generating such an $\boldsymbol{x}$ by choosing each bit x_i independently and uniformly from $\{-1,1\}$. Hence, $\mathrm{E}_{\boldsymbol{x}}\left[f(x)\right] = 2^{-n}\sum_x f(x)$, and

$$\langle f, g \rangle = \mathrm{E}_{\boldsymbol{x}}\left[f(x) \cdot g(x)\right]\,.$$

Important observation is that the set of all monomials χ_S forms an *orthonormal* basis for the space of functions $f : \{-1,+1\}^n \to \mathbb{R}$. That is,

$$\langle \chi_S, \chi_T \rangle = \begin{cases} 1 & \text{if } S = T, \\ 0 & \text{if } S \neq T. \end{cases}$$

Indeed,

$$\langle \chi_S, \chi_T \rangle = \mathrm{E}_{\boldsymbol{x}} \Big[\prod_{i \in S} x_i \cdot \prod_{j \in T} x_j \Big] = \mathrm{E}_{\boldsymbol{x}} \Big[\prod_{i \in S \oplus T} x_i \Big],$$

because $x_i^2 = 1$; here $S \oplus T = (S - T) \cup (T - S)$ is the symmetric difference of sets S and T. Thus, if S and T are identical, then $\langle \chi_S, \chi_T \rangle = 1$. If, however, $S \neq T$ then $S \oplus T \neq \emptyset$, and we obtain:

$$\langle \chi_S, \chi_T \rangle = \mathrm{E}_{\boldsymbol{x}} \Big[\prod_{i \in S \oplus T} x_i \Big] = \prod_{i \in S \oplus T} \mathrm{E}\left[x_i \right] = \prod_{i \in S \oplus T} \Big[\frac{1}{2} \cdot (+1) + \frac{1}{2} \cdot (-1) \Big] = 0 .$$

Since the χ_S form an *orthonormal* basis, the Sth Fourier coefficient c_S of f in the expression (13.5)—which is usually denoted by $\widehat{f}(S)$—is found via (cf. Exercise 13.6):

$$\begin{aligned} \widehat{f}(S) &:= \langle f, \chi_S \rangle \\ &= 2^{-n} \sum_x f(x) \chi_S(x) \\ &= \mathrm{E}_{\boldsymbol{x}} \left[f(x) \cdot \chi_S(x) \right] . \end{aligned}$$

In particular, each coefficient $\widehat{f}(S)$ lies in the interval $[-1,1]$. If $f : \{0,1\}^n \to \{0,1\}$ is a boolean function, then $\widehat{f}(S)$ is defined to be the Sth Fourier coefficient $\widehat{f'}(S)$ of its ± 1 version f'. Hence, for a boolean function f, $\widehat{f}(\emptyset)$ is equal to the probability that the function takes value 1, and for $S \neq \emptyset$, the coefficients

$$\widehat{f}(S) = \Pr_x[f(x) = \oplus_{i \in S} x_i] - \Pr_x[f(x) \neq \oplus_{i \in S} x_i]$$

measure the correlation between the function and the parities of subsets of its arguments.

OBSERVATION 13.12. If the value of f does not depend on the ith variable, that is,

$$f(x_1, \ldots, x_{i-1}, +1, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, -1, x_{i+1}, \ldots, x_n),$$

then $\widehat{f}(S) = 0$ for every S with $i \in S$.

PROOF. For a vector $x \in \{-1,1\}^n$ and a coordinate $i \in [n]$, let $x^{(i)}$ denote the vector x with its ith coordinate x_i replaced by $-x_i$. If $i \in S$, then we have that $f(x^{(i)}) = f(x)$ but $\chi_S(x^{(i)}) = -\chi_S(x)$, implying that $\sum_x f(x)\chi_S(x) = 0$. □

This observation allows to compute Fourier coefficients of arithmetic combination of some functions with disjoint sets of variables.

PROPOSITION 13.13. *Let $S = S_1 \cup S_2$ be a partition of S into two disjoint nonempty blocks. Let $g, h : \{-1,1\}^S \to \{-1,1\}$ be functions such that g only depends on variables x_i with $i \in S_1$, and h only depends on variables x_i with $i \in S_2$. Then*

$$\widehat{f}(S) = \begin{cases} 0 & \text{if } f = g + h; \\ \widehat{g}(S_1) \cdot \widehat{h}(S_2) & \text{if } f = g \cdot h. \end{cases}$$

We leave the proof of this as an exercise.

Fourier coefficient can be used to prove lower bounds on the circuit complexity of boolean functions. So, for example, Linial, Mansour and Nisan (1989) have proved the following lower bound for unbounded fanin DeMorgan $\{\wedge, \vee, \neg\}$-circuits.

THEOREM 13.14. *Let f be a boolean function in n variables computable by a DeMorgan circuit of depth d and size M, and t be any integer. Then*

$$\sum_{|S|>t} \widehat{f}(S)^2 \leq M \cdot 2^{-t^{1/d}/20}.$$

In the case of decision trees we have the following lower bound.

LEMMA 13.15 (Spectral Lower Bound). *For every boolean function f in n variables and every subset of indices $S \subseteq \{1, \ldots, n\}$ we have the bound*

$$\text{Size}(f) \geq 2^{|S|} \cdot \sum_{T \supseteq S} |\widehat{f}(T)|. \tag{13.6}$$

PROOF. Take a decision tree for f of size $\text{Size}(f)$. For a leaf ℓ, let $\text{val}(\ell) \in \{-1, +1\}$ be its label (recall that we are in ± 1-notation), and let I_ℓ be the set of indices of those variables, which are tested on the path to ℓ. Let $B_\ell \subseteq \{-1, +1\}^n$ be the set of all the inputs that reach leaf ℓ; hence, $|B_\ell| = 2^{n-|I_\ell|}$.

Since each input reaches a *unique* leaf, the sets B_ℓ are mutually disjoint. Hence, for every $T \subseteq [n]$,

$$\widehat{f}(T) = 2^{-n} \sum_x f(x) \cdot \chi_T(x) = 2^{-n} \sum_\ell \sum_{x \in B_\ell} f(x) \cdot \chi_T(x) = \sum_\ell \text{val}(\ell) \cdot \Delta(T, \ell),$$

where

$$\Delta(T, \ell) := 2^{-n} \sum_{x \in B_\ell} \chi_T(x).$$

Now, if $T \not\subseteq I_\ell$, that is, if some variable x_i with $i \in T$ is not tested along the path from the root to the leaf ℓ, then $\chi_T(x) = +1$ for exactly half of the inputs $x \in B_\ell$, and hence, $\Delta(T, \ell) = 0$. If $T \subseteq I_\ell$ then the value of χ_T is fixed on B_ℓ to either $+1$ or -1, and so,

$$|\Delta(T, \ell)| = 2^{-n} \cdot |B_\ell| = 2^{-|I_\ell|}.$$

Thus, in both cases, $|\Delta(T, \ell)| \leq 2^{-|I_\ell|}$. Since for any $S \subseteq [n]$ there are only $2^{|I_\ell|-|S|}$ sets T satisfying $S \subseteq T \subseteq I_\ell$, we conclude that

$$\begin{aligned} \sum_{T: T \supseteq S} |\widehat{f}(T)| &\leq \sum_{T: T \supseteq S} \sum_\ell |\Delta(T, \ell)| = \sum_\ell \sum_{T: T \supseteq S} |\Delta(T, \ell)| \\ &\leq \sum_\ell 2^{-|S|} = 2^{-|S|} \cdot \text{Size}(f), \end{aligned}$$

and the desired bound (13.6) follows. □

We are going to apply Lemma 13.15 for $S = [n]$ to the Iterated Majority function and for $S = \emptyset$ to the Iterated NAND function.

13.4.2. Iterated Majority. Recall that our goal is to exhibit a boolean function f which requires decision tree of size super-polynomial in its weight $N = \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$. For this purpose we take the Iterated Majority function which is defined as follows.

The majority of three boolean variables is given by

$$\mathrm{Maj}_3(x_1, x_2, x_3) = x_1x_2 \vee x_1x_3 \vee x_2x_3 .$$

In Example 13.11 we have shown that in the $(-1,+1)$-representation (i.e., when the correspondence $1 \to -1$ and $0 \to +1$ is assumed) we have that

$$\mathrm{Maj}_3(x_1, x_2, x_3) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3 .$$

Consider now the monotone function F_h in $n = 3^h$ variables which is defined by the balanced read-once formula of height h in which every gate is Maj_3, the majority of 3 variables. That is, $F_0 = x$, $F_1 = \mathrm{Maj}_3(x_1, x_2, x_3)$ and for $h \geq 2$,

$$F_h = \mathrm{Maj}_3(F_{h-1}^{(1)}, F_{h-1}^{(2)}, F_{h-1}^{(3)}) \tag{13.7}$$

where $F_{h-1}^{(\nu)}$ are three copies of F_{h-1} with disjoint(!) sets of variables.

THEOREM 13.16. *Let F_h be the iterated majority function and $N = \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$ be its weight. Then*

$$\mathrm{Size}(F_h) \geq N^{\Omega(\sqrt{\log_2 N})} .$$

PROOF. It can be shown (Exercise 13.10) that the function $F_h(x_1, x_2, \ldots, x_n)$ has

$$n = 3^h = 2^{c \cdot h}$$

variables, where $c = \log_2 3 > 3/2$, and has weight

$$N = 2 \cdot 3^{2^h - 1} = 2^{\Theta(2^h)} = 2^{\Theta(n^{2/3})} .$$

Since $2^{\Omega(n)} \geq 2^{\Omega(\log^{3/2} N)} = N^{\Omega(\sqrt{\log_2 N})}$, it is enough to prove the lower bound

$$\mathrm{Size}(F_h) \geq 2^{\Omega(n)}.$$

To prove this, we will apply Lemma 13.15 with $S = [n] = \{1, \ldots, n\}$. Letting

$$a_h := \left|\widehat{F_h}([n])\right|$$

to denote the absolute value of the leading Fourier coefficient of F_h, this lemma yields

$$\mathrm{Size}(F_h) \geq a_h \cdot 2^n .$$

It remains therefore to prove an appropriate lower bound on a_h. We proceed by induction on h.

Clearly, $a_0 = 1$, since F_0 is a variable (cf. Exercise 13.7), and $a_1 = 1/2$ by the above representation of Maj_3.

For the inductive step recall that in the $(-1,+1)$-representation,

$$\mathrm{Maj}_3(x_1, x_2, x_3) = \frac{1}{2}\left(\sum_{i=1}^{3} x_i - \prod_{i=1}^{3} x_i \right).$$

Thus,

$$F_h = \frac{1}{2}\sum_{\nu=1}^{3} F_{h-1}^{(\nu)} - \frac{1}{2}\prod_{\nu=1}^{3} F_{h-1}^{(\nu)} .$$

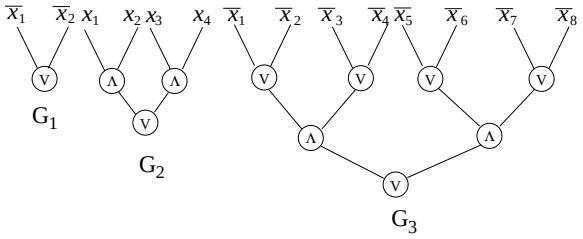


FIGURE 2. Iterated NAND functions G_1, G_2 and G_3.

By Proposition 13.13, the first summand does not contribute to $\widehat{F}_h([n])$ and we obtain that

$$a_h = \frac{1}{2}\, a_{h-1}^3 .$$

Together with the condition $a_0 = 1$, this recursion resolves to

$$a_h = 2^{-3^0} \cdot a_{h-1}^3 = 2^{-3^0-3^1} \cdot a_{h-2}^3 = 2^{-3^0-3^1-3^2} \cdot a_{h-3}^3 = \ldots = 2^{-\Delta},$$

where

$$\Delta = 3^0 + 3^1 + 3^2 + \cdots + 3^{h-1} = \frac{3^h - 1}{3-1} = (3^h - 1)/2 = (n-1)/2 .$$

Thus

$$\mathrm{Size}(F_h) \geq a_h \cdot 2^n \geq 2^{-(n-1)/2} \cdot 2^n = 2^{(n+1)/2},$$

as desired. □

13.4.3. Iterated NAND. Consider the function in $n = 2^h$ variables which is computed by the balanced read-once formula of height h in which every gate is NAND, the negated AND operation $NAND(x,y) = \neg(x \wedge y) = \neg x \vee \neg y$. Up to complementation of the inputs this is equivalent to a monotone read-once formula with alternating levels of AND and OR gates (see Fig. 2). Let us denote this function by G_h.

THEOREM 13.17. $\mathrm{Size}(G_h) \geq \exp\left(\Omega(\log^2 N_h)\right)$, *where* $N_h := N(G_h)$.

PROOF. $\mathrm{dnf}(G_0) = \mathrm{dnf}(\neg G_0) = 1$ (since G_0 is a single variable), and it is easy to see that for every $h \geq 1$ we have $\mathrm{dnf}(G_h) \leq 2 \cdot \mathrm{dnf}(\neg G_{h-1})$ and $\mathrm{dnf}(\neg G_h) \leq \mathrm{dnf}(G_{h-1})^2$. By induction on h one obtains $\mathrm{dnf}(G_h) \leq 2^{2^{(h+1)/2}-1}$ and $\mathrm{dnf}(\neg G_h) \leq 2^{2^{(h/2)+1}-2}$. Hence, we have $N_h \leq 2^{2^{(h/2)+1}}$. Since $n = 2^h$, our statement boils down to showing

$$\mathrm{Size}(G_h) \geq 2^{\Omega(n)} .$$

Let us say that a Fourier coefficient $\widehat{G}_h(S)$ is *dense* if for every subtree of height 2, S contains the index of at least one of the four variables in that subtree. We are going to calculate exactly the sum of absolute values of dense coefficients. Denote this sum by c_h. Note that in the $(-1,+1)$-representation, we have $NAND(x,y) = (xy - x - y - 1)/2$. Hence,

$$G_h = \frac{1}{2}\left(G_{h-1}^{(1)} \cdot G_{h-1}^{(2)} - G_{h-1}^{(1)} - G_{h-1}^{(2)} - 1\right), \tag{13.8}$$

where $G_{h-1}^{(1)}$, $G_{h-1}^{(2)}$ are two copies of G_{h-1} with disjoint sets of variables.

In order to compute c_2, we use the following transformation. Let $f_1 = G_1^{(1)} + 1/2$ and $f_2 = G_1^{(2)} + 1/2$. Then it follows from (13.8) that

$$G_2 = \frac{1}{2} f_1 f_2 - \frac{3}{4} f_1 - \frac{3}{4} f_2 + \frac{1}{8}.$$

Since each monomial in f_1 and f_2 contains at least one variable and the sets of variables of f_1 and f_2 are disjoint, there are no common monomials in the four terms in the above expression of G_2. Hence, it is easy to calculate the sum of the absolute values of the coefficients in the non-constant monomials, which is $c_2 = 1/2 \cdot r_1 \cdot r_2 + 3/4 \cdot (r_1 + r_2) = 27/8 = 3.375$, where $r_1 = r_2 = 3/2$ is the sum of the absolute values of the coefficients in f_1 and f_2.

In order to compute c_h for $h > 2$, we use (13.8) directly. Only the first term $G_{h-1}^{(1)} \cdot G_{h-1}^{(2)}$ in this equation can contribute to dense coefficients, and its individual contributions do not cancel each other. Hence, we have the recursion

$$c_h = \frac{1}{2} c_{h-1}^2.$$

This resolves to $c_h = 2(c_2/2)^{2^{h-2}}$ which is $2^{\Omega(n)}$ since $c_2 > 2$. The proof is now completed by applying Lemma 13.15 (this time with $S = \emptyset$). □

13.5. Decision trees for search problems

So far we considered decision trees solving *decision problems*. That is, for each input the decision tree must give an answer "yes" (1) or "no" (0). For example, if $n = \binom{v}{2}$ then each input $x \in \{0,1\}^n$ can be interpreted as a graph G on v vertices, where $x_e = 1$ means that the edge e is present in G, and $x_e = 0$ means that the edge e is not present in G. There are a lot of decision problem for graphs. Is the graph connected? Has the graph a clique of size k? Is the graph colorable by k colors?

But decision alone is often not that what we actually need. Knowing the answer "the graph has a triangle", we would like to find any of these triangles. Given an unsatisfiable CNF and an assignment to its variables, we would like to find a clause which is not satisfied.

In general, a search problem is specified by n boolean variables and a collection W of "witnesses." In addition, this collection must have the property that every assignment to the n variables is associated with at least one witness.

That is, a *search problem* is specified by a relation $F \subseteq \{0,1\}^n \times W$ such that, for every $x \in \{0,1\}^n$ the exists at least one $w \in W$ such that $(x,w) \in F$. The problem itself is:

Given an input string $x \in \{0,1\}^n$, find a witness $w \in W$ such that $(x,w) \in F$.

With every boolean function $f : \{0,1\}^n \to \{0,1\}$ we can associate the relation $F \subseteq \{0,1\}^n \times W$, where $W = \{0,1\}$ and $(x,w) \in F$ iff $f(x) = w$. Hence, decision problems (=boolean functions) are special case of search problems.

EXAMPLE 13.18. Consider the graphs G_x on v vertices, encoded by binary strings $x \in \{0,1\}^n$ of length $n = \binom{v}{2}$, one bit for each potential edge. As a set W of witnesses we can take some special element λ and the set of all triangles. Define the relation F by: $(x,w) \in F$ if $w = \lambda$ and graph G_x is triangle-free, or $w \neq \lambda$ and w is a triangle in G_x. Then the search problem is, given an input $x \in \{0,1\}^n$, either to answer "no triangle" if G_x is triangle-free, or to find a triangle in G_x.

Given a bipartite graph $G = (U \cup V, E)$, define the search problem Degree(G) in the following way. We have $|E|$ variables x_e, one for each edge $e \in E$. Each assignment $x \in \{0,1\}^E$ to these variables is interpreted as a subgraph G_x of G, defined by those edges e for which $x_e = 1$, that is, $G_x = \{e \in E \mid x_e = 1\}$. The search problem Degree(G) is:

Given an input vector x, find a vertex whose degree in G_x is not one.

It is clear that such a vertex always exist, as long as the sides of the graph are not equal. Thus, as long as $|U| \neq |V|$, Degree(G) is a valid search problem. Note also that Degree(G) can be solved by a *nondeterministic* decision tree of depth at most d, where d is the maximum degree of G. For this it is enough to guess a vertex of degree $\neq 1$ and check the incident edges of this vertex.

We will now show that *deterministic* decision trees must have much larger depth. For this we take a bipartite $(2n) \times n$ graph $G = (U \cup V, E)$ of maximum degree d. Suppose that G has the following expansion property: every subset $S \subseteq U$ of $|S| \leq n/4$ vertices has at least $2|S|$ neighbors in V. Such graphs exist for $d = O(1)$ and infinitely many n's, and can be efficiently constructed using known expander graphs.

THEOREM 13.19. *If G has an expansion property then every deterministic decision tree for* Degree(G) *requires depth* $\Omega(n)$.

PROOF. At each step, Bob (a deterministic decision tree) queries some edge $e \in E$. Based on what edges Bob has queried so far, Alice (the adversary) answers either "$x_e = 1$" (the edge e is present) or "$x_e = 0$" (the edge e is not present) in the subgraph. We will show that Alice can cause Bob to probe $\Omega(n)$ edges of G. The adversary will be limited to produce, in each step, a subgraph in which all vertices in U have degree at most 1 and all vertices in V have degree exactly 1. Hence, the answer in a vertex in U.

To describe the adversary strategy we need some definitions. For step i (after i edges were already probed), let G_i be the subgraph of G obtained by removing all edges $e \in E$ such that:

- $x_e = 0$, that is, the edge e was already probed and was not included in the subgraph;
- $x_e \neq *$ (edge e was not probed yet) but $e \cap e' \neq \emptyset$ for some e' with $x_{e'} = 1$.

That is, G_i contains all the edges that are still possible for the adversary to use in her final subgraph without violating the above limitations.

A set $S \subseteq U$ cannot be matched to V in G_i if it has fewer than $|S|$ neighbors in V, that is, if $|N_i(S)| < |S|$ where $N_i(S) = \{v \in V \mid (u,v) \in G_i \text{ for some } u \in S\}$. Let $S(G_i)$ denote a minimum cardinality unmatchable set in G_i.

By the above limitation on the adversary, at step i the subgraph G_i is a (partial) matching from U to V. Bob cannot know the answer as long as there is no isolated vertex in G_i. Such a vertex itself is a minimum unmatchable set of size 1.

Initially, since the graph G has an expansion property, we have that $|S(G)| > n/4$. Thus, Alice's strategy is to make sure that the minimum unmatchable set size does not decrease too fast.

To describe her strategy, suppose that an edge $e \in E$ is probed in step i (after i edges were already probed). In order to give an answer "$x_e = 1$" or "$x_e = 0$" Alice first constructs two sets of vertices:

- $S^0(e) := S(G_i - e)$, that is, the minimum unmatchable set that would occur if Alice would answer "$x_e = 0$".

- $S^1(e) := S(G_i - \{e' \mid e' \neq e, x_{e'} = *,\ v \in e'\})$, that is, the minimum unmatchable set that would occur if Alice would answer "$x_e = 1$".

Alice than chooses the answer on e so as to make $S(G_{i+1})$ the larger of $S^0(e)$ and $S^1(e)$. The heart of the argument is the following claim

CLAIM 13.20. $|S(G_{i+1})| \geq \frac{1}{2}|S(G_i)|$.

PROOF. Assume e is asked in step $i+1$. By the above strategy,

$$|S(G_{i+1})| = \max\{|S^0(e)|, |S^1(e)|\}\,.$$

Consider the set $S = S^0(e) \cup S^1(e)$. This set cannot be matched into V in G_i, for otherwise either $S^0(e)$ or $S^1(e)$ would be matchable after the decision about e is made. Thus, S contains an unmatchable set for step i of cardinality no more than

$$|S^0(e) \cup S^1(e)| \leq 2 \cdot \max\{|S^0(e)|, |S^1(e)|\} = 2 \cdot |S(G_{i+1})|. \qquad \square$$

We can now complete the proof of the theorem by the following argument. During the game between Alice and Bob, a sequence $S_0, S_1, \ldots, S_t$ of minimum unmatchable sets $S_i = S(G_i)$ of vertices in U is constructed. At the beginning $|S_0| > n/4$, and $|S_t| = 1$ at the end. Moreover, by Claim 13.20, we have that the cardinality of the S_i does not decrease by more than a factor of 2. It must therefore be a step i at which $n/16 \leq |S_i| \leq n/8$ and $|N_i(S_i)| < |S_i|$. That is, S_i has fewer than $|S_i|$ neighbors in the ith subgraph G_i of G. However, by the expansion property of G, the set S_i has had at least $2|S_i|$ neighbors in the original graph G. Since at each step and for any set, the number of its neighbors can drop down by at most a factor of $1/d$, at least $|S_i|/d = \Omega(n)$ edges were probed up to step i. $\square$

Exercises

EX. 13.1. Consider the following function $f(X)$ on $n = m^2$ boolean variables:

$$f = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} x_{ij}\,. \tag{13.9}$$

Show that for this function f we have that $D_0(f) = D_1(f) = m$ but $\mathrm{Depth}(f) = m^2$.
Hint: Take an arbitrary deterministic decision tree for f and construct a path from the root by the following "adversary" rule. Suppose we have reached a node v labeled by x_{ij}. Then follow the outgoing edge marked by 1 if and only if *all* the variables x_{il} with $l \neq j$ were already tested before we reached the node v.

EX. 13.2. Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, and let $k = k(f)$ be the largest natural number such that $|f^{-1}(0)|$ is divisible by 2^k. Show that $\mathrm{Depth}(f) \geq n - k(f)$. *Hint*: The number of inputs $\boldsymbol{x} \in f^{-1}(0)$ leading to a given leaf of depth d is either 0 or 2^{n-d}.

EX. 13.3. Let D_1 be a DNF of a boolean function f, and D_2 be a DNF of its negation $\neg f$. Show the following *cross-intersection property*: if K is a monomial in D_1 then every monomial in D_2 contains at least one literal which is contradictory to at least one literal in K.

EX. 13.4. Show that, for the boolean function f defined by (13.9), we have that $\mathrm{Size}(f) \geq 2^{\mathrm{dnf}(f)}$.
Hint: Observe that all the minterms and maxterms of f have length m. Show that every such function requires a decision tree of size at least 2^m.

Ex. 13.5. Show that in $(-1,+1)$ notation the $\mathrm{AND}(x,y)$ turns to the function $\mathrm{AND}(x,y) = (x+y-x\cdot y+1)/2$. What about $\mathrm{OR}(x,y)$? What about the parity function $x \oplus y$?

Ex. 13.6. Show that the Sth Fourier coefficient $\widehat{f}(S)$ a function $f : \{-1,+1\}^n \to \{-1,+1\}$ is found via:

$$\widehat{f}(S) := \langle f, \chi_S \rangle = 2^{-n} \sum_x f(x)\chi_S(x)\,.$$

Hint: Suppose that $f = \sum_S \widehat{f}(S)\cdot\chi_S$. To find $\widehat{f}(T)$, take the scalar product of f with χ_T.

Ex. 13.7. Let $f = x_i$ be a single variable. Show that then $\widehat{f}(\{i\}) = 1$.

Ex. 13.8. What is the leading Fourier coefficient $\widehat{f}([n])$ of the parity function $f = x_1 \oplus x_2 \oplus \cdots \oplus x_n$?

Ex. 13.9. The most basic result in Fourier analysis is the following fact, known as *Perseval's Theorem*. Let $\boldsymbol{x} = (x_1,\ldots,x_n)$ be a random vector uniformly distributed in $\{-1,1\}^n$. We can think of generating such an $\boldsymbol{x}$ by choosing each bit x_i independently and uniformly from $\{-1,1\}$. Hence, $\mathrm{E}_{\boldsymbol{x}}\left[f(x)\right] = 2^{-n}\sum_x f(x)$.

Prove that, for every $f : \{-1,1\}^n \to \mathbb{R}$,

$$\sum_{S\subseteq[n]} \widehat{f}(S)^2 = \mathrm{E}_{\boldsymbol{x}}\left[f(x)^2\right]\,.$$

In particular, for $f : \{-1,1\}^n \to \{-1,1\}$, we have that

$$\sum_{S\subseteq[n]} \widehat{f}(S)^2 = 1\,.$$

Hint: Just compute $\mathrm{E}_{\boldsymbol{x}}\left[f(x)^2\right]$ using: (1) linearity of expectation, (2) the fact that $\mathrm{E}[X\cdot Y] = \mathrm{E}[X]\cdot\mathrm{E}[Y]$, if random variables X and Y are independent, (3) $\chi_S(x)\cdot\chi_T(x) = \chi_{S\oplus T}(x)$, where $S\oplus T$ is the symmetric difference of sets S and T, (4) $\mathrm{E}_{\boldsymbol{x}}\left[\chi_S(x)\right] = 0$, unless $S = \emptyset$, in which case its is 1.

Ex. 13.10. Show that the iterated majority function F_h, defined by (13.7), has $n = 3^h$ variables and its weight is $2\cdot 3^{2^h-1}$.

Hint: Observe that: (1) $\mathrm{dnf}(F_0) = 1$ and $\mathrm{dnf}(F_h) = 3\cdot\mathrm{dnf}(F_{h-1})^2$, and (2) the minimal DNF of the negation $\neg F_h$ coincides with the DNF of F_h with all the variables negated.

Ex. 13.11. A $\vee$-*decision tree* is a generalization of a deterministic decision tree, where at each node an OR $g(x) = \bigvee_{i\in S} x_i$ of some subset of variables can be tested. Hence, decision trees correspond to the case when $|S| = 1$. Consider the threshold-k function $Th_k^n(x_1,\ldots,x_n) = 1$ iff $x_1+\cdots+x_n \geq k$.

Show that any $\vee$-decision tree for Th_k^n requires at least $\binom{n}{k-1}$ leaves.

Hint: Look at Th_k^n as accepting/rejecting subsets of $[n]$. Suppose that some two different $(k-1)$-element subsets $A,B\subseteq[n]$ reach the same leaf. Show that then also the set $C = A\cup B$ will reach that leaf.

Bibliographic Notes

Theorem 13.1 has been re-discovered by many authors in different contexts: Blum and Impagliazzo (1987), Hartmanis and Hemachandra (1991), and Tardos (1989). Theorem 13.2 is due to Nisan (1989). Theorem 13.8 is due to Ehrenfeucht and Haussler (1989). Theorem 13.5 is due to Rivest and Vuillemin (1976). The term "little

birdie principle" as well as the proof of Theorem 13.7 are borrowed from Jeff Erickson. That $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ for the size of decision trees (Section 13.4) was proved in [**86**]. Lemma 13.15 is a combination of Lemma 4 in Linial et al. (1989) with Lemma 5.1 of Kushilevitz and Mansour (1991). Theorem 13.19 is from Lovász et al. (1995).

CHAPTER 14

General Branching Programs

A branching program is a generalization of a decision tree, where instead of a tree, the underlying graph can be an arbitrary directed acyclic graph. The model of branching programs is one of the most fundamental *sequential* (in contrast to *parallel*, like circuits or formulas) model of computations. This model captures in a natural way the deterministic space whereas nondeterministic branching programs do the same for the nondeterministic mode of computation.

14.1. Nechiporuk's bounds for branching programs

The best we can do so far for unrestricted programs is a quadratic lower bound $\Omega(n^2/\log^2 n)$ for deterministic programs, and $\Omega(n^{3/2}/\log n)$ for nondeterministic programs. These bounds can be shown by counting arguments due to Nechiporuk (1966): just compare the number of subfunctions with the number of distinct subprograms.

Recall that a *deterministic branching program* is a directed acyclic graph with one source node and two sinks, i.e., nodes of out-degree 0. The sinks are labeled by 1 (accept) and by 0 (reject). Each non-sink node has out-degree 2, and the two outgoing edges are labeled by the tests $x_i = 0$ and $x_i = 1$ for some $i \in \{1,\dots,n\}$. Such a program computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ in a natural way: given an input vector $a \in \{0,1\}^n$, we start in the source node and follow the unique path whose tests are consistent with the corresponding bits of a; this path is the *computation* on a. This way we reach a sink, and the input a is accepted iff this is the 1-sink.

A *nondeterministic branching program* (or a switching-and-rectifier network) is a directed acyclic graph $G = (V, E)$ with two specified vertices $s, t \in V$, *some* of whose edges are labeled by variables x_i or their negations $\overline{x}_i$. The *size* of G is defined as the number of labeled edges (not vertices!).

Each input $a = (a_1,\dots,a_n) \in \{0,1\}^n$ defines a subgraph $G(a)$ of G obtained by deleting all edges whose labels are evaluated by a to 0, and removing the labels from the remaining edges. Let $|G(a)|$ denote the number of s-t paths in $G(a)$. A network G computes a boolean function in a natural way: it accepts the input a if and only if $|G(a)| > 0$. This is a *nondeterministic* mode of computation: we accept the input if and only if the labels of at least one s-t path in G are consistent with it.

A *parity branching program* is a network with a *counting* mode of computation: we accept the input a if and only if the number of s-t paths consistent with a is odd, i.e., iff $|G(a)| = 1 \mod 2$.

Let $\mathrm{BP}(f)$, $\mathrm{NBP}(f)$ and $\oplus\mathrm{BP}(f)$ denote, respectively, the minimal size of deterministic, nondeterministic and parity branching program computing f.

Let $f(X)$ be a boolean function. Fix a partition of the variable set X into m disjoint subsets $Y_1,\dots,Y_m$. For every $i \in [m]$, let $c_i(f)$ be the number of distinct subfunctions of f on the variables Y_i obtained by fixing the remaining variables to constants in all possible ways.

THEOREM 14.1. *There exist a constant $\varepsilon > 0$ such that for every boolean function f and for every partition of its variables into $m \geq 1$ sets,*

$$\mathrm{BP}(f) \geq \frac{\varepsilon}{\log_2 n} \sum_{i=1}^{m} \log c_i(f) \tag{14.1}$$

and

$$\min\{\mathrm{NBP}(f), \oplus\mathrm{BP}(f)\} \geq \varepsilon \sum_{i=1}^{m} \sqrt{\log_2 c_i(f)}\,. \tag{14.2}$$

PROOF. Take a partition $Y_1, \ldots, Y_m$ of variables of f. For each $i \in [m]$, each setting of constants to variables outside Y_i yields an induced branching program on the nodes labeled by variables from Y_i plus an accept and a reject node. Say there are h_i such nodes. The number of deterministic branching programs on h_i nodes is at most $n^{h_i} h_i^{2h_i}$: there are at most n^{h_i} ways to assign n variables to h_i nodes, and at most $h_i^{2h_i}$ ways to chose the two successors for each of h_i nodes. Thus $n^{h_i} h_i^{2h_i} \geq c_i(f)$. Since $\mathrm{BP}(f) \geq \sum_{i=1}^{m} h_i$, the desired lower bound (14.1) on $\mathrm{BP}(f)$ follows.

To prove the lower bounds on $\mathrm{NBP}(f)$ and $\oplus\mathrm{BP}(f)$, let $G(V,E)$ be the given program, using nondeterministic or parity accepting mode, to compute f. Any fixing of the variables outside Y_i to constants results in a reduced branching program for the resulting subfunction. Let E_i be the edges of E whose labels are literals of variables from Y_i, and let V_i be the set of vertices touched by these edges. Then without loss of generality the reduced program uses only the vertices V_i, on which we have the edges E_i and perhaps some extra edges labeled 1 that resulted from fixing values. But there are at most $2^{|V_i|^2}$ different possible programs, and as $|V_i| \leq 2|E_i|$ and the size of our program is $\sum_{i=1}^{k} |E_i|$, the desired lower bounds (14.2) on $\mathrm{NBP}(f)$ and $\oplus\mathrm{BP}(f)$ follow. □

The *element distinctness* function takes a string $s_1, \ldots, s_m$ of m elements of the set $[m^2] = \{1, \ldots, m^2\}$ and outputs 1 iff all the s_i are distinct. If we encode the elements of $[m^2]$ by binary strings of length $2\log m$, then we obtain a boolean version of this function in $n = 2m \log m$ variables. Consider the input vector in $\{0,1\}^n$ to represent m strings $s_1, \ldots, s_m$ each of length $2\log m$ where $n = 2m \log m$. Define the function ED_n so that it is 1 iff all the s_i are distinct.

THEOREM 14.2 (Nechiporuk 1966). *The element distinctness function ED_n requires deterministic branching programs of size $\Omega(n^2/\log^2 n)$, and nondeterministic as well as parity branching programs of size $\Omega(n^{3/2}/\log n)$.*

PROOF. Take a partition $Y_1, \ldots, Y_m$ of variables according to the blocks $s_1, \ldots, s_m$. We already know (see the proof of Theorem 2.11) that for each of these m blocks s_i there are $\binom{m^2}{m-1}$ ways of setting the remaining s_j's distinctly and each way gives a different subfunction. Hence,

$$c_i(ED_n) \geq \binom{m^2}{m-1} \geq \left(\frac{m^2}{m-1}\right)^{m-1} = 2^{\Theta(m \log m)}\,.$$

Since $m = \Omega(n/\log n)$, Theorem 14.1 yields the desired lower bounds on the size of branching program size of ED_n. □

14.2. Nondeterministic versus counting programs

Our goal is to show that, at the cost of a slight increase of size, every (nondeterministic) network can be simulated by a parity network. That is, in the model of switching networks nondeterminism is not much more powerful than counting. But perhaps more interesting, than the result itself, is its proof: it uses in a non-trivial manner an interesting fact that random weighting of elements will almost surely isolate exactly one member of a family.

14.2.1. The isolation lemma. Let X be some set of n points, and $\mathscr{F}$ be a family of subsets of X. Let us assign a weight $w(x)$ to each point $x \in X$ and let us define the weight of a set E to be $w(E) = \sum_{x\in E} w(x)$. It may happen that several sets of $\mathscr{F}$ will have the minimal weight. If this is not the case, i.e., if $\min_{E\in\mathscr{F}} w(E)$ is achieved by a unique $E \in \mathscr{F}$, then we say that w is *isolating for* $\mathscr{F}$.

LEMMA 14.3. *Let $\mathscr{F}$ be a family of subsets of an n-element set X. Let $\boldsymbol{w} : X \to \{1,\ldots,N\}$ be a random function, each $\boldsymbol{w}(x)$ independently and uniformly chosen over the range. Then*

$$\Pr[\boldsymbol{w} \text{ is isolating for } \mathscr{F}] \geq 1 - \frac{n}{N}.$$

PROOF. For a point $x \in X$, set

$$\alpha(x) = \min_{E\in\mathscr{F};\, x\notin E} \boldsymbol{w}(E) - \min_{E\in\mathscr{F};\, x\in E} \boldsymbol{w}(E - \{x\}).$$

A crucial observation is that evaluation of $\alpha(x)$ does not require knowledge of $\boldsymbol{w}(x)$. As $\boldsymbol{w}(x)$ is selected uniformly from $\{1,\ldots,N\}$,

$$\Pr[\boldsymbol{w}(x) = \alpha(x)] \leq 1/N,$$

so that

$$\Pr[\boldsymbol{w}(x) = \alpha(x) \text{ for some } x \in X] \leq n/N.$$

But if $\boldsymbol{w}$ had two sets $A, B \in \mathscr{F}$ of minimal weight $\boldsymbol{w}(A) = \boldsymbol{w}(B)$ and $x \in A - B$, then

$$\min_{E\in\mathscr{F};\, x\notin E} \boldsymbol{w}(E) = \boldsymbol{w}(B),$$

$$\min_{E\in\mathscr{F};\, x\in E} \boldsymbol{w}(E - \{x\}) = \boldsymbol{w}(A) - \boldsymbol{w}(x),$$

so $\boldsymbol{w}(x) = \alpha(x)$. Thus, if $\boldsymbol{w}$ is *not* isolating for $\mathscr{F}$ then $\boldsymbol{w}(x) = \alpha(x)$ for some $x \in X$, and we have already established that the last event can happen with probability at most n/N. □

14.2.2. Counting is powerful. Using the isolation lemma we can now show that every (nondeterministic) network may be simulated by a parity network.

THEOREM 14.4. *There is a constant c such that for every boolean function f in n variables,*

$$\oplus\mathrm{BP}(f) \leq cn \cdot \mathrm{NBP}(f)^5.$$

PROOF. Let $G = (V, E)$ be a directed graph and $w : E \to \{1,\ldots,2\cdot|E|\}$ a weight function on its edges. The weight of an s-t path is the sum of weights of its edges; a path is *lightest* if its weight is minimal. Let $d_w(G)$ denote the weight of the shortest s-t path in G; hence,

$$d_w(G) \leq M := 2|V| \cdot |E|.$$

Having a weight function w and an integer l, define the (unweighted, layered) version $G^l_w = (V', E')$ of G as follows. Replace every vertex $u \in V$ by $l + 1$ new vertices

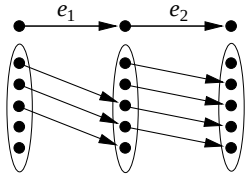


FIGURE 1. $l = 4$, $w(e_1) = 2$ and $w(e_2) = 1$

$u_0, u_1, \ldots, u_l$ in V' (i.e., V' consists of $l+1$ copies of V, arranged in layers). For every edge $e = (u, v)$ in E and every $0 \le i \le l - w(e)$ we put an edge $\left(u_i, v_{i+w(e)}\right)$ in E' (see Fig. 1); hence, $|V'| \le (1+l)|V|$ and $|E'| \le (l+1)|E|$.

CLAIM 14.5. The graphs G_w^l have the following properties:

(i) if G has no s-t path, then for every w and l, G_w^l has no s_0-t_l path;
(ii) if G has an s-t path and $l = d_w(G)$, then G_w^l has an s_0-t_l path. Moreover, the later path is unique if the lightest s-t path in G is unique.

PROOF. Let $P = (e_1, e_2, \ldots, e_k)$ be an s-t path in G. The first node of this path is s. In the new graph G_w^l the first node is s_0 and, following the path P in this new graph, at the ith edge e_i we move by $w(e_i)$ vertices down (in the next, $(i+1)$th layer of nodes). Hence, P can produce an s_0-t_l path in G_w^l iff $\sum_{i=1}^k w(e_i) \le l$. That is, a graph G_w^l has an s_0-t_l iff G has an s-t path *and* $\sum_{i=1}^k w(e_i) \le l$. For $l = d_w(G)$, only lightest paths can fulfill this last condition. □

Now let $G = (V, E)$ be a network computing a given boolean function $f(x_1, \ldots, x_n)$. Say that a weight function w is *good* for an input $a \in \{0,1\}^n$ if either $G(a)$ has no s-t paths or the lightest s-t path in $G(a)$ is unique.

For each input $a \in \{0,1\}^n$, taking the family $\mathscr{F}_a$ to be all s-t paths in the graph $G(a)$, the isolation lemma (Lemma 14.3) implies that at least one-half of all weight functions w are good for a. By a standard counting argument, there exists a set W of $|W| \le 1 + \log_2(2^n) = n+1$ weight functions such that at least one $w \in W$ is good for every input a. If w is good for a, then the graph $G_w^l(a)$ with $l = d_w(G(a))$ has the properties (i) and (ii). For different inputs a, the corresponding values of l may be different, but they all lie in the interval $1, \ldots, M$. Thus, there exist $m \le (n+1) \cdot M$ networks $H_1, \ldots, H_m$ (with each $H_j = G_w^l$ for some $w \in W$ and $1 \le l \le M$) such that, for every input $a \in \{0,1\}^n$, the following holds:

(iii) if $|G(a)| = 0$, then $|H_j(a)| = 0$ for all j;
(iv) if $|G(a)| > 0$, then $|H_j(a)| = 1$ for at least one j.

Let s_j, t_j be the specified vertices in H_j, $j = 1, \ldots, m$. We construct the desired parity network H as follows: to each H_j add the unlabeled edge (s_j, t_j), identify t_j and s_{j+1} for every $j < m$, and add the unlabeled edge (s_1, t_m) (see Fig. 2).

It is easy to see that, for every input $a \in \{0,1\}^n$, $|H(a)| = 1 \bmod 2$ if and only if $|G(a)| > 0$. Indeed, if $|G(a)| = 0$, then by (iii), $H(a)$ has precisely two s_1-t_m paths (formed by added unlabeled edges). On the other hand, if $|G(a)| > 0$, then by (iv), at least one $H_j(a)$ has precisely one s_j-t_j path, implying that the total number of s_1-t_m paths in $H(a)$ is odd. Thus, H is a parity network computing the same boolean function

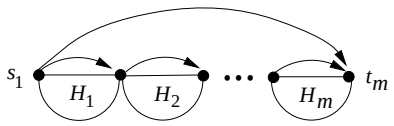


FIGURE 2. Construction of the parity network H

f. Since $l \leq M$ and $m \leq nM$ with $M = 2|V| \cdot |E| \leq 2|E|^2$, the size of (the number of edges in) H is at most $m(l+1)|E| = O(n|E|^5)$. □

Bibliographic Notes

The first estimate (14.1) in Theorem 14.1 (for deterministic branching programs) is due to Nechiporuk (1966). Pavel Pudlák extended this to nondeterministic branching programs, and Karchmer and Wigderson (1993) to parity branching programs. Isolation Lemma (Lemma 14.3) is due to Mulmuley, U. Vazirani and V. Vazirani (1987). The poof we gave is due to Spencer (1995). Theorem 14.4 was proved by Wigderson (1994).

CHAPTER 15

Bounded Width

To define the width of a branching program, divide the nodes of the underlying graph into levels such that all edges out of nodes in the ith level go to nodes in the $(i+1)$th level. We can make a graph leveled by adding more nodes, possibly squaring the size but keeping the length (that is, the number of edges in a longest path) the same. The *width* is then the number of nodes of the largest level in an optimal division into levels. A leveled program is *oblivious* if in each level all its nodes are labeled by the same variable. It is not difficult to see that every leveled branching program of length ℓ and width w can be transformed into an oblivious program of length $w\ell$ and width w.

15.1. Width versus length

Every boolean function in n variables can be computed by a trivial branching program of length $\ell = n$ and width $w = 2^n$: just take a decision tree. But what if we restrict the width w—how long then the program must be? To answer this question we use communication complexity arguments.

An *s-mixed protocol* for a boolean function $f : \{0,1\}^n \to \{0,1\}$ is a communication protocol between two players, Alice and Bob, whose access to input variables fulfills the following conditions:

a. Alice cannot see at least s variables seen by Bob, and
b. Bob cannot see at least s variables seen by Alice.

The remaining $n - 2s$ variables are seen by both players!

Let $D_s(f)$ denote the minimum number of bits communicated by a best deterministic s-mixed protocol for f. The larger the number $n - 2s$ of common variables is, the easier is the game. Hence, $s \le t$ implies that $D_s(f) \le D_t(f)$.

THEOREM 15.1. *If a boolean function $f : \{0,1\}^n \to \{0,1\}$ can be computed by an oblivious branching program of width w and length $\ell \le 0.1n \log n$, then*

$$D_s(f) = O\left(\frac{\ell \log w}{n}\right) \quad \textit{for} \quad s \ge n^{0.6}/4\,.$$

PROOF. Because the branching program is oblivious, we can think of its labels as forming a string z of length ℓ over the alphabet $[n]$. To obtain a communication protocol from the program, we need the following combinatorial result.

Let z be a string over an alphabet $X = \{x_1, \ldots, x_n\}$. Given two sets $S, T \subseteq X$ of letters, say that a string z has an (r, S, R)-partition if z can be partitioned into r substrings $z = z_1 z_2 \cdots z_r$ such that none of the substrings z_i contains letters from both sets S and T.

CLAIM 15.2. Let $A, B \subseteq X$ be two disjoint sets of size $|A| = |B| = m$. Let z be a string over X such that each $a \in A$ appears in z at most k_A times and each $b \in B$ appears in

z at most k_B times. Then there are $A' \subseteq A$ and $B' \subseteq B$ of size at least $m/2^k$ such that z has a (k, A', B')-partition, where $k = k_A + k_B$.

PROOF. Induction on k. If $k = 1$, then either k_A of k_B is 0, and we can take $A' = A$ and $B' = B$. For the induction step, assume w.l.o.g. that each letter appears in z at least once (otherwise, extend z with the missing letters in an arbitrary way). Examine the letters of z one by one until we reach a location where we already have seen $m/2$ letters of one of A and B but less than $m/2$ of the other; such a location must exist since $A \cap B = \emptyset$. Denote the prefix by z' and the rest of z by z''. Let it was A whose $m/2$ letters appear in z' (the case when it is B is dual). Let $A^* = \{a \in A \mid a \in z'\}$ be those letters of A that appear in z', and $B^* = \{b \in B \mid b \notin z'\}$ be those letters of B that *do not* appear in z'. It follows that $|A^*|, |B^*| \geq m/2$.

Consider now the suffix z''. Each letter of A^* appears in z'' at most $k_A - 1$ times, since each of them already appeared in z' at least once. Hence, we can apply the induction hypothesis to the string z'' for sets A^* and B^*, and obtain subsets $A' \subseteq A^*$ and $B' \subseteq B^*$ such that z'' has a $(k-1, A', B')$-partition with $|A'| \geq |A^*|/2^{k-1} \geq m/2^k$ and $|B'| \geq |B^*|/2^{k-1} \geq m/2^k$. Since the prefix z' can only contain letters of A' but none of B', the entire string $z = z'z''$ also has a $(k-1, A', B')$-partition. □

Let now z be the string over $X = \{x_1, \ldots, x_n\}$ of length ℓ corresponding to the labels of our branching program. Observe that at least $n/2$ variables must appear at most $2\ell/n$ times, for otherwise the length of the string would be larger than $(n/2)(2\ell/n) = \ell$. Partition these variables into two sets A and B each of size $n/4$ in an arbitrary way. By Claim 15.2 with $m = n/4$, $k_A = k_B = 2\ell/n$ and $k = 4\ell/n$, there are disjoint sets of variables A' and B' such that $|A'|, |B'| \geq n/(4 \cdot 2^k)$ and z is a (k, A', B')-partition. Moreover, since $\ell \leq 0.1 n \log n$, we have that

$$k = \frac{4\ell}{n} \leq \frac{0.4 n \log n}{n} = 0.4 \log n\,.$$

Hence,

$$|A'|, |B'| \geq n/(4 \cdot 2^k) \geq n^{0.6}/4\,.$$

Since the sequence z of variables, tested along the ℓ levels of the program, has a (k, A', B')-partition, its is possible to split z into k substrings $z = z_1 z_2 \cdots z_k$ such that no substring z_i contains variables from both subsets A' and B'. Hence, if we give all variables in A' to Alice, all variables in B' to Bob and the rest to both players, the players can determine the value of our function by communicating according to the underlying branching program. To carry out the simulation, the players need to tell each other, at the end of each of k blocks, the name of the node in the next level from which the simulation should proceed; for this $\log w$ bits are sufficient. Hence, the obtained protocol communicates $O(k \cdot \log w) = O((\ell \log w)/n)$ bits in total. The protocol is s-mixed for $s \geq \min\{|A'|, |B'|\} \geq n^{0.6}/4$. □

Thus, to obtain a large tradeoff between the width and the length of oblivious branching programs, we need boolean functions of large mixed communication complexity. We will now show that such are characteristic functions of good codes.

A linear (n, m, d)-code is a linear subspace $C \subseteq GF(2)^n$ of dimension $n - m$ such that the Hamming distance between any two vectors in C is at least $2d+1$. An (n, m, d)-code function is the characteristic function f_C of a linear (n, m, d)-code C, that is, $f_C(x) = 1$ iff $x \in C$.

LEMMA 15.3. *For every (n,m,d)-code function f, we have that*

$$D_s(f_C) \geq 2d \log_2 \binom{s}{d} - m\,.$$

PROOF. Take an arbitrary s-mixed protocol for $f(X)$. Let $A \subseteq X$ be the set of variables seen only by Alice, and $B \subseteq X$ be the set of variables seen only by Bob. Hence, $|A|, |B| \geq s$, and at most $r = n - 2s$ variables are seen by both players. We can assume w.l.o.g. that $|A| = |B| = s$. Since there are only 2^r possible settings α of constants to these (common) variables, at least one of these settings gives us a subfunction f_α of f in $2s$ variables which is the characteristic function of some linear $(n-r, m-r, d)$-code C.

After this setting, our protocol turns to a usual communication protocol for the truth matrix $M = \{f_\alpha(x,y)\}$ of f_α. From Section 7.1.3 we know that this last protocol must communicate at least $\log_2 \mathrm{Cov}(M)$ bits, where $\mathrm{Cov}(M)$ is the smallest number of (not necessarily disjoint) all-1 submatrices of M covering all its 1s. (In fact, $\log_2 \mathrm{Cov}(M)$ is a lower bound even for *nondeterministic* communication complexity of M, but we will not need this now.) Since the matrix has $|M| = 2^{n-m-r} = 2^{2s-m}$ ones, the desired lower bound on $\log_2 \mathrm{Cov}(M)$, and hence, on $D_s(f_C)$ follows from:

CLAIM 15.4. Every all-1 submatrix of M has at most $2^{2s}\binom{s}{d}^{-2}$ ones.

To show this, look at one row $x \in \{0,1\}^s$ of M. Since the Hamming distance between any two vectors in C is at least $2d+1$, we have that any two columns $y \neq y' \in \{0,1\}^s$ of M such that $M[x,y] = M[x,y'] = 1$ must also be at Hamming distance at least $2d+1$. Hence, no Hamming ball of radius d over a column y with $M[x,y] = 1$ can contain another column y' with $M[x,y'] = 1$. Since each such ball has $\sum_{i=0}^{d}\binom{s}{i} > \binom{s}{d}$ vectors, this implies that each row and each column of M can have at most $2^s\binom{s}{d}^{-1}$ ones.

This completes the proof of the claim, and thus, of Lemma 15.3. □

Since the parity-check matrix of any linear (n,m,d)-code C has m rows, the characteristic function f_C of C is just an AND of m negations of parity functions. This AND can be computed by an oblivious branching program of width $w = m = O(d \log n)$ and length $\ell = mn = O(dn \log n)$.

If, however, we would require the length be smaller than $n \log n$, then some linear codes would require exponential width. To see this, consider Bose-Chaudhury codes (BCH-codes). These are linear (n,m,d)-codes C with $m \leq d \log_2(n+1)$. Such codes can be constructed for any n such that $n+1$ is a power of 2, and for every $d < n/2$.

COROLLARY 15.5. *Let C be a BCH-code of minimal distance $2d+1$ where $d = \lfloor n^{0.01} \rfloor$. Then any oblivious branching program for f_C must either have width exponential in $n^{0.01}$ or have length $\ell = \Omega(n \log n)$.*

PROOF. Since $m \leq d \log_2(n+1)$, Lemma 15.3 implies that, for $s = \Omega(n^{3/5})$,

$$D_s(f_C) \geq 2d \log_2(s/d) - m \geq 0.59 \cdot 2d \log_2 n - d \log_2(n+1) - O(1) = \Omega(d \log n)\,.$$

Hence, Theorem 15.1 implies that $\ell \log w = \Omega(dn \log n) = \Omega(n^{1.01} \log n)$. □

Recall that the Majority function Maj is defined by:

$$\mathrm{Maj}(x_1, \ldots, x_n) = 1 \quad \text{iff} \quad x_1 + \cdots + x_n \geq n/2\,.$$

EXERCISE 15.6. Show that any *constant-width* branching program for Maj must have length $\ell = \Omega(n \log n)$. *Hint*: Show that $D_s(\mathrm{Maj}) = \Omega(\log s)$ and use Theorem 15.1.

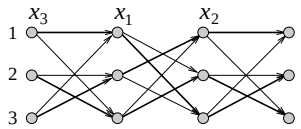


FIGURE 1. A width-3 permuting branching program on three variables. On input vector $x = (0,1,1)$ this program outputs the permutation $P(x) = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$. Bold arrows correspond to tests $x_i = 1$, the remaining ones to tests $x_i = 0$.

15.2. Width-5 programs and formulas

We now consider branching programs of constant(!) width. At first glance, it seems that such a drastical width restriction might be very crucial: if the width is bounded by some constant then, when going from one level to the next, we can keep only a constant amount of information about what we have done before. It was therefore conjectured by many researchers that, due to this "information bottleneck," even such function as the Majority function $\mathrm{Maj}(x_1,\ldots,x_n)$ should require very long branching programs, if their width is constant. Trivial constant-width branching program would try to remember the number of 1's among the bits, which were already read; but this would require non-constant width of about $\log_2 n$.

With a surprisingly simple construction, Barrington (1986) disproved this conjecture. He showed that constant-width branching programs are unexpectedly powerful: every function with a polynomial size DeMorgan formula, including the Majority function, *can* be computed by a width-5 branching program of polynomial length.

The intuition behind his construction is an observation that, when working on a given input vector $a \in \{0,1\}^n$, a branching program collects the information about a not necessarily gradually: if a passed a test $x_i = 1$ it knows the ith bit of a. But this is also the way in which information is collected by decision trees, a very special kind of branching programs! Most important aspect of general branching programs is the possibility to re-test the bits. If a passed also the second test at some node v, the information "$a_i = 1$" is at this point useless. But in this case the additional information about a is encoded in the underlying *graph* of the program, namely, by the fact that a reached this particular node v, and not the other one. That is, a way in which a program collects an information about an input is not gradual but rather global. It is encoded by the structure of the underlying graph.

15.2.1. Permutation branching programs. An arbitrary graph of width w and length l can be converted into a $w \times l$ array of nodes by adding dummy nodes, possibly multiplying the size by w. So, we may assume that our programs have this form. That is, the program has l levels. All levels have w nodes and all nodes at a given level are labeled by the same variable. Moreover, at leach level the 0-edges and the 1-edges going to the next level form two mappings from $[w] = \{1,\ldots,w\}$ to $[w]$ (see Fig. 1). Additionally, we will require these mappings be *permutations*, that is, are bijective mappings. Then any input vector yields a permutation which is the composition of the selected permutations at each level.

Call such a branching program P a *permuting* branching program, and let $P(x)$ be the resulting permutation on input $x \in \{0,1\}^n$. For a boolean function f and a

permutation σ, say that branching program P *σ-computes* f if for every input x,

$$P(x) = \begin{cases} \sigma & \text{if } f(x) = 1, \\ e & \text{if } f(x) = 0, \end{cases}$$

where e is the identity permutation. A permutation is *cyclic* if it is composed of a single cycle on all its elements. For example,

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 5 & 4 & 2 \\ 3 & 5 & 4 & 2 & 1 \end{pmatrix}$$

is a cyclic permutation, which we will denote as

$$1 \to 3 \to 5 \to 4 \to 2 \to 1 \quad \text{or shortly as} \quad \sigma = (13542).$$

15.2.2. Barrington's theorem. The following simple properties of permuting branching programs give a key for the whole Barrington's argument. Let σ and τ be *cyclic* permutations, f and g boolean functions, P and Q permuting branching programs.

LEMMA 15.7 (Changing output). *If P σ-computes f then there is a permuting branching program of the same size τ-computing f.*

PROOF. Since σ and τ are both cyclic permutations, we may write $\tau = \theta\sigma\theta^{-1}$ for some permutation θ. Then simply reorder the left and right nodes of P according to θ to obtain the τ-computing branching program P':

$$\text{if } P(x) = \sigma_1\sigma_2\cdots\sigma_t = \sigma \text{ then } P'(x) = \theta\sigma_1\sigma_2\cdots\sigma_t\theta^{-1} = \theta\sigma\theta^{-1} = \tau.$$

That is, we replace the permutation σ_1 computed at the first layer by the permutation $\theta\sigma_1$, and the permutation σ_t computed at the last layer by the permutation $\sigma_t\theta^{-1}$. □

LEMMA 15.8 (Negation). *If P σ-computes f then there is a permuting branching program of the same size σ-computing $\neg f$.*

PROOF. Use the previous lemma to obtain a branching program P' σ^{-1}-computing f. Hence, $P'(x) = \sigma^{-1}$ if $f(x) = 1$, and $P'(x) = e$ if $f(x) = 0$. Then reorder the final level by σ so that the resulting program P'' σ-computes $\neg f$:

$$\text{if } P'(x) = \sigma_1\sigma_2\cdots\sigma_t \text{ then } P''(x) = \sigma_1\sigma_2\cdots\sigma_t\sigma.$$

This way, $P''(x)$ outputs e if $P'(x) = \sigma^{-1}$, and hence, if $f(x) = 1$; otherwise, $P''(x)$ outputs σ. □

LEMMA 15.9 (Computing AND). *If P σ-computes f and Q τ-computes g, then there is a permuting branching program $\sigma\tau\sigma^{-1}\tau^{-1}$-computing $f \wedge g$ of size*

$$2(\text{size}(P) + \text{size}(Q)).$$

PROOF. Use Lemma 15.7 to get a program σ^{-1}-computing f and τ^{-1}-computing g. Then compose these four programs in the order $\sigma, \tau, \sigma^{-1}, \tau^{-1}$. This has the desired effect because replacing either σ or τ by e in $\sigma\tau\sigma^{-1}\tau^{-1}$ yields e. □

The next lemma is the only place where the value $w = 5$ is important; neither $w = 3$ nor $w = 4$ suites for this purpose.

LEMMA 15.10. *There are cyclic permutations σ and τ of $\{1,2,3,4,5\}$ such that their commutator $\varrho = \sigma\tau\sigma^{-1}\tau^{-1}$ is cyclic.*

PROOF. See Fig. 2. □

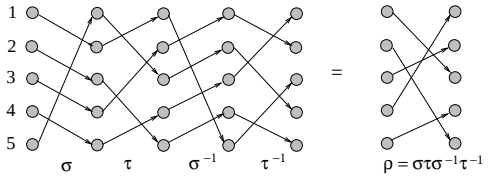


FIGURE 2. Cyclic permutations $\sigma = (12345)$, $\tau = (13542)$ of $\{1,2,3,4,5\}$ and their commutator $\varrho = \sigma\tau\sigma^{-1}\tau^{-1} = (13254)$.

THEOREM 15.11. *Suppose that a boolean function f be computed by a DeMorgan circuit of depth d. Then f is also computable by a width-5 branching program of length at most 4^d.*

In particular, if a boolean function f can be computed by a DeMorgan formula of polynomial leafsize, then f can be computed by a width-5 branching program of polynomial size. That is, width-5 branching programs are not weaker than DeMorgan formulas!

PROOF. We will prove a somewhat stronger claim: If f can be computed by a DeMorgan circuit of depth d, then there exists a *cyclic* permutation σ of $\{1,2,3,4,5\}$ and a permutation branching program P of width 5 such that P σ-computes f and has size at most 4^d. We prove this claim by the induction on the depth d.

If $d = 0$, the whole circuit for f is either a variable x_i or its negation $\neg x_i$, and f can be easily computed by one-instruction program.

Suppose now that $d \geq 1$. By Lemma 15.8, we can assume that $f = g \wedge h$, where g and h have formulas of depth $d-1$, and thus (by induction hypothesis) width-5 permuting branching programs G and H of length at most 4^{d-1}.

Let σ and τ be the permutations from Lemma 15.10. By Lemma 15.7, we may assume that G σ-computes g and H τ-computes h. By Lemma 15.9, there is a permuting program of size at most $2(\text{size}(G) + \text{size}(H)) \leq 4^d$ which $\sigma\tau\sigma^{-1}\tau^{-1}$-computes f. Since, by Lemma 15.10, the permutation $\sigma\tau\sigma^{-1}\tau^{-1}$ is cyclic, we are done. □

Bibliographic Notes

Theorem 15.1 is due to Alon and Maass (1988). Theorem 15.11 was proved by Barrington (1989).

CHAPTER 16

Bounded Replication

We have already seen that restricting the width of a branching program to a constant does not reduce their power too much: the resulting model is at least as powerful as boolean formulas. But what if we restrict the *length* of a program—can we then show that some explicit boolean functions require exponential width?

Restricting the length means restricting the number of repeated tests along a computation. In this chapter we consider branching programs in which we restrict the number of variables that are allowed to be tested more than once.

Namely, define the *replication number* of a program as the minimal number R such that along every computation path, at most R variables are tested more than once. The sets of variables re-tested along different computations may be different! Also, the (up to R) re-tested variables may be re-tested an arbitrary number of times! Thus, restricted replication does not mean restricted length of computations—they may be arbitrarily long. Branching programs with replication number R are also called in the literature *branching* $(1,+R)$*-programs*, meaning that we have a read-one branching program with up to R exceptions along each computation.

Note that for *every* branching program in n variables we have $0 \leq R \leq n$. Moreover, every boolean function f in n variables can be computed by a branching program with $R = 0$: just take a decision tree. However, the size S of such (trivial) branching programs is then exponential for most functions. It is therefore interesting to understand whether S can be substantially reduced by allowing larger values of R.

The goal is to prove exponential lower bounds on the size of branching programs of as large replication number R as possible. An ultimate goal would be to do this for $R = n$: then we would have an exponential lower bound for *unrestricted* branching programs.

In this chapter we will come quite "close" to this goal by exhibiting boolean functions f (based on expander graphs) with the following property: there is an absolute constant $\varepsilon > 0$ such that every branching program computing f must either have replication number $R > \varepsilon n$ or must have exponential size.

16.1. Read-once programs: $R = 0$

To "warm up", we start with *read-once* branching programs (1-b.p.), that is, programs along each path of which no variable can be tested more than once.

It is not difficult to see that read-once programs are just a small generalization of decision trees. The only difference is that now we count not the total number of nodes (=total number of subtrees) but only the total number of *non-isomorphic* subtrees: if we glue up isomorphic subtrees of a decision tree, then what we obtain is a read-once program. Similarly, if we envelope a read-once program to a tree, then what we obtain is a decision tree.

Since subtrees correspond to subfunctions, the number of non-isomorphic subtrees in T (and hence, the size of P) must be large, if f has many different subfunctions. This motivates the following definition.

Let $f(X)$ be a boolean function in variables $X = \{x_1, \ldots, x_n\}$, and let $Y \subseteq X$. An *assignment* to Y is a mapping $a : Y \to \{0,1\}$, and its *length* is the size $|Y|$ of its domain. The *subfunction* of f, induced by such an assignment, is a boolean function $f_a(X - Y)$ obtained from f by setting the variables $x_i \in Y$ to constants $a(x_i) \in \{0,1\}$. Hence, f_a is a boolean function $f_a : \{0,1\}^{n-|Y|} \to \{0,1\}$.

DEFINITION 16.1. A boolean function $f(X)$ is *m-mixed* if for every $Y \subseteq X$ of size $|Y| = m$ and any two different assignments $a, b : Y \to \{0,1\}$, the obtained subfunctions f_a and f_b are different, that is, take different values on at least one input vector.

LEMMA 16.2. *If f is an m-mixed boolean function, then every deterministic read-once branching program computing f must have at least $2^m - 1$ nodes.*

PROOF. Let P be a deterministic read-once branching program computing f. Our goal is to show that the initial part of P must be a complete binary tree of depth $m-1$. For this, it is enough to show that no two initial paths (starting in the source node) of length $m-1$ can meet in a node. For the sake of contradiction, assume that some two paths p_1 and p_2 of length $m-1$ meet in some node v, and let f_v be a boolean function computed by a subprogram P_v of P with the source node v. Let Y_i be the set of variables tested along the path p_i, and $a_i : Y_i \to \{0,1\}$ be the assignment consistent with this path.

CLAIM 16.3. $Y_1 = Y_2$.

PROOF. Suppose there is a variable $x_i \in Y_2$ such that $x_i \notin Y_1$. Then we can extend the assignment a_1 to two assignments $a_1', a_1'' : Y \cup \{x_i\} \to \{0,1\}$ by setting $a_1'(x_i) = 0$ and $a_1''(x_i) = 1$. Since the variable x_i belongs to Y_2, it was tested along the path p_2. Since our program is read-once, this means that x_i cannot be re-tested in the subprogram P_v. Hence, $f_{a_1'} = f_{a_1''}$. But the assignments a_1' and a_1'' are different and both have length $|Y_1| + 1 = m$, a contradiction with the m-mixness of f. □

By Claim 16.3, both assignments a_1 and a_2 have the same domain $Y = Y_1 = Y_2$. Moreover, these assignments are different since the computations on them split before they meet. Since, due to the read-once constrain, none of the variables in Y can be tested after the node v, we have that $f_{a_1} = f_{a_2}$, a contradiction with the m-mixness of f. □

16.1.1. P ≠ NP ∩ co-NP for read-once programs. We now consider *nondeterministic* branching programs. Call such a program *read-once* (or a 1-n.b.p.) if along any path from the source node to the target node every variable appears at most once. Note that this is a "syntactic" restriction: such a program cannot contain any inconsistent paths, that is, paths along which a variable x_i and its negation $\neg x_i$ is tested.

Just like we have done for the size of decision trees, we can ask the **P** versus **NP ∩ co-NP** question for their (slight) generalization—read-once programs. We will show that here **P ≠ NP ∩ co-NP**.

Namely, we will exhibit a boolean function f in n variables (a "pointer function") such that both f and $\neg f$ have nondeterministic read-once branching programs of polynomial size but any deterministic read-once program for f must have exponential size.

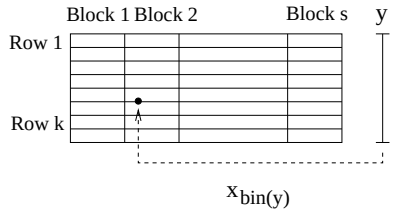


FIGURE 1. The pointer function.

The *pointer function* $f_n(x_1,\ldots,x_n)$ is defined as follows. Let s and k be such that $ks^2 = n$ and $k \geq \log n$. Arrange the n indices $1,\ldots,n$ of the variables into a $k \times s^2$ matrix, split the i-th row ($1 \leq i \leq k$) into s blocks $B_{i1}, B_{i2}, \ldots, B_{is}$ of size s each, and let y_i be the OR of ANDs of variables in these blocks:

$$y_i = \bigvee_{j=1}^{s} \left(\bigwedge_{x \in B_{ij}} x \right) \qquad i = 1,\ldots,k, \tag{16.1}$$

Then define the function by

$$f_n(x_1,\ldots,x_n) = x_{\mathrm{bin}(y)},$$

where $\mathrm{bin}(y) = \sum_{i=1}^{k} y_i 2^{i-1}$ is the number whose binary code is $y = (y_1,\ldots,y_k)$.

THEOREM 16.4. *Both f_n and $\neg f_n$ have 1-n.b.p. of size $O(n)$ whereas any 1-b.p. computing f_n must have size at least $2^{s-1} = \exp\left(\Omega(n/\log n)^{1/2}\right)$.*

PROOF. Upper bound. On input vector $x = (x_1,\ldots,x_n)$ in $\{0,1\}^n$, the desired 1-n.b.p. first guesses a binary string $a = (a_1,\ldots,a_k) \in \{0,1\}^k$, after which it remains to test if the values $y_1 = a_1,\ldots,y_k = a_k$ satisfy the equalities (16.1) and if the corresponding (to the string a) variable $x_v = x_{\mathrm{bin}(a)}$ has the value 1 (or 0 in the case of $\neg f_n$). It is clear that the resulting program is read-once, except that the variable x_v could be tested two times: once – in the program P_i making that of the tests (16.1) for which $v \in B_{i1} \cup \ldots \cup B_{is}$, and then once more at the end of a computation. Simple (but crucial) observation is that we can safely replace the variable x_v in that program P_i by the constant 1 (or by 0, in the case of $\neg f_n$), so that the whole program is read-once.

Lower bound. By Lemma 16.2 it is enough to show that the function f_n is m-mixed for $m = s-1$. To show this take any two different assignments a and b of constants to a set of m variables in X. Since m is strictly less than s, we have that: (i) every block B_{ij} has at least one unspecified variable, and (ii) in every row, at least one block consists entirely of unspecified variables. This means that (independent of actual values of a and b) we can arrange the rest so that the resulting string $(y_1,\ldots,y_k)$ points to a bit x_v where the assignments a and b differ. □

Theorem 16.4 shows that 1-n.b.p. may be exponentially more powerful than their deterministic counterparts, 1-b.p. Thus, it is harder to prove good lower bounds for 1-n.b.p. Still, also here we have a general lower bounds criterion.

For a set of inputs $A \subseteq \{0,1\}^n$ and an integer $0 \leq k \leq n$, we define the *k-th degree* $d_k(A)$ as the maximum number of inputs in A, all of which have 1's on some fixed set of k coordinates. That is,

$$d_k(A) = \max_{|I|=k} \#\{a \in A \mid a_i = 1 \text{ for all } i \in I\}.$$

An input a is a *lower one* of a boolean function f if $f(a) = 1$ and $f(b) = 0$ for all inputs $b \neq a$ such that $b \leq a$. *Lowest ones* are lower ones with the smallest number of 1's.

THEOREM 16.5. *Let f be a boolean function, $A \subseteq f^{-1}(1)$ be the set of its lowest ones and ℓ be the number of 1's in them. Then, for every $0 \leq s \leq \ell$, every 1-n.b.p. computing f has size at least*

$$\frac{|A|}{d_s(A) \cdot d_{\ell-s}(A)}.$$

PROOF. Let P be a 1-n.b.p. computing f. For each input $a \in A$, fix an accepting path p_a consistent with a. Since a has ℓ 1-bits, and no vector with a smaller number of 1-bits can be accepted (a is a lowest one), all the ℓ 1-bits of a must be tested along p_a. Split this path into two segments $p_a = (p'_a p''_a)$, where p'_a is an initial segment of p_a along which exactly s 1-bits of a are tested. We denote the corresponding set of bits by I_a, and let J_a denote the set of remaining $\ell - s$ 1-bits of a. For a node v of P, let A_v denote the set of all inputs $a \in A$ such that v is the terminal node of p'_a. We are going to finish the proof by showing that $|A_v| \leq d_s(A)d_{\ell-s}(A)$ for every node v.

Fix some node v of P, and let $\mathscr{I} = \{I_a : a \in A_v\}$, $\mathscr{J} = \{J_b : b \in A_v\}$. Since our program is read-once, we have that $I \cap J = \emptyset$ for all $I \in \mathscr{I}$ and $J \in \mathscr{J}$. Take now an arbitrary pair $I \in \mathscr{I}$, $J \in \mathscr{J}$, and denote by $c_{I,J}$ the input defined by $c_{I,J}(i) = 1$ iff $i \in I \cup J$.

CLAIM 16.6. For every $I \in \mathscr{I}$ and $J \in \mathscr{J}$, the combined input $c_{I,J}$ belongs to A.

PROOF. Choose some $a, b \in A_v$ such that $I = I_a$, $J = J_b$. Since I and J are disjoint, the path $p = (p'_a p''_b)$ is consistent with the input $c_{I,J}$. Hence, this input is accepted because p leads to an accepting sink. But since $|I| + |J| = \ell$ and ℓ is the smallest number of 1's in an accepted input, this is possible only when this combined input $c_{I,J}$ belongs to A. □

With this claim in mind, we fix an arbitrary $J \in \mathscr{J}$ and notice that $\{c_{I,J} \mid I \in \mathscr{I}\}$ is a set of different inputs from A, all of which have 1's on J. Hence, $|\mathscr{I}| \leq d_{|J|}(A) \leq d_{\ell-s}(A)$ (provided $\mathscr{J} \neq \emptyset$). Similarly, $|\mathscr{J}| \leq d_s(A)$ which implies

$$|\mathscr{I}| \cdot |\mathscr{J}| \leq d_s(A)d_{\ell-s}(A).$$

Finally, every $a \in A_v$ is uniquely determined by the pair (I_a, J_a), therefore $|A_v| \leq |\mathscr{I}| \cdot |\mathscr{J}|$. This completes the proof of the desired inequality $|A_v| \leq d_s(A)d_{\ell-s}(A)$, and thus, the proof of the theorem. □

The *exact perfect matching* function is a boolean function EPM_n in n^2 variables, encoding the edges of a bipartite graph with parts of size n; the function computes 1 iff the input graph is a perfect matching. That is, EPM_n takes an $n \times n$ $(0,1)$ matrix as an input, and outputs 1 iff each row and each column has exactly one 1.

COROLLARY 16.7. *Every 1-n.b.p. computing EPM_n must have size $2^{\Omega(n)}$.*

PROOF. Lowest ones for EPM_n are perfect matchings. Hence, we have $n!$ lowest ones. Since, for every $1 \leq s \leq n$, only $(n-s)!$ perfect matchings can share s edges in common, we have that $d_{n/2}(A) \leq (n/2)!$. By Theorem 16.5, any 1-n.b.p. computing EPM_n must have size at least

$$\frac{n!}{(n/2)! \cdot (n/2)!} = \binom{n}{n/2} = 2^{\Omega(n)}.$$ □

What makes the analysis of branching programs difficult is the fact, that they can contain a lot of "redundant" paths, that is, paths containing a contact x_i as well as $\neg x_i$, for some i. These paths (called also "null-chains") are consistent with *no* input vector, but it is known that there presence can exponentially reduce the total size of a program: these paths enable one to merge non-isomorphic subprograms.

Say that a nondeterministic branching program is *weakly read-once* if along any *consistent* s-t path no variable is tested more than once. That is, we now put no restrictions on inconsistent paths.

The following problem is one of the "easiest" questions about branching programs, but it still remains open!

RESEARCH PROBLEM 16.8. *Prove an exponential lower bound for nondeterministic weakly read-once branching program.*

That such programs may be much more powerful than 1-n.b.p.'s shows the following

PROPOSITION 16.9. *The function EPM_n can be computed by a nondeterministic weakly read-once branching program of size $O(n^3)$.*

PROOF. To test that a given square $(0,1)$ matrix is a permutation matrix, it is enough to test whether:

a. every row has at least one 1, and
b. every column has at least $n-1$ 0's.

These two tests can be made by two nondeterministic branching programs P_1 and P_2 designed using the formulas

$$P_1(X) = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} x_{i,j} \quad \text{and} \quad P_2(X) = \bigwedge_{j=1}^{n} \bigvee_{k=1}^{n} \bigwedge_{\substack{i=1 \\ i\neq k}}^{n} \neg x_{i,j}\,.$$

Let $P = P_1 \wedge P_2$ be the AND of these two programs, that is, the sink-node of P_1 is the source-node of P_2. The entire program has size $O(n^3)$. It remains to verify that P is read-once. But this is obvious because all the contacts in P_1 are positive whereas all contacts in P_2 are negative; so every s-t path in the whole program P is either inconsistent or is read-once. □

16.1.2. Parity branching programs. The highest lower bound for parity branching programs remains the Nechiporuk's bound of $\Omega(n^{3/2}/\log)$ shown in Section 14.1 (see Theorem 14.2). Curiously enough, no such lower bound is known even for *read-once* parity branching programs, where along any s-t path (be it consistent or not) every variable appears at most once! This is quite different from deterministic and nondeterministic read-once branching programs were exponential lower bounds are known (we have shown this in previous sections).

RESEARCH PROBLEM 16.10. *Prove an exponential lower bound for read-once parity branching programs.*

So far, exponential lower bounds for such programs are only known under the additional restriction that the program is *oblivious*. The nodes are partitioned into at most n levels so that edges go only from one level to the next, all the edges of one level are labeled by contacts of one and the same variable, and different levels have different

variables. Let us refer to read-once parity branching programs with this restriction as 1-p.b.p.

To prove exponential lower bounds for 1-p.b.p.'s, we will employ one specific property of linear codes – their "universality".

Recall that a linear code is just a set of vectors $C \subseteq \{0,1\}^n$ which forms a linear subspace of GF(2)^n. The *minimal distance* of a code C is a minimal Hamming distance between any pair of distinct vectors in C. It is well known (and easy to show) that minimal distance of C coincides with the minimum weight of (i.e. the number of 1's in) a non-zero vector form C. The *dual* of C is the set $C^\perp$ of all those vectors $x \in \{0,1\}^n$, which are orthogonal to all the vectors from C, i.e., $\sum_{i=1}^n x_i y_i = 0 \bmod 2$ for all $y \in C$.

A set of vectors $C \subseteq \{0,1\}^n$ is *k-universal* if for any subset of k coordinates $I \subseteq \{1,\ldots,n\}$ the projection of vectors from C onto this set I gives the whole cube $\{0,1\}^I$. A nice property of linear codes is that their duals are universal.

PROPOSITION 16.11. *If C is a linear code of minimal distance $k+1$ then its dual $C^\perp$ is k-universal.*

PROOF. Take a set $I \subseteq \{1,\ldots,n\}$ with $|I| \leq k$. The set of all projections of strings in C onto I is a linear subspace in $\{0,1\}^I$, and this subspace is proper if and only if all strings $a \in C$ satisfy a non-trivial linear relation $\sum_i \xi_i a_i = 0 \bmod 2$ whose support $\{i : \xi_i = 1\}$ is contained in I. But, by definition, $C^\perp$ consists exactly of all relations ξ satisfied by C, and its minimal distance is exactly the minimal possible cardinality of a set I for which the projection of C onto $\{0,1\}^I$ is proper. □

A *characteristic function* of a set $C \subseteq \{0,1\}^n$ is a boolean function f_C such that $f_C(x) = 1$ iff $x \in C$.

THEOREM 16.12. *Let $C \subseteq \{0,1\}^n$ be a linear code with minimal distance d_1, and let d_2 be the minimal distance of the dual code $C^\perp$. Then every 1-p.b.p. computing the characteristic function f_C of C has size at least $2^{\min\{d_1,d_2\}-1}$.*

PROOF. Let P be a 1-p.b.p. computing f, $k = \min\{d_1,d_2\} - 1$ and let $I \subseteq \{1,\ldots,n\}$ be the set of bits tested on the first $k = |I|$ levels of P. Every assignment $a : I \to \{0,1\}$ (treated for this purpose as a restriction) defines a subfunction f_a of f in $n - |I|$ variables which is obtained from f by setting x_i to $a(i)$ for all $i \in I$. Let $\mathscr{F}$ be the subspace of the 2^{n-k}-dimensional space of all boolean functions on $n-k$ variables, generated by the subfunctions f_a of f with $a : I \to \{0,1\}$. It is not difficult to see that $size(P) \geq \dim(\mathscr{F})$. Indeed, if $v_1,\ldots,v_r$ are the nodes at the k-th level of P, then for every assignment $a : I \to \{0,1\}$, the subfunction f_a is a linear combination of the functions computed by a 1-p.b.p.'s with source-nodes $v_1,\ldots,v_r$: $f_a(b) = 1$ iff the number of accepting paths in $P(a,b)$ is odd. Hence, we need at least $r \geq \dim(\mathscr{F})$ such functions to get all the subfunctions in $\mathscr{F}$.

Now we can finish the proof as follows. Since the dual of C has distance $d_2 \geq k+1$, we have by Proposition 16.11, that the code C itself is k-universal. This, in particular, means that for every assignment $a : I \to \{0,1\}$ there is an assignment $x_a : \overline{I} \to \{0,1\}$ such that $(a, x_a) \in C$. Moreover, since C has distance $d_1 > k = |I|$, we have that $(b, x_a) \notin C$ for every other assignment $b : I \to \{0,1\}$, $b \neq a$. Thus, if we describe the subfunctions f_a, $a : I \to \{0,1\}$, as rows of a $2^k \times 2^{n-k}$ matrix, then this matrix contains a diagonal $2^k \times 2^k$ submatrix with entries $f(a,x)$ such that $f(a,x) = 1$ iff $x = x_a$. So, the matrix has full row-rank equal 2^k, which means that the subfunctions in $\mathscr{F}$ are linearly independent (over any field, including GF(2)). Thus, $size(P) \geq \dim(\mathscr{F}) = |\mathscr{F}| \geq 2^k$, as desired. □

To give an explicit lower bound, recall that the r-th order binary Reed–Muller code $R(r,t)$ of length $n=2^t$ is the set of graphs of all polynomials in t variables over $\mathbb{F}_2$ of degree at most r. This code is linear and has minimal distance 2^{t-r}.

COROLLARY 16.13. *Let $n=2^t$ and $r=\lfloor t/2\rfloor$. Then every 1-p.b.p. computing the characteristic function of the Reed-Muller code $R(r,t)$ has size at least $2^{\Omega(\sqrt{n})}$.*

PROOF. It is known (see, e.g., [**107**, p. 374]) that the dual of $R(r,t)$ is $R(t-r-1,t)$. Hence, in the notation of Theorem 16.12 we have that $d_1=2^{t-r}\geq\Omega(\sqrt{n})$ and $d_2=2^{r+1}\geq\Omega(\sqrt{n})$. The desired bound follows. □

16.2. Linear codes require large replication

Recall that the *replication number* of a program is the minimal number R such that along every computation path, at most R variables are tested more than once. The sets of variables re-tested along different computations may be different. We will now prove an exponential lower bound for deterministic branching programs with replication number R about $n/\log n$ and even about εn for a constant $\varepsilon>0$. Recall that $R=n$ is the maximal possible value corresponding to unrestricted branching programs.

But before we start, let us first show that testing just one bit twice can help much. For this, let us again consider the pointer function f_n, introduced in Section 16.1.1. We already know (see Theorem 16.4) that any deterministic branching program of replication number $R=0$ (read-once program) for this function must have exponential size. We now show that allowing to re-test just one bit along each path reduces the size drastically.

PROPOSITION 16.14. *The pointer function f_n can be computed by a deterministic branching program of size $O(n^2/\log n)$ and replication number $R=1$.*

PROOF. For each $i=1,\ldots,s$, let P_i be an obvious 1-b.p. of size $s^2=n/k\leq n/\log n$ computing the function $y_i=\bigvee_{j=1}^{s}\left(\bigwedge_{x\in B_{ij}}x\right)$. Arrange these programs into a binary tree of height k. This way we obtain a read-once program of size $O(2^k n/k)=O(n^2/\log n)$. This program has $2^k=n$ leaves, each labeled by the corresponding string $a=(a_1,\ldots,a_k)$, and hence, by the corresponding index $v=\mathrm{bin}(a)$. Replace each such leaf by a size-1 branching program testing the corresponding variable x_v. The resulting program has replication number $R=1$, computes f_n and has the desired size. □

Thus, even when going from programs with $R=0$ to programs with $R=1$, the size may decrease drastically (from exponential to quadratic size).

We are now going to show that some explicit boolean functions require large replications number R, growing with the number n of variables. We will present two entirely different lower bounds arguments for $(1,+R)$-branching programs. The first one, presented in this section, is numerically weaker—works only for $R=o(n/\log n)$—but is (apparently) more instructive. Moreover, it works for important objects—characteristic functions of linear codes. A different argument, presented in the next section, gives exponential lower bounds for programs of almost maximal replication $R=\Omega(n)$, but the functions for which it works are no more as "simple"—they are quadratic functions of good expander graphs.

A *partial input* is a mapping $a:[n]\to\{0,1,*\}$ where $[n]=\{1,\ldots,n\}$. If $a(i)=*$ we say that the ith bit in a is *unspecified* (or undefined). The *support* $S(a)$ of a is the set of all specified bits, that is, bits i for which $a(i)\neq *$. A composition $b=a_1a_2\cdots a_s$ of (partial) inputs $a_1,a_2,\ldots,a_s$, whose supports are pairwise disjoint, is a (partial) input

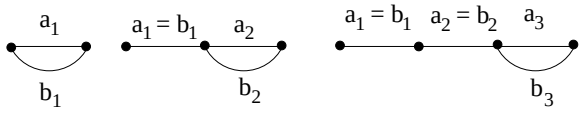


FIGURE 2. Forgetting pairs a_1 and b_1, a_1b_2 and a_1b_2, $a_1a_2a_3$ and $a_1a_2b_3$.

defined by $b(i) = a_j(i)$ for $i \in S(a_j)$. The *length* $|a|$ of a is the number of bits in $S(a)$. For two partial inputs a and b, let $D(a,b)$ be the set of all bits where they both are defined and have different values.

Given a boolean function $f(x_1,\ldots,x_n)$, every partial input a (treated for this purpose as a restriction) defines the subfunction $f\restriction_a$ of f in $n-|a|$ variables in a usual manner. A partial input a is a 0-*term* of f if $f\restriction_a \equiv 0$, and 1-*term* if $f\restriction_a \equiv 1$. We say that:

- f is d-*rare* if $|D(a,b)| \geq d$ for every two different totally defined inputs a, b such that $f(a) = f(b) = 1$;
- f is m-*dense* if $|a| \geq m$ for every 0-term a of f.

That is, f is d-rare if the Hamming distance between any two vectors in $f^{-1}(1)$ is at least d, and is m-dense, if it is not possible to make the function be constant 0 by fixing fever that m variables.

THEOREM 16.15. *Let $0 \leq d, m, R \leq n$ be arbitrary integers. Every $(1,+R)$-branching program computing a d-rare and m-dense function must have size at least*

$$2^{(\min\{d,\ m/(R+1)\}-1)/2}.$$

The idea behind the proof of this fact is the following. If all computations are long (of length at least m) and the program is not too large, a lot of computation paths must split and join again. At that node were they join again, some information about the inputs leading to this node is lost. If too much information is lost and not too many (at most R) variables may be re-tested once again, it is not possible to compute the correct value of the function.

The intuition about the "loss of information" is captured by the following notion of "forgetting pairs" of inputs.

DEFINITION 16.16. Let a, b be (partial) inputs with $S(a) = S(b)$. Given a branching program P, the pair a, b is called a *forgetting pair* (for P) if there exists a node w such that w belongs to both $comp(a)$ and $comp(b)$, and both computations read all the variables with indices in $D(a,b)$ at least once before reaching w.

LEMMA 16.17. *Let P be a branching program in which every computation reads at least m different variables. Let s be a natural number in the interval*

$$1 \leq s \leq \frac{m}{2\log_2|P|+1}.$$

Then there exist pairwise disjoint subsets $I_1,\ldots,I_s$ of $[n]$ and partial inputs $a_j \neq b_j$ with $S(a_j) = S(b_j) = I_j$ such that for all $j = 1,2,\ldots,s$ we have:

(i) $|I_j| \leq 2\log_2|P|+1$,

(ii) *the inputs $a_1\cdots a_{j-1}a_j$ and $a_1\cdots a_{j-1}b_j$ form a forgetting pair.*

PROOF. Given a b.p. P, one can get a forgetting pair by following all the computations until $r := \lfloor \log_2|P| \rfloor + 1$ different bits are tested along each of them. Since

$|P| < 2^r$, at least two of these paths must first split and then stick in some node. Take the corresponding partial inputs a'_1 and b'_1 and extend them to a_1 and b_1 such that $S(a_1) = S(b_1) = S(a'_1) \cup S(b'_1)$ and $D(a_1, b_1) \subseteq S(a'_1) \cap S(b'_1)$. This way we get a forgetting pair of inputs $a_1 \neq b_1$ both of which are defined on the same set of at most $|S(a'_1) \cup S(b'_1)| \leq 2r-1$ bits. We can now repeat the argument for $P|_{a_1}$ and obtain next forgetting pair of inputs a_1a_2 and a_1b_2, etc. We can continue this procedure for s steps until $s(2r-1) \leq s(2\log_2|P|+1)$ does not exceed the minimum number m of different variables tested on a computation of P. □

PROOF OF THEOREM 16.15. Suppose the contrary, that some $(1,+R)$-b.p. P computes a d-rare and m-dense function and has size less than $2^{(\min\{d,\ m/(R+1)\}-1)/2}$. We can assume w.l.o.g. that $d \geq 2$ (otherwise the bound becomes trivial), and this implies that every 1-term of f has size $n \geq m$. Hence, in order to force f to either 0 or 1 we must specify at least m positions, therefore every computation of P must read at least m different variables. Since

$$|P| \leq 2^{(m/(R+1)-1)/2},$$

we can apply Lemma 16.17 (with $s := R+1$) and find $R+1$ sets I_i and partial inputs $a_i, b_i : [n] \to \{0,1,*\}$ with properties (i) and (ii). From (i) and the bound on $|P|$ we have $|I_j| < \min\{d,\ m/(R+1)\}$, and this implies that the partial input $a_1 \cdots a_{R+1}$ specifies strictly less than m variables. Since f is m-dense, $a_1 \cdots a_{R+1}$ can be extended to a totally defined input a such that $f(a) = 1$.

As the sets $I_1, \ldots, I_{R+1}$ are pairwise disjoint and at most R variables can be re-tested along any computation, there must exist j such that all variables with indices from I_j are tested *at most once* along $comp(a)$. Now, let w be the node that corresponds to the forgetting pair

$$a_1 \cdots a_{j-1}a_j \quad \text{and} \quad a_1 \cdots a_{j-1}b_j\,;$$

w is on $comp(a)$. All variables with indices from $D(a_j, b_j) \subseteq I_j$ are already tested along $comp(a)$ before w, hence no such variable is tested after w, and the computation on the input c obtained from a by replacing a_j with b_j can not diverge from $comp(a)$ after the node w. Therefore, $f(c) = f(a) = 1$. But this, along with $|I_j| < d$, contradicts the d-rareness of f, and the proof of Theorem 16.15 is completed. □

This theorem is especially useful for (characteristic functions of) linear codes, that is, for linear subspaces of $GF(2)^n$.

OBSERVATION 16.18. The characteristic function of a linear code C is d-rare if and only if the minimal distance of C is at least d, and is m-dense if and only if the minimal distance of its dual $C^\perp$ is at least m.

PROOF. The first claim is obvious, the second follows from Proposition 16.11. □

Hence, Theorem 16.15 implies:

THEOREM 16.19. *Let C be a linear code with minimal distance d_1, and let d_2 be the minimal distance of the dual code $C^\perp$. Then every $(1,+R)$-branching program computing the characteristic function of C has size exponential in* $\min\{d_1,\ d_2/R\}$.

This theorem yields exponential lower bounds on the size of $(1,+R)$-branching programs computing characteristic functions of many linear codes. The largest allowed replication number $R = O(n/\log n)$ is achieved by Bose–Chaudhury-Hocquenghem codes, known as BCH-codes.

Let $n = 2^\ell - 1$, and let $C \subseteq \{0,1\}^n$ be a BCH-code with designed distance $\delta = 2t+1$, where $t \leq \sqrt{n}/4$, and let f_C be its characteristic function. Let d_2 be the minimal distance of its dual $C^\perp$. The Carliz–Uchiyama bound (see, e.g., [**107**, p. 280]) says that $d_2 \geq 2^{\ell-1} - (t-1)2^{\ell/2}$ which is $\Omega(n)$ due to our assumption on t. Since the minimal distance d_1 of a BCH-code is always at least its designed distance δ, we get from Theorem 16.19

COROLLARY 16.20. *Every $(1,+R)$-branching program computing f_C has size exponential in* $\min\{t,\ n/R\}$.

In particular, if $t = \omega(\log n)$ then every such program must have super-polynomial size as long as $R = o(n/\log n)$.

16.3. Replication of rectangle-free functions

We are now going to prove exponential lower bound on the size of branching programs with almost maximal replication number $R = \Omega(n)$. The functions for which we prove such a bound will have the form

$$f_n(x_1 \ldots, x_n) = (x_1 \oplus \cdots \oplus x_n \oplus 1) \wedge \left(\bigvee_{\{i,j\} \in E} x_i \wedge x_j \right),$$

where E is the set of edges of a specially chosen graph $G = ([n], E)$, so called, Ramanujan graph. That is, given an input vector $a \in \{0,1\}^n$, we remove all vertices i with $a_i = 0$, and let $f_n(a) = 1$ iff the number of 1's in a is even and the number of survived edges is odd.

It is clear that f_n can be computed by an unrestricted ($R = n$) branching program of size $O(n^2)$. We will show that good expanding properties of the graph G imply that every branching program computing f_n with replication number $R = o(n)$ must already have exponential size.

But first we will prove a general theorem telling us what properties boolean function force the replication number of their branching programs be large.

A boolean function $r(x_1, \ldots, x_n)$ is a *rectangular function* if there is a balanced partition of its variables into two parts such that r can be written as an AND of two boolean functions, each depending on variables in only one part of the partition. A set $R \subseteq \{0,1\}^n$ of vectors is a *combinatorial rectangle* (or just a *rectangle*) if $R = r^{-1}(1)$ for some rectangular function r. So, each combinatorial rectangle has a form $R = R_0 \times R_1$ where $R_0 \subseteq \{0,1\}^{I_0}$ and $R_1 \subseteq \{0,1\}^{I_1}$ for some partition $[n] = I_0 \cup I_1$ of $[n] = \{1, \ldots, n\}$ into two disjoint parts I_0 and I_1 whose sizes differ by at most 1.

The *rectangle number*, $\varrho(f)$, of a boolean function f is the maximum size $|R|$ of a rectangle R such that $f(a) = 1$ for all $a \in R$. Finally, we say that a boolean function f in n variables is:

a. *sensitive* if any two accepted vectors differ in at least two bits;
b. *dense* if $|f^{-1}(1)| \geq 2^{n-o(n)}$, and
c. *rectangle-free* if $\varrho(f) \leq 2^{n-\Omega(n)}$.

THEOREM 16.21. *There is a constant $\varepsilon > 0$ with the following property: If f is a sensitive, dense and rectangle-free boolean function in n variables, than any deterministic branching program computing f with the replication number $R \leq \varepsilon n$ must have size $S = 2^{\Omega(n)}$.*

PROOF. Let f be a sensitive and dense boolean function in n variables. Suppose also that the function f is rectangle-free, that is, $f^{-1}(1)$ does not contain a rectangle of size larger than $2^{n-\delta n}$, for some constant $\delta > 0$. Take an arbitrary deterministic branching program computing f with replication number $R \le \varepsilon n$, where $\varepsilon > 0$ is a sufficiently small constant to be specified later; this constant will only depend on the constant δ. Our goal is to prove that then the program must have at least $2^{\Omega(n)}$ nodes.

For an input $a \in \{0,1\}^n$ accepted by f, let $comp(a)$ denote the (accepting) computation path on a. Since the function f is sensitive, all n bits are tested at least once along each of these paths. Split each of the paths $comp(a)$ into two parts $comp(a) = (p_a, q_a)$, where p_a is an initial segment of $comp(a)$ along which $n/2$ different bits are tested. Hence, the remaining part q_a can test at most $n/2+R$ different bits.[1] Looking at segments p_a and q_a as monomials (ANDs of literals), we obtain that f can be written as an OR of ANDs $P \wedge Q$ of two DNFs satisfying the following three conditions:

(i) All monomials have length at least $n/2$ and at most $n/2+R$. This holds by the choice of segments p_a and q_a.

(ii) Any two monomials in each DNF are inconsistent, that is, one contains a variable and the other contains its negation. This holds because the program is deterministic: the paths must split before they meet.

(iii) For all monomials $p \in P$ and $q \in Q$, either $pq = 0$ (the monomials are inconsistent) or $|X(p) \cap X(q)| \le R$ and $|X(p) \cup X(q)| = n$, where $X(p)$ is the set of variables in a monomial p. This holds because the program has replication number R and the function f is sensitive.

Fix now one AND $P \wedge Q$ for which the set B of accepted vectors is the largest one; hence, the program must have at least $|f^{-1}(1)|/|B| \ge 2^{n-o(n)}/|B|$ nodes, and it remains to show that the set B cannot be too large, namely, that

$$|B| \le 2^{n-\Omega(n)}.$$

We do this by showing that otherwise the set B, and hence, also the set $f^{-1}(1)$, would contain a large rectangle in contradiction with the rectangle-freeness of f. When doing this we only use the fact that all vectors of B must be accepted by an AND of DNFs satisfying the properties (i)-(iii) above.

By (iii) we know that every vector $a \in B$ must be accepted by some pair of monomials $p \in P$ and $q \in Q$ such that $|X(p) \cap X(q)| \le R$. A (potential) problem, however, is that for different vectors a the corresponding monomials p and q may share *different* variables in common. This may prohibit their combination into a rectangle (see Remark 16.23 below). To get rid of this problem, we just fix a set Y of $|Y| \le R$ variables for which the set $A \subseteq B$ of all vectors in B accepted by pairs of monomials with $X(p) \cap X(q) = Y$ is the largest one. Hence,

$$|A| \ge |B| \Big/ \sum_{i=0}^{R} \binom{n}{i} \ge |B| \cdot 2^{-n \cdot H(\varepsilon)},$$

where $H(x) = -x \log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function.

CLAIM 16.22. The set A contains a rectangle C of size

$$|C| \ge \frac{|A|^2}{9 \cdot 2^{n+R}}.$$

[1] Note that we only count the number of tests of *different* bits—the total length of (the number of tests along) $comp(a)$ may be much larger than $n+R$.

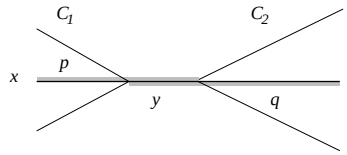


FIGURE 3. $C = C_1 \times \{y\} \times C_2$ forms a rectangle.

Assuming the claim, we can finish the proof of the theorem as follows. By the rectangle-freeness of f, we know that $|C| \le 2^{n-\delta n}$ for a constant $\delta > 0$. By Claim 16.22, we know that

$$|A| \le 3 \cdot 2^{(n+R)/2}|C| \le 3 \cdot 2^{(1+\varepsilon)n/2+(1-\delta)n}.$$

Hence, if $R \le \varepsilon n$ for a constant $\varepsilon > 0$ satisfying $\varepsilon + 2H(\varepsilon) < 2\delta$, then

$$|B| \le |A| \cdot 2^{H(\varepsilon)n} \le 3 \cdot 2^{n-(2\delta-\varepsilon-2H(\varepsilon))n/2} \le 2^{n-\Omega(n)}.$$

It remains therefore to prove Claim 16.22.

Each monomial of length at most k accepts at least a 2^{-k} fraction of all vectors from $\{0,1\}^n$. Hence, there can be at most 2^k mutually inconsistent monomials of length at most k. By (i) and (ii), this implies that

$$|P| \le 2^{n/2} \text{ and } |Q| \le 2^{n/2+R}. \tag{16.2}$$

For each monomial $p \in P \cup Q$, let $A_p = \{a \in A \mid p(a) = 1\}$ be the set of all vectors in A accepted by p; we call these vectors *extensions* of p. Note that, by the definition of the set A, $a \in A_p$ iff $pq(a) = 1$ for some monomial $q \in Q$ such that $X(p) \cap X(q) = Y$.

Since, by (ii), the monomials in P are mutually inconsistent, no two of them can have a common extension. Since every vector from A is an extension of at least one monomial $p \in P$, the sets A_p with $p \in P$ form a partition of A into $|P|$ disjoint blocks. The average size of a block in this partition is $|A|/|P|$. Say that a monomial $p \in P$ is *rich* if the corresponding block A_p contains $|A_p| \ge \frac{1}{3}|A|/|P|$ vectors. Similarly for monomials in Q. By averaging, at least two-thirds of vectors in A must be extensions of rich monomials in P. Since the same holds also for monomials in Q, at least one vector $x \in A$ must be an extension of some rich monomial $p \in P$ and, at the same time, of some rich monomial $q \in Q$.

Let y be the projection of x onto $Y = X(p) \cap X(q)$. Since all variables in Y are tested in both monomials p and q, all the vectors in A_p and in A_q coincide with y on Y. Consider the set of vectors $C = C_1 \times \{y\} \times C_2$, where C_1 is the set of projections of vectors in A_q onto the set of variables $X - X(q)$, and C_2 is the set of projections of A_p onto the set of variables $X - X(p)$ (see Fig. 3). Since both monomials p and q have at least $n/2$ variables, the set C is a rectangle of size

$$|C| = |C_1| \cdot |C_2| = |A_p| \cdot |A_q| \ge \frac{|A|}{3|P|} \cdot \frac{|A|}{3|Q|} \ge \frac{1}{9}\frac{|A|}{2^{n/2}} \cdot \frac{|A|}{2^{n/2+R}} = \frac{1}{9}\frac{|A|^2}{2^{n+R}}.$$

Hence, it remains to verify that $C \subseteq A$, i. e., that all vectors $c \in C$ are accepted by $P \wedge Q$.

The vector x belongs to C and has the form $x = (x_1, y, x_2)$ with $x_i \in C_i$. Take now an arbitrary vector $c = (c_1, y, c_2)$ in C. The vector (x_1, y, c_2) belongs to A_p. Hence, there must be a monomial $q' \in Q$ such that $X(p) \cap X(q') = Y$ and pq' accepts this vector. Since all bits of x_1 are tested in p and none of them belongs to Y, none of these bits is tested in q'. Hence, q' must accept also the vector $c = (c_1, y, c_2)$. Similarly, using

the fact that (c_1, y, x_2) belongs to A_q, we can conclude that the vector $c = (c_1, y, c_2)$ is accepted by some monomial $p' \in P$. Thus, the vector c is accepted by the monomial $p'q'$, and hence, by $P \wedge Q$.

This completes the proof of the proof of Claim 16.22, and thus, the proof of Theorem 16.28. □

REMARK 16.23. Note that in the last step of the proof it was important that every vector from A is accepted by a pair of monomials sharing the *same* set of variables Y. Would A not have this property, then the rectangle C would not necessarily lie within the set A. Take for example $P = \{x_1, \neg x_1\}$ and $Q = \{x_2, x_1 \neg x_2\}$ with $p = x_1$ and $q = x_2$. The AND $P \wedge Q$ accepts the set of vectors $A = \{11, 01, 10\}$. The projection of $A_q = \{11, 01\}$ onto $X - X(q) = \{x_1\}$ is $C_1 = \{0, 1\}$, and the projection of $A_p = \{11, 10\}$ onto $X - X(p) = \{x_2\}$ is also $C_2 = \{0, 1\}$. But $C = C_1 \times C_2 \not\subseteq A$, because 00 does not belong to A.

Important in our proof was also that the branching program is *deterministic*: this resulted in the property (ii) in the proof of Theorem 16.21, and hence, into upper bounds (16.2) on the number of monomials. In the case of nondeterministic branching programs we do not necessarily have this property, and in this case no exponential lower bounds are known even for $R = 1$ (cf. Problem 16.8).

16.3.1. Graph expansion implies rectangle-freeness. To apply Theorem 16.21 we need an explicit boolean function that is sensitive, dense and rectangle-free. Note that the first two conditions—being sensitive and dense—are easy to ensure. A difficult thing is to ensure rectangle-freeness. The problem here is that f must be rectangle-free under *any* balanced partition of its variables. We define such functions using graphs.

Let $G = (V, E)$ be an undirected graph on $V = \{1, \ldots, n\}$. The quadratic function of G over $GF(2)$ is a boolean function

$$f_G(x_1, \ldots, x_n) = \sum_{\{i,j\} \in E} x_i x_j \bmod 2 .$$

That is, given an input vector $a \in \{0, 1\}^n$, we remove all vertices i with $a_i = 0$, and count the number of the surviving edges modulo 2.

16.3.1.1. *Density.* That quadratic functions f_G accept many vectors follows from the a more general fact about polynomials.

LEMMA 16.24. *Every nonzero polynomial of degree k in n variables over $GF(2)$ has at least 2^{n-k} nonzero points.*

PROOF. In each such polynomial $f(x_1, \ldots, x_n)$ we can find a monomial $X_I = \prod_{i \in I} x_i$ with $|I| = k$ which is *maximal* in a sense that no monomial $X_{I'}$ with $I' \supset I$ is present in f. Hence, after each of 2^{n-k} assignments a of constants to variables x_j with $j \notin I$, we obtain a polynomial f_a in k variables $\{x_i \mid i \in I\}$ whose all monomials, other than X_I, have degree strictly less than k. Our goal is to show that then $f_a(b) = 1$ for at least one $b \in \{0, 1\}^I$. The function f_a has a form $f_a = X_I \oplus g$, where g is a polynomial of degree $d < k$ in k variables.

If g has no monomials at all, i.e., is a constant polynomial $g \equiv c$ for $c \in \{0, 1\}$, then $f_a(b) = 1$ for $b = (c \oplus 1, \ldots, c \oplus 1)$.

If g is a non-constant polynomial, then take one its monomial X_J which is *minimal* in a sense that no monomial $X_{J'}$ with $J' \subset J$ is present in g. Let $c \in \{0, 1\}$ be the free coefficient of g. If $c = 1$, then $f_a(0, \ldots, 0) = c = 1$, and we are done. Otherwise (if

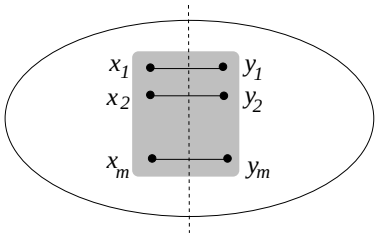


FIGURE 4. After the setting to 0 all variables outside the induced matching, the function $f_G = \bigoplus_{\{i,j\}\in E} x_i y_j$ turns to the inner product function $IP_{2m} = x_1y_1 \oplus \cdots \oplus x_m y_m$.

$c = 0$) take the vector $b \in \{0,1\}^I$ with $b_i = 1$ for all $i \in J$, and $b_i = 0$ for all $i \in I - J$. Then $g(b) = X_J(b) = 1$ due to the minimality of J, and $X_I(b) = 0$ since $I - J \neq \emptyset$. Hence, we again have that $f_a(b) = 1$. □

16.3.1.2. *Matching number.* The next question is: What properties of a graph G do ensure that its quadratic function f_G is rectangle-free? We will now show that such is the *matching number* $\mathrm{m}(G)$ of the underlying graph. The measure $\mathrm{m}(G)$ is defined as the largest number m such that, for every balanced partition of vertices of G, at least m crossing edges form an induced[2] matching; and edge is crossing if it joins a vertex in one part of the partition with a vertex in the other part.

The fact that such a matching must be *induced matching* means that the endpoints of any two of its edges are not adjacent in G. This last property is important: if $M = \{x_1y_1, \ldots, x_my_m\}$ is an induced matching of G, then we can set to 0 all variables of f_G outside M, and what we obtain is the inner product function $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_my_m$. Then we can use the fact that the rectangle number of the inner product function is small.

LEMMA 16.25. *For every graph G on n vertices, we have*

$$\varrho(f_G) \leq 2^{n-\mathrm{m}(G)}.$$

PROOF. Fix an arbitrary balanced partition of the vertices of G into two parts. The partition corresponds to a partition (x, y) of the variables of f_G. Let $r = r_1(x) \wedge r_2(y)$ be an arbitrary rectangle function with respect to this partition, and suppose that $r \leq f$. Our goal is to show that then r can accept at most $2^{n-\mathrm{m}(G)}$ vectors.

By the definition of $\mathrm{m}(G)$, some set $M = \{x_1y_1, \ldots, x_my_m\}$ of $m = \mathrm{m}(G)$ crossing edges x_iy_i forms an induced matching of G. We set to 0 all variables corresponding to vertices outside the matching M (see Fig. 4). Since M is an *induced* subgraph of G, the obtained subfunction of f_G is just the inner product function

$$IP_{2m}(x_1, \ldots, x_m, y_1, \ldots, y_m) = \sum_{i=1}^{m} x_i y_i \bmod 2.$$

The obtained subfunction $r' = r'_1(x_1, \ldots, x_m) \wedge r'_2(y_1, \ldots, y_m)$ of the rectangle function $r = r_1 \wedge r_2$ is also a rectangle function such that $r'(a) \leq IP_{2m}(a)$ for all $a \in \{0,1\}^{2m}$.

[2]An *induced subgraph* of a graph is obtained by removing vertices together with their adjacent edges.

Since r' was obtained from r by setting to 0 at most $n-2m$ variables, we have that $|r^{-1}(1)| \leq |B| \cdot 2^{n-2m}$ where $B = \{a \mid r'(a) = 1\}$. Hence, it remains to show that $|B| \leq 2^m$. For this, let H be a $2^m \times 2^m$ matrix defined by

$$H[x,y] = (-1)^{IP_{2m}(x,y)\oplus 1}.$$

Since, for every $x \neq 0$, $IP_{2m}(x,y) = 1$ for exactly half of vectors y, this matrix is a Hadamard matrix. The following property of Hadamard matrices is a special case of a more general Lindsey's Lemma (Lemma 10.25).

CLAIM 16.26. An $n \times n$ Hadamard matrix H can contain an $a \times b$ all-1 submatrix only if $ab \leq n$.

PROOF. Take an $a \times b$ all-1 submatrix, and let $v = v_1 + \cdots + v_a$ be the sum of the corresponding rows of H. Since this is an all-1 submatrix, the vector v must contain at least b entries equal to a, implying that $\|v\|^2 \geq a^2 b$. On the other hand, since the rows of H are pairwise orthogonal, we have that

$$\|v\|^2 = \sum_{i=1}^{a} \langle v_i, v_j \rangle = \sum_{i=1}^{a} \langle v_i, v_i \rangle = an.$$

Altogether this yields $ab \leq n$. □

Since our set $B \subseteq \{0,1\}^m \times \{0,1\}^m$ lies within $IP_{2m}^{-1}(1)$, it corresponds to an all-1 submatrix of H. Claim 16.26 implies that $|B| \leq 2^m$, as desired. □

16.3.1.3. *Mixed graphs.* By Lemma 16.25, we need graphs G such that, for any balanced partition of their vertices, many crossing edges form an induced matching. To ensure this, it is enough that the graph is "mixed enough".

Say that a graph is *s-mixed* if every two disjoint sets of at least s vertices are joined by at least one edge.

LEMMA 16.27. *If an n-vertex graph G of maximum degree d is s-mixed, then*

$$\mathrm{m}(G) \geq \frac{n-2s}{4(d+1)},$$

and hence,

$$\log_2 \varrho(f_G) \leq n - \frac{n}{4(d+1)} + \frac{s}{2(d+1)}.$$

PROOF. Fix an arbitrary balanced partition of the vertices of G into two parts. To construct the desired induced matching, formed by crossing edges, we repeatedly take a crossing edge and remove it together with all its neighbors. In each step we remove at most $2d+1$ vertices. If the graph is s-mixed, then the procedure will run for m steps as long as $\lfloor n/2 \rfloor - (2d+1)m$ is at least s. □

By Corollary 9.20 in Section 9.7, Ramanujan graphs $G = RG(n,q)$ are are δn-mixed for a constant $\delta < 1/2$, as long as $q \geq 2^6$. Since the degree of $G = RG(n,q)$ is $q+1$ (a constant, if q is constant), Lemma 16.27 implies

$$\varrho(f_G) \leq 2^{n-\Omega(n)} \tag{16.3}$$

Fix now a Ramanujan graphs $G = RG(n,q)$ with $q \geq 2^6$, and consider the boolean function

$$f_n = f_G \wedge (x_1 \oplus \cdots \oplus x_n \oplus 1).$$

That is, given an input vector $a \in \{0,1\}^n$, we remove all vertices i with $a_i = 0$, and let $f_n(a) = 1$ iff the number of 1's in a is even and the number of survived edges is odd.

THEOREM 16.28. *There is a constant $\varepsilon > 0$ such that any deterministic branching program computing f_n with the replication number $R \leq \varepsilon n$ requires size $2^{\Omega(n)}$.*

PROOF. By (16.3), the function f_G, and hence, also the function f_n is rectangle-free. Since the parity function is sensitive, the function f_n is sensitive as well. Finally, since f_n is a polynomial of degree at most 3 over $GF(2)$, Lemma 16.24 implies that f_n accepts at least 2^{n-3} vectors, and hence, is also dense, and Theorem 16.21 yields the desired lower bound. □

Bibliographic Notes

Lemma 16.17 is due to Zák (1995) and Savický and Zák (1997). Theorem 16.15 was proved in [**85**]. Theorems 16.21 and 16.28 are from [**82**].

CHAPTER 17

Bounded Time

We consider functions $f : D^n \to \{0,1\}$, where D is a finite domain, not necessarily $\{0,1\}$. We can extend the notion of nondeterministic (as well as deterministic) branching programs also for this case. In the case of boolean n.b.p. (when $D = \{0,1\}$) at each edge some test of the form "$x_i = \sigma$" with $\sigma \in \{0,1\}$ is made. By a *D-way* branching program we will mean a branching program where the tests of the form "$x_i = d$" with $d \in D$ are made. Different edges leaving the same node may make the same test—this is why a program is nondeterministic. Such a program accepts an input vector $a \in D^n$ if and only if all the tests along at least one s-t path are passed.

17.1. Short time forces large rectangles

We say that a program computes a given function f in *time* T if for every input $a \in f^{-1}(1)$ there is a path from the source to a 1-sink which is consistent with a and along which at most T tests are made. Important here is that the restriction concerns only *consistent* paths, that is, paths along which no two tests $x_i = d_1$ and $x_i = d_2$ for $d_1 \neq d_2$ are made. This makes the lower bounds problem more difficult. The "syntactic" case, where the restriction is on *all* paths, be they consistent or not, is much easier.

We now consider nondeterministic D-way branching programs working in time kn where k is an arbitrary large constant. We want to show that some explicit functions $f : D^n \to \{0,1\}$ cannot be computed by such programs using polynomial number of nodes. The idea is to show that, if the number of nodes is small then the program must accept all vectors of a large configuration, called "broom". Having shown this, we construct a function f that cannot accept many vectors of any broom. This will imply that any program for f working in time kn must have large size.

Let $X = \{x_1, \ldots, x_n\}$ be a set of n variables. A subset $R \subseteq \{0,1\}^n$ of vectors is an *s-broom*, if there exist two disjoint s-element subsets X_0 and X_1 of X, subsets $R_0 \subseteq D^{X_0}$ and $R_1 \subseteq D^{X_1}$ of vectors, and a vector $w \in D^{X-(X_0 \cup X_1)}$ such that (after some permutation of the variables) the set R can be written as $R_0 \times \{w\} \times R_1$; the vector w is then the *stick* of the broom. That is, on the variables outside $X_0 \cup X_1$ all vectors in R have the same values as the vector w. With some abuse of notation we will write $R = R_0 \times \{w\} \times R_1$, meaning that this holds after the corresponding permutation of variables.

Note that, for $s \leq n/2$, each s-broom R is also a combinatorial rectangle, as defined in the previous chapter. Such a rectangle has, however, special form: all vectors in R have the same values in each of $n - 2m$ positions, corresponding to the stick w of the broom. This, in particular, implies that no s-broom can have more than $|D|^{2s}$ vectors.

The main property of s-brooms (as well as of rectangles, we have considered before) is the "cut-and-paste" property: if the broom R contains two vectors (a_0, w, a_1) and (b_0, w, b_1), then it must contain both vectors (b_0, w, a_1) and (a_0, w, b_1).

A function $f : D^n \to \{0,1\}$ is *sensitive* if any two accepted vectors differ in at least two coordinates. The only property of such functions we will use is that in any branching program computing such a function, along any accepting computation each variable must be tested at least once.

LEMMA 17.1. *Let $f : D^n \to \{0,1\}$ be a sensitive function and suppose that f can be computed by a nondeterministic branching program of size ℓ working in time kn. Let $r = 10k^2$ and $K = k^{5k}$. Then for every $s \le n/K$, there exists an s-broom $R \subseteq f^{-1}(1)$ of size*

$$|R| \ge \frac{|f^{-1}(1)| \cdot |D|^{2s-n}}{\ell^r \binom{n}{s}^2}. \tag{17.1}$$

In the proof we will use the following simple combinatorial fact. Say that a sequence $S_1, \ldots, S_r$ of finite subsets of an n-element set X is *s-separated* if there exist two disjoint subsets X_0 and X_1 of X, each of size at least $n/2N$ and such that, for each $i = 1, \ldots, r$, either $S_i \cap X_0 = \emptyset$ or $S_i \cap X_1 = \emptyset$ (or both) hold.

CLAIM 17.2. Let $r > 8k^2$ and $N := \sum_{i=0}^{2k} \binom{r}{i}$. If $|S_i| \le kn/r$ for all $i = 1, \ldots, r$ then the sequence $S_1, \ldots, S_r$ of sets is s-separated with $s \ge n/2N$.

PROOF OF CLAIM 17.2. Associate with each element $x \in X$, its *trace*

$$T(x) = \{i \mid x \in S_i\}.$$

By double-counting,

$$\sum_{x \in X} |T(x)| = \sum_{i=1}^{r} |S_i| \le kn. \tag{17.2}$$

We will concentrate on elements whose traces are not too large. Namely, say that an element $x \in X$ is *legal* if $|T(x)| \le 2k$. It is clear that we must have at least $n/2$ legal elements, for otherwise the first sum in (17.2) would be larger than $(2k)(n/2) = kn$.

Partite the legal elements into blocks, where two elements x and y belong to the same block iff they have the same trace, that is, iff $T(x) = T(y)$. Since $|T(x)| \le 2k$, each block in this partition is determined by a subset of $\{1, \ldots, r\}$ of size at most $2k$. So, the total number of blocks does not exceed $\sum_{i=0}^{2k} \binom{r}{i} = N$.

Say that a legal element $x \in X$ is *happy* if the (unique) block, which it belongs to, has at least $n/2N$ elements. If we will find two legal elements $x \ne y \in X$ such that both of them are happy and $T(x) \cap T(y) = \emptyset$, then we are done.

First observe that, by the same averaging argument as above, at least half of all $n/2$ legal elements must be happy (belong to large blocks); hence, at least $n/4$ elements are both legal and happy. Fix any such element x. We have only to show that there is yet another legal and happy element y which belongs to none of the $|T(x)| \le 2k$ sets S_i containing x. For this it is enough to observe that the total number of elements that belong to some of the sets S_i containing x is

$$\left| \bigcup_{i \in T(x)} S_i \right| \le \sum_{i \in T(x)} |S_i| \le |T(x)| \cdot \max |S_i| \le \frac{2k^2 n}{r}$$

which, due to our assumption $r > 8k^2$, is strictly smaller than the total number $n/4$ of legal and happy elements. □

PROOF OF LEMMA 17.1. For each input $a \in f^{-1}(1)$, fix one accepting computation path $comp(a)$, and split it into r sub-paths $p_1, \ldots, p_r$ of length at most kn/r; the length of a sub-path p_i is the number of tests made along it. That is, we have r *time segments* $1, \ldots, r$, and in the ith of them the computation on a follows the sub-path p_i.

Say that two inputs a and b in $f^{-1}(1)$ are equivalent if the starting nodes of the corresponding sub-paths $comp(a) = (p_1, \ldots, p_r)$ and $comp(b) = (q_1, \ldots, q_r)$ coincide. Since we have at most s nodes in the program, the number of possible equivalence classes does not exceed ℓ^r, where ℓ is the total number of nodes in our branching program. Fix some largest equivalence class $A \subseteq f^{-1}(1)$; hence,

$$|A| \geq |f^{-1}(1)|/\ell^r \,.$$

Call a pair of disjoint subsets of variables X_0 and X_1 *good* for a vector $a \in A$ if along the computation $comp(a) = (p_1, \ldots, p_r)$ no subpath p_i tests variables from both sets X_0 and X_1. That is, if some variable from one set is tested along p_i, then none of the variables from the other set is tested along p_i.

By letting S_i be the set of variables tested along the ith time segment p_i, Claim 17.2 implies that every every vector $a \in f^{-1}(1)$ has at least one good pair X_0, X_1 with $|X_0|, |X_1| \geq n/K$, where $K = 2N = 2\sum_{i=0}^{2k} \binom{r}{i} \leq k^{5k}$. Since we have at most $\binom{n}{s}^2$ pairs of disjoint s-element subsets of variables, some of these pairs X_0, X_1 must be good for all vectors in a subset $B \subseteq A$ of size[1] $|B| \geq |A|\binom{n}{s}^{-2}$. To finish the proof of Lemma 17.1, we will now show that this set B forces a large broom lying entirely in A, and hence, in $f^{-1}(1)$.

We can write each vector $a \in D^n$ as $a = (a_0, w, a_1)$, where a_0 is the projection of a onto X_0, a_1 is the projection of a onto X_1, and w is the projection of a onto $X - (X_0 \cup X_1)$. Say that two vectors $a = (a_0, w, a_1)$ and $b = (b_0, w', b_1)$ are equivalent if $w = w'$. Since the sets of variables X_0 and X_1 are disjoint, each equivalence class is an m-broom.

Let $R \subseteq B$ be a largest equivalence class lying in B; hence

$$|R| \quad \geq \quad \frac{|B|}{|D|^{n-2s}} \geq \frac{|A|}{\binom{n}{s}^2 |D|^{n-2s}} \geq \frac{|f^{-1}(1)|}{\ell^r \binom{n}{s}^2 |D|^{n-2s}} \,.$$

So, it remains to show that all vectors of the broom R are accepted by the program. This is a direct consequence of the following more general claim.

CLAIM 17.3. If both vectors $a = (a_0, w, a_1)$ and $b = (b_0, w, b_1)$ belong to B, then the combined vector (a_0, w, b_1) belongs to A.

To prove the claim, let $comp(a) = (p_1, \ldots, p_r)$ be an accepting computation on $a = (a_0, w, a_1)$, and $comp(b) = (q_1, \ldots, q_r)$ an accepting computation on $b = (b_0, w, b_1)$. Consider the combined vector $c = (a_0, w, b_1)$. Our goal is to show that then $p_t(c) \vee q_t(c) = 1$ for all $t = 1, \ldots, r$. That is, that for each $t = 1, \ldots, r$, the combined vector c must be accepted by (must be consistent with) at least one of the sub-paths p_t or q_t.

To show this, assume that c is not accepted by p_t. Since p_t accepts the vector $a = (a_0, w, a_1)$, and this vector coincides with the combined vector $c = (a_0, w, b_1)$ on all the variables outside X_1, this means that at least one variable from X_1 must be tested along p_t. But then, by the goodness of the pair X_0, X_1, no variable from X_0 can be tested along the sub-path q_t. Since q_t accepts the vector $b = (b_0, w, b_1)$, and the

[1] This rough estimate makes the whole argument useless for the boolean case, that is, when $D = \{0, 1\}$. At this place Ajtai (2005) uses probabilistic arguments to obtain a nontrivial lower bound on $|B|$ also in the boolean case; the arguments, however, do not work for *nondeterministic* branching programs.

combined vector $c = (a_0, w, b_1)$ coincides with this vector on all the variables outside X_0, the sub-path q_t must accept the vector c, as desired.

This completes the proof of Claim 17.3, and thus the proof of Lemma 17.1. □

17.2. A lower bound for code functions

Let q be a sufficiently large prime power; $q \geq 3k^{5k}$ is enough. Take $D = GF(q)$ and consider the function $g(Y, \boldsymbol{x})$ in $n^2 + n$ variables, the first n^2 of which are arranged in an $n \times n$ matrix Y. Let

$$g(Y, \boldsymbol{x}) = 1 \text{ iff the vector } \boldsymbol{x} \text{ is orthogonal over } GF(q) \text{ to all rows of } Y.$$

In other words, $g(Y, \boldsymbol{x}) = 1$ iff the vector $\boldsymbol{x}$ belongs to a linear code defined by the parity-check matrix Y.

THEOREM 17.4. *Every nondeterministic D-way branching program computing $g(Y, \boldsymbol{x})$ in linear time must have size $2^{\Omega(n)}$.*

The time restriction in this theorem concerns only the last n variables—the first n^2 variables from Y can be tested an arbitrary number of times!

PROOF. Let $k \geq 1$ be an arbitrary integer, and take a nondeterministic branching program computing $g(Y, \boldsymbol{x})$ in time at most kn. Let r and K be the constants from Lemma 17.1. Take $d = s + 1$ where $s := \lfloor n/K \rfloor$. By the Gilbert–Varshamov bound, linear codes $C \subseteq GF(q)^n$ of distance d and size $|C| \geq q^n / V(n, s)$ exist, where

$$V(n,s) = \sum_{i=0}^{s} (q-1)^i \binom{n}{i} \leq dq^s \binom{n}{s}$$

is the number of vectors in a Hamming ball of radius s around a vector in $GF(q)^n$.

Let Y be the parity-check matrix of such a code, and consider the function $f : GF(q)^n \to \{0,1\}$ such that $f(\boldsymbol{x}) = 1$ iff $Y \cdot \boldsymbol{x} = \boldsymbol{0}$. That is, $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} \in C$. The function $f(\boldsymbol{x})$ is a sub-function of $g(Y, \boldsymbol{x})$. Hence, if the function $g(Y, \boldsymbol{x})$ can be computed by a nondeterministic branching program working in time kn, then the size of this program must be at least the size ℓ of a nondeterministic branching program computing $f(\boldsymbol{x})$ in time kn. To finish the proof of Theorem 17.4, it remains therefore to show that ℓ must be exponential in $s = n/K$.

The function $f(\boldsymbol{x})$ accepts $|f^{-1}(1)| \geq q^n / V(n,s)$ vectors. Hence, by Lemma 17.1, the code C must contain an s-broom $R = R_0 \times \{w\} \times R_1$ of size

$$|R| \geq \frac{|f^{-1}(1)|}{\ell^r \binom{n}{s}^2} \cdot q^{2s-n} = \frac{q^{2s}}{\ell^r \binom{n}{s}^2 V(n,s)} \geq \frac{q^s}{\ell^r d \binom{n}{s}^3}. \tag{17.3}$$

On the other hand, since the Hamming distance between any two vectors in C is at least $d = s + 1$, none of the sets R_0 and R_1 can have more than one vector. Hence, $|R| \leq 1$. Remembering that $s = \lfloor n/K \rfloor$ and q is large enough this, we have that $\binom{n}{s}^3 \leq (q/2)^s$. Together with $|R| \leq 1$ and (17.3), this implies that $\ell^r \geq 2^s / d = 2^{\Omega(s)}$, and the desired lower bound $\ell = 2^{\Omega(s/r)} = 2^{\Omega(n)}$ follows. □

RESEARCH PROBLEM 17.5. *Prove an exponential lower bound on the size of for nondeterministic* boolean *($D = \{0,1\}$) branching programs in n variables, all whose consistent paths have length at most $n + 1$.*

Thus, if the computed function is sensitive, then only one variable is allowed to be re-tested along each accepting computation. But what makes the problem non-trivial is that the restriction is only on *consistent* paths. The case when the restriction is on *all* paths (be they consistent or not) is much easier to analyze, and we do this in the next section.

17.3. Syntactic read-k times programs

A nondeterministic branching program is *syntactic read-k times program* (k-n.b.p.) if along each its path (be it consistent or not) from the source to the target node each variable appears at most k times.

LEMMA 17.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a sensitive boolean function, $r = 10k^2$ and $K = k^{5k}$. If f can be computed by a k-n.b.p. of size ℓ, then for every $s \le n/K$, there exists an s-broom $R \subseteq f^{-1}(1)$ of size*

$$|R| \ge |f^{-1}(1)| \cdot 2^{2s-n} \cdot \ell^{-r}\,.$$

PROOF. As in the proof of Lemma 17.1, for each input $a \in f^{-1}(1)$, fix one accepting computation path $comp(a)$, and split it into r sub-paths $comp_1(a), \ldots, comp_r(a)$ length at most kn/r; as before, the length of a sub-path is the number of tests made along it. Call two inputs a and b in $f^{-1}(1)$ equivalent if, for each $i = 1, \ldots, r$, the starting nodes of the corresponding sub-paths $comp_i(a)$ and $comp_i(b)$ coincide. Since we have at most ℓ nodes in the program, the number of possible equivalence classes does not exceed ℓ^r. Fix some largest equivalence class $A \subseteq f^{-1}(1)$; hence, $|A| \ge |f^{-1}(1)|/\ell^r$. Let

$$S_i = \{x_j \mid x_j \text{ or } \neg x_j \text{ appears along } comp_i(a) \text{ for some } a \in A\}$$

be the set of variables that are tested along the ith sub-computation on at least one vector from A. Since our program is *syntactic* read-k, no variable can belong to more than k of the sets $S_1, \ldots, S_r$: otherwise we could find a (non necessarily consistent) path containing more than k ocurrencies of this variable. So, Claim 17.2 implies that there must be a pair X_0, X_1 of disjoint subsets of variables, each of size at least $n/2N$ with $N = \sum_{i=0}^{2k} \binom{r}{i}$ and such that $S_i \cap X_0 = \emptyset$ or $S_i \cap X_1 = \emptyset$ for each $i = 1, \ldots, r$. The rest of the proof is now the same as that of Lemma 17.1 with set A instead of B. □

To show an explicit lower bound for the size of k-n.b.p., consider binary linear (n, m, d)-codes. Recall that such a code is a linear subspace $C \subseteq GF(2)^n$ of dimension $n - m$ such that the Hamming distance between any two vectors in C is at least $2d + 1$. Bose-Chaudhury codes (BCH-codes) are linear (n, m, d)-codes C with $m \le d \log_2(n + 1)$. Such codes can be constructed for any n such that $n + 1$ is a power of 2, and for every $d < n/2$.

COROLLARY 17.7. *For every integer $k \ge 1$, the characteristic function of BCH-codes of minimal distance $d = \Omega(\sqrt{n})$ require k-n.b.p. of size $2^{\Omega(\sqrt{n})}$.*

PROOF. Let $C \subseteq \{0,1\}^n$ be a BCH (n, m, d)-code with $d = \lfloor \varepsilon \sqrt{n} \rfloor$, where $\varepsilon > 0$ is a sufficiently small constant. Let also f_C be the characteristic function of C, that is, $f_C(x) = 1$ iff $x \in C$. Let ℓ be the smallest size of a k-n.b.p. computing f_C. Since k is constant, both $r = 10k^2$ and $K = k^{5k}$ are constants. Moreover, $m \le d \log_2(n + 1)$, implying that

$$|f^{-1}(1)| = 2^{n-m} \ge 2^n/(n+1)^d\,.$$

Let $s = \lfloor n/K \rfloor$; hence, $s = \Theta(n)$. We already know (see Claim 15.4 in Section 15.1) that no code of minimal distance at least $2d+1$ can contain an s-broom of size larger than $2^{2s}\binom{s}{d}^{-2}$. By Theorem 17.6, this implies that

$$\ell^r \geq |f^{-1}(1)| \cdot 2^{-n} \cdot \binom{s}{d}^2 \geq \exp\left(2d\log_2(s/d) - d\log_2(n+1)\right)$$
$$= \exp\left(d\log_2 \frac{s^2}{d^2(n+1)}\right) = \exp(d) = 2^{\Omega(\sqrt{n})}. \qquad \square$$

Bibliographic Notes

The first exponential lower bound for linear time nondeterministic D-way branching programs was proved by Ajtai (2002). This was, however, done for domains D of size $|D| \geq n$. Beame, Jayram,[2] and Saks (2001) gave an exponential lower bound for functions over much smaller domains D whose size $|D| \approx 2^{2^k}$ only depends on k not on n. In the same paper they also proved the first exponential lower bound on the size of *deterministic* branching programs that work in time $(1+\varepsilon)n$ for a very small constant $\varepsilon > 0$ but are both *boolean* (work over $D = \{0,1\}$). They also showed that the lower bound $s \geq n/k^{5k}$ in Lemma 17.1 can be improved to $s \geq 2^{k+1}$, by using probabilistic arguments. Then Ajtai (2005) was able to modify his proof for D-way nondeterministic programs so that it gives exponential lower bounds for boolean deterministic branching programs working in time kn for *any* constant k. Ajtai achieves this by a very delicate probabilistic reasoning leading to a much sharper version of Lemma 17.1. This important result is, however, still too lengthy to be included here. We only note that Ajtai's argument essentially employs the fact that the underlying branching program is *deterministic*: this gives a 1-to-1 correspondence between input vectors and computations on them. For *nondeterministic* boolean branching programs nothing similar is known, even for programs working in time $n+1$. Exponential lower (for another functions) on the size of nondeterministic *syntactic* read-k times branching programs were proved by Borodin, Razborov and Smolensky (1993).

[2] Formerly Jayram S. Thathachar

CHAPTER 18

Propositional Proof Complexity

Propositional proof systems operate with boolean formulas, simplest of them being clauses, i.e., with ORs of literals, where each literal is either a variable x_i or its negation $\neg x_i$. A *truth-assignment* is an assignment of constants 0 and 1 to all the variables. Such an assignment *satisfies* (*falsifies*) a clause if it evaluates at least one (respectively, none) of its literals to 1. A set of clauses, that is, a CNF formula is satisfiable if there is an assignment which satisfies all its clauses. The basic question is:

> Given an *unsatisfiable* CNF formula F, what is the length of (number of clauses in) a proof that F is indeed unsatisfiable.

Such a proof starts with clauses of F (called *axioms*), at each step applies one of several (fixed in advance) simple rules of inferring new clauses from old ones, and must eventually produce an empty clause $\mathbf{0}$ (which, by definition, is satisfied by none of the assignments.

For such a derivation to be a legal proof, the rules must be *sound* in the following sense: if some assignment (of constants to all variables) falsifies the derived clause, then it must falsify at least one of the clauses from which it was derived. Then the fact that $\mathbf{0}$ was derived implies that the CNF F was indeed unsatisfiable: given any assignment α we can traverse the proof going from $\mathbf{0}$ to an axiom (a clause of F), and soundness of rules will give us a clause of F which is not satisfied by α.

The main goal of proof complexity is to show that some unsatisfiable CNFs require long proofs. The reason is its connection with the famous **P** versus **NP** question. This is because the problem SAT—given a CNF formula F, detect whether F is satisfiable or not—is an **NP**-complete problem. It is long known that **NP** = **co-NP** iff there is a propositional proof system giving rise to short (polynomial in $|F|$ length) proofs of unsatisfiability of all unsatisfiable CNFs F.

Thus, a natural strategy to approach the **P** versus **NP** problem is, just like in the circuit complexity, to investigate more and more powerful proof systems and show that some unsatisfiable CNFs require exponentially long proofs. In this chapter we will demonstrate this line of research on some basic proof systems, like resolution and cutting planes proofs.

18.1. Resolution and branching programs

The resolution proof system was introduced by Blake (1937) and has been made popular as a theorem-proving technique by Davis and Putnam (1960) and Robinson (1965).

Let F be a set of clauses and suppose that F is not satisfiable. A *resolution refutation proof*) (or simply, a *resolution proof*) for F is a sequence of clauses $\mathscr{R} = (C_1, \ldots, C_t)$ where $C_t = \mathbf{0}$ is the empty clause (which, by definition, is satisfied by no assignment) and each intermediate clause C_i either belongs to F or is derived from some previous

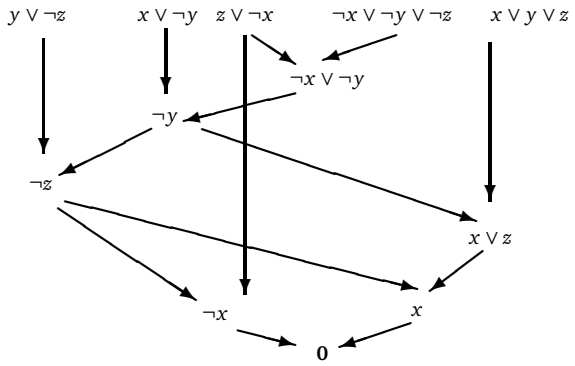


FIGURE 1. A resolution refutation proof of a CNF formula F. Leaves (fanin-0 nodes) are clauses of F, and each inner node is a clause obtained from previous ones by the resolution rule.

two clauses using the following *resolution rule*:

$$\frac{A \vee x_i \qquad B \vee \overline{x}_i}{A \vee B} \tag{18.1}$$

meaning that

the clause $A \vee B$ can be inferred from two clauses $A \vee x_i$ and $B \vee \neg x_i$.

In this case one also says that the variable x_i was *resolved* to derive the clause $A \vee B$. The *size* (or *length*) of such a proof is equal to the total number t of clauses in the derivation. It is often useful to describe a resolution proof as a directed acyclic graph, see Fig. 1. If this graph is a tree, then one speaks about a *tree-like* Resolution proof. For technical reasons the following "redundant" rule, the *weakening rule*, is also allowed: a clause $A \vee B$ can be inferred from A.

Observe that the resolution rule is *sound*: if some assignment (of constants to all variables) falsifies the derived clause $A \vee B$, then it must falsify at least one of the clauses $A \vee x_i$ and $B \vee \neg x_i$ from which it was derived. It is also known (and easy to show, see Exercise 18.2) that Resolution is *complete*: every unsatisfiable set of clauses has a resolution refutation proof.

What about the size of such derivations? Due to its practical importance, this question bothered complexity theoreticians and logicians for a long time. Interestingly enough resolution proof have a relation to a model we already considered above—branching programs.

This is a reason why we include resolution proofs in a part devoted to branching programs.

Let F be an unsatisfiable CNF formula, that is, for every input $a \in \{0,1\}^n$ there is a clause $C \in F$ for which $C(a) = 0$. The *search problem* for F is, given a, to find such a clause. (There may be several such clauses; the goal is to find at least one of them.) Such a problem may be solved by a branching program with $|F|$ leaves: label the leaves by clauses from F; then every input (truth assignment) $a \in \{0,1\}^n$ follows a unique path and finally reaches some leaf labeled by a clause C for which $C(a) = 0$.

Let $S_R(F)$ be the smallest size of a resolution refutation of F, and $\mathrm{BP}(F)$ the smallest size of a deterministic branching program solving the search problem for F. It is not difficult to show that $S_R(F) \geq \mathrm{BP}(F)$ (see the first part of the proof of Theorem 18.1

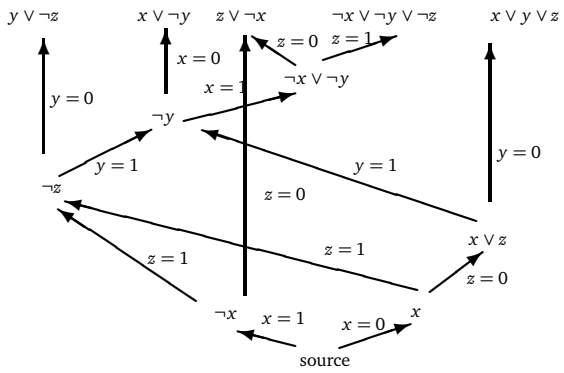


FIGURE 2. A branching program obtained from the resolution proof given in Fig. 1: just reverse the direction of arcs and label them accordingly. The program is not read-once.

below). But the gap between these two measures may be exponential: any unsatisfiable CNF F has a trivial branching program of size $|F|$ whereas, as we will show in the next section, some CNFs require $S_R(F)$ exponential in it variables. It is an interesting question to find a model of computation that is polynomialy equivalent to resolution.

First exponential lower bounds for Resolution proofs were obtained long ago by Tseitin (1968) under additional restriction that along every path every particular variable x_i can be resolved at most once. He called this model *regular* resolution. In particular, every tree-like resolution proof is regular. It turns out that this model just coincides(!) with the known model of read-once branching programs.

Let 1-$S_R(F)$ be the smallest size of a regular resolution refutation proof for F, and 1-$BP(F)$ the smallest size of a deterministic read-once branching program solving the search problem for F.

THEOREM 18.1. *For every unsatisfiable CNF formula F, we have that*

$$S_R(F) \geq \mathrm{BP}(F) \quad \textit{and} \quad \text{1-}S_R(F) = \text{1-}BP(F).$$

PROOF. To show that 1-$S_R(F) \geq$ 1-$BP(F)$, let $\mathscr{R}$ be a resolution refutation proof for F. Construct a branching program as follows.

a. The nodes of the program are clauses C of $\mathscr{R}$.
b. The source node is the last clause in $\mathscr{R}$ (the empty one), the sinks are the initial clauses from F.
c. Each non-sink node C has fanout 2 and the two edges directed from C to the two clauses C_0 and C_1 from which this clause is derived by one application of the resolution rule; if the resolved variable of this inference is x_i then the edge going to the clause containing x_i is labeled by the test $x_i = 0$, and the edge going to the clause containing $\neg x_i$ is labeled by the test $x_i = 1$ (see Fig. 2).

It is straightforward to verify that *all* clauses on a path determined by an input $a \in \{0,1\}^n$ are falsified by a, and hence, the last clause of F reached by this path is also falsified by a. That is, the obtained branching program solves the search problem and is read-once if $\mathscr{R}$ was regular.

It remains to prove the more interesting direction that 1-$S_R(F) \leq$ 1-$BP(F)$. Let P be a deterministic read-once branching program (1-b.p.) which solves the search

problem for F. That is, for every input $a \in \{0,1\}^n$ the (unique) computation path on a leads to a clause $C \in F$ such that $C(a) = 0$. We will associate a clause to every node of P such that P becomes a graph of a resolution refutation for F. A vertex v labeled by a variable will be associated with a clause C_v with the property that

$$C_v(a) = 0 \text{ for every input } a \in \{0,1\}^n \text{ that reaches } v. \tag{18.2}$$

We associate clauses inductively from the sinks backwards. If v is a sink then let C_v be the clause form F labeling this sink in the program P.

Assume now that the node v of P corresponds to a variable x_i and has edges (v,u_0) for $x_i = 0$ and (v,u_1) for $x_i = 1$. By induction we may assume that u_0 and u_1 are labeled by clauses C_0 and C_1 satisfying (18.2).

CLAIM 18.2. *C_0 does not contain $\neg x_i$ and C_1 does not contain x_i.*

PROOF. Otherwise, if C_0 contains $\neg x_i$, take an input a with $a_i = 0$ that reaches v. Such an input exists since by the read-once assumption on P, the ith bit x_i was not asked along any path from the source to v. The input a can reach u_0 and it satisfies C_0, in contradiction to the inductive hypothesis. The proof in the case when C_1 contains x_i is similar. □

We conclude that either: (i) $C_0 = (x_i \vee A)$ and $C_1 = (\neg x_i \vee B)$, or (ii) one of C_0 and C_1 does not contain $x_i, \neg x_i$ at all. In the first case label v with $C_v = A \vee B$. In the second case label v with the clause that does not contain $x_i, \neg x_i$. (If both clauses do not contain $x_i, \neg x_i$ chose any of them.)

It is easy to see that the inductive hypothesis (18.2) holds for C_v: since the program P is read-once, any computation path from the source node to v can be prolonged in *both* directions. Moreover, the clause associated with the source node must be the empty clauses, just because *every* input reaches it. Thus the obtained labeled digraph represents a regular resolution derivation for F (possibly with some redundant steps that correspond to the second case (ii) in the labeling above. □

REMARK 18.3. Note that Claim 18.2 holds for *any* deterministic branching program, not just for read-once ones: it is enough that P is a *minimal* program. Indeed, in this case a node must be reachable by (at least) two inputs a and b such that $a_i = 0$ and $b_i = 1$, for otherwise the test on the ith bit made at the node v would be redundant. Where read-once property was important is the conclusion that so constructed clause C_v satisfies (18.2). Namely, if $C_0 = (x_i \vee A)$, $C_1 = (\neg x_i \vee B)$ and $C_v = A \vee B$, and if $A(a) = B(b) = 0$ but $A(b) = 1$ or $B(a) = 1$, then $C_0(a) = 0$ and $C_1(b) = 0$ but $C_v(a) = 1$ or $C_v(b) = 1$. In the read-once case such a situation cannot occur because then every (single) computation reaching a node v can be extended in *both* directions.

18.2. Exponential lower bound for resolution

The pigeonhole principle asserts that if $m \geq n+1$ then m pigeons cannot sit in n holes so that every pigeon is alone in its hole. In terms of 0-1 matrices, this principle asserts that, if $m \geq n+1$ then every $m \times n$ 0-1 matrix satisfies *precisely one* of the following two conditions:

1. Every row has at least one 1.
2. Every column has at most one 1.

To write this principle as an unsatisfiable CNF formula, we introduce boolean variables $x_{i,j}$ interpreted as:

$$x_{i,j} = 1 \text{ if and only if the } i\text{th pigeon sits in the } j\text{th hole.}$$

Let PHP_n^m denote the AND of the following clauses (we call them *axioms*):

a. Pigeon Axioms: each of the m pigeon sits in at least one of n holes:

$$x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n} \text{ for all } i = 1, \ldots, m.$$

b. Hole Axioms: no two pigeons sit is one hole:

$$\neg x_{i_1,j} \vee \neg x_{i_2,j} \text{ for all } i_1 \neq i_2 \text{ and } j = 1, \ldots, n.$$

Hence, truth assignments in this case are $m \times n$ $(0,1)$ matrices α. Such a matrix can satisfy all pigeon axioms iff every row has at least one 1, whereas it can satisfy all hole axioms iff every column has at most one 1. Since $m \geq n+1$, no assignment can satisfy pigeon axioms and hole axioms at the same time. So, PHP_n^m is indeed an unsatisfiable CNF.

THEOREM 18.4. *For a sufficiently large n, any resolution refutation proof of PHP_{n-1}^n requires size $2^{\Omega(n)}$.*

PROOF. The proof is by contradiction. We define an appropriate notion of a "fat" clause and show two things:

a. If PHP_{n-1}^n has a short resolution proof, then it is possible to set some variables to constants so that the resulting proof is a refutation of PHP_{m-1}^m for a large enough m, and has no fat clauses.
b. If m is large enough, then every refutation proof for PHP_{m-1}^m must have at least one fat clause.

This implies that PHP_{n-1}^n cannot have short resolution proofs.

In the case of the CNF formula PHP_{n-1}^n truth assignments α are n by $n-1$ $(0,1)$ matrices. We say that a truth assignment α is *i-critical* if

a. the ith row of α is the all-0 row, and
b. every column has exactly one 1.

Note that each such assignment α is barely unsatisfying: it satisfies all hole axioms as well as the axioms of all but the ith pigeon. That is, the only axiom it falsifies is the pigeon axiom $C_i = x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n-1}$.

The properties of critical truth assignments make it convenient to convert each clause C to a positive clause C^+ that is satisfied by precisely the same set of critical assignments as C. More precisely to produce C^+, we replace each negated literal $\neg x_{i,j}$ with the OR

$$X_{i,j} = x_{1,j} \vee \cdots \vee x_{i-1,j} \vee x_{i+1,j} \vee \cdots \vee x_{n,j}\,.$$

CLAIM 18.5. For every critical truth assignment α, $C^+(\alpha) = C(\alpha)$.

PROOF. Suppose there is a critical assignment α such that $C^+(\alpha) \neq C(\alpha)$. This can only happen if C contains a literal $\neg x_{i,j}$ such that $\neg x_{i,j}(\alpha) \neq X_{i,j}(\alpha)$. But this is impossible, since α has precisely one 1 in the jth column. □

Associate with each clause in a refutation of PHP_{n-1}^n the set

$$\text{Pigeon}(C) = \{i \mid \text{there is some } i\text{-critical assignment } \alpha \text{ such that } C(\alpha) = 0\}$$

of pigeons that are "bad" for this clause: some critical assignments of these pigeons falsify C.

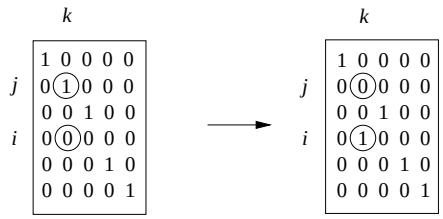


FIGURE 3. Assignment α' is obtained from α by interchanging the ith and jth rows.

CLAIM 18.6. Every resolution refutation of PHP^n_{n-1} must have a clause C with $|C^+| \geq n^2/9$.

PROOF. Define the *weight* of a clause C as $\mu(C) := |\text{Pigeon}(C)|$. By the definition, each hole axiom has weight 0, each pigeon axiom has weight 1, and the last (empty) clause has weight n since it is falsified by any truth assignment. Moreover, this weight measure is "subadditive:" if clauses A and B imply clause C, then $\mu(C) \leq \mu(A) + \mu(B)$: every assignment falsifying C must falsify at least one of the clauses A and B. Therefore, if C is the first clause in the proof with $\mu(C) > n/3$, we must have

$$n/3 < \mu(C) \leq 2n/3\,. \tag{18.3}$$

Fix such a "medium heavy" clause C and let $s = \mu(C)$ be its weight. Since $n/3 < s \leq 2n/3$, it is enough to show that the positive version C^+ of this clause must have $|C^+| \geq s(n-s)$ distinct variables.

Fix some $i \in \text{Pigeon}(C)$ and let α be an i-critical truth assignment with $C(\alpha) = 0$. For each $j \notin \text{Pigeon}(C)$, define the j-critical assignment α', obtained from α by toggling rows i and j. That is, if α maps the ith pigeon to the kth hole, then α' maps the jth pigeon to this hole (see Fig. 3).

Now $C(\alpha') = 1$ since $j \notin \text{Pigeon}(C)$. By Claim 18.5, we have that $C^+(\alpha) = 0$ and $C^+(\alpha') = 1$. Since the assignments α,α' differ only in the variables $x_{i,k}$ and $x_{j,k}$, this can only happen when C^+ contains the variable $x_{i,k}$.

Running the same argument over all $n-s$ pigeons $j \notin \text{Pigeon}(C)$ (using the same α), it follows that C^+ must contain at least $n-s$ of the variables $x_{i,1}, \ldots, x_{i,n-1}$ corresponding to the ith pigeon. Repeating the argument for all pigeons $i \in \text{Pigeon}(C)$ shows that C^+ contains at least $s(n-s)$ variables, as claimed. □

We can now finish the proof of Theorem 18.4 as follows. Let $\mathscr{R}$ be a resolution refutation proof of PHP^n_{n-1}, and $S = |\mathscr{R}|$ be the total number of clauses in it. Let a and $b \geq 2$ be positive constants (to be specified later). For the sake of contradiction, assume that

$$S < e^{n/a}\,.$$

Together with $\mathscr{R}$ we consider the set $\mathscr{R}^+ = \{C^+ \mid C \in \mathscr{R}\}$ of positive versions of clauses in $\mathscr{R}$. Say that a clause of $\mathscr{R}^+$ is *fat* if it has at least n^2/b variables. Since each fat clause has at least a $1/b$ fraction of all the variables, there must be (by the pigeonhole principle!) a variable $x_{i,j}$ which occurs in at least S/b of fat clauses in $\mathscr{R}^+$.

Set this "popular" variable to 1, and at the same time set to 0 all the variables $x_{i,j'}$ and $x_{i',j}$ for all $j' \neq j, i' \neq i$ (see Fig. 4). After this setting, all the clauses containing

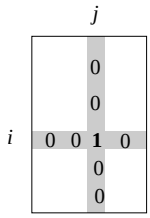


FIGURE 4. Setting of constants to eliminate clauses containing $x_{i,j}$; non-shaded positions are not set. This way PHP^n_{n-1} is reduced to PHP^{n-1}_{n-2}.

$x_{i,j}$ will disappear from $\mathscr{R}^+$ (they all get the value 1) and the variables which are set to 0 will disappear from the remaining clauses.

Applying this restriction to the entire proof $\mathscr{R}$ leaves us with a refutation proof $\mathscr{R}_1$ for PHP^{n-1}_{n-2}, where the number of fat clauses in $\mathscr{R}^+_1$ is at most $S(1-1/b)$. Continue in this fashion until we have set all fat clauses to 1. Applying this argument iteratively $d = b\ln S < (b/a)n$ times, we are guaranteed to have knocked out all fat clauses, because

$$S(1-1/b)^d < e^{\ln S - d/b} = 1\,.$$

Thus, we are left with a refutation proof for PHP^m_{m-1}, where

$$m = n - d \geq (1 - b/a)n\,,$$

and where $|C^+| < n^2/b$ for *all* its clauses. But Claim 18.6 implies that any refutation proof of PHP^m_{m-1} must contain a clause C for which

$$n^2/b > |C^+| \geq m^2/9 = (1-b/a)^2 n^2/9\,.$$

To get the desired contradiction, it is enough to chose the parameters a and b so that $(1-b/a)^2 \geq 9/b$ which, in particular, is the case for $b = 16$ and $a = 4b$. □

The reader may wonder where in this proof we used that the clauses in a refutation are derived using only resolution and weakening rules? The same argument seems to work for more general derivations? And this is indeed so—the only important thing was that the formulas in such a derivation are clauses: this allowed us to kill off a clause by setting just one variable to constant.

A closer look at the proof shows that it also works for any *semantic* derivation $\mathscr{R} = (C_1, \ldots, C_t)$ such that $C_t = 0$ is the empty clause and each C_j is either an axion (belongs to F) or is implied by some k previous clauses $C_{i_1}, \ldots, C_{i_k}$ in a sense that, for all $\alpha \in \{0,1\}^n$,

$$C_{i_1}(\alpha) = 1, \ldots, C_{i_k}(\alpha) = 1 \quad \text{implies} \quad C_j(\alpha) = 1\,.$$

The only difference is that now instead of (18.3) we will have

$$\frac{n}{k+1} < \mu(C) \leq \frac{kn}{k+1}\,,$$

which results in a lower bound $|C^+| \geq n^2/(k+1)^2$ in Claim 18.6. The rest ist the same with constants $b := 4(k+1)$ and $a := 2b$.

18.3. Short proofs are narrow

We have already seen that "fat" clauses—those whose length exceeds some given threshold value—play a crucial role in trying to show that the size of a resolution proof (= the total number of lines in it) must be large. We are now going to show that this is a general phenomenon, not just an accident: If any resolution proof for an unsatisfiable CNF formula F must contain at least one fat clause, then F cannot have a short resolution proof.

The width $w(C)$ of a clause is just the number of literals in it. If F is a set of clauses then its width $w(F)$ is the maximum width of its clause. Recall that each resolution refutation $\mathscr{R}$ is also a set (more precisely, a sequence) of clauses. Hence, the width of a refutation is also the maximum width of a clause participating in it.

Let now F be an unsatisfiable CNF in n variables. Define its *resolution refutation width* $w_R(F)$ as the minimum width of a resolution refutation of F. The *resolution refutation size* $S_R(F)$ is, as before, the minimum number of clauses in a refutation of F. That is,

$$w_R(F) = \min\{w(\mathscr{R}) : \mathscr{R} \text{ is a resolution refutation proof of } F\}$$

and

$$S_R(F) = \min\{|\mathscr{R}| : \mathscr{R} \text{ is a resolution refutation proof of } F\}.$$

Note that refutation proofs $\mathscr{R}$ achieving $w_R(F)$ and $S_R(F)$ may be different!

What is the relation between these parameters? If we use all clauses of the CNF F in its refutation, then $w_R(F) \geq w(F)$. But it is not true in general: one may not use all clauses of F for its refutation, one might be able to deduce the empty clause from a subset of its clauses.

The relation $S_R(F) \leq (2n)^{w_R(F)}$ between prof-size and proof-width is easy to see: since we only have $2n$ literals, the number of all possible clauses of width k does not exceed $(2n)^k$. Much more interesting is the following *lower* bound on proof size in terms of proof width: only CNF formulas having narrow proofs can be proved in a short time!

THEOREM 18.7. *For any unsatisfiable k-CNF formula F in n variables,*

$$\log_2 S_R(F) \geq \frac{(w_R(F)-k)^2}{16n}.$$

For the proof of this theorem we need a concept of a *restriction* of CNFs and of refutation proofs. Let F be some set of clauses (think of F as a CNF or as a refutation proof). Let x be some of its literals. If we set this literal to 0 and to 1, then we obtain two sets of clauses:

$F_{x=0}$ is F with literal x removed from all clauses of F;
$F_{x=1}$ is F with all clauses containing x removed from F.

Note that, if F was an unsatisfiable CNF, then both CNFs $F_{x=0}$ and $F_{x=1}$ remain unsatisfiable. Moreover, if $\mathscr{R}$ was a resolution refutation proof of F and $a \in \{0,1\}$, then $\mathscr{R}_{x=a}$ is also a resolution refutation proof of $F_{x=a}$. If at some step in $\mathscr{R}$ a literal x is resolved using the resolution rule, then this step in $\mathscr{R}_{x=a}$ corresponds to an application of the weakening rule:

$$\frac{A \vee x \quad B \vee \neg x}{A \vee B} \quad \mapsto \quad \frac{A}{A \vee B} \quad \text{or} \quad \frac{B}{A \vee B}.$$

LEMMA 18.8. *If $w_R(F_{x=1}) \leq w-1$ and $w_R(F_{x=0}) \leq w$, then $w_R(F) \leq \max\{w,k\}$.*

PROOF. The idea is to combine refutations for $F_{x=1}$ and for $F_{x=0}$ into one refutation proof for F. First we can deduce $\neg x$ from $F_{x=1}$ using clauses of width at most w. To do this, follow closely the deduction of an empty clause from $F_{x=1}$, which uses clauses of width at most $w-1$, and add the literal $\neg x$ to every clause in that deduction. Let $\mathscr{R}$ be the resulting deduction of $\neg x$ from $F_{x=1}$. Now, from $\neg x$ and F we can deduce $F_{x=0}$ by using the resolution rule: just resolve $\neg x$ with each clause of F containing x to get $F_{x=0}$. This step does not introduce any clause of width more than k. Finally, deduce the empty clause from $F_{x=0}$ using clauses of width at most w. □

Let now W be a parameter (to be specified later), and call a clause *fat* if it has width larger than W. Set also

$$a := \left(1 - \frac{W}{2n}\right)^{-1} \geq e^{W/2n}.$$

LEMMA 18.9. *If a k-CNF F has a refutation that contains less than a^b fat clauses then*

$$w_R(F) \leq W + b + k.$$

PROOF. We prove this by induction on b and n. The base case $b = 0$ is trivial, since then we have no fat clauses at all implying that $w_R(F) \leq W + k$.

Assume now that the claim holds for all smaller values of n and b. Take a resolution refutation $\mathscr{R}$ of F using $< a^b$ fat clauses. Since there are at most $2n$ literals and any fat clause contains at least W of them, an average literal must occur in at least $W/2n$ fraction of fat clauses. Choose a literal x that occurs most frequently in fat clauses and set it to 1. The obtained refutation $\mathscr{R}_{x=1}$ of $F_{x=1}$ has fewer than $a^b(1 - \frac{W}{2n}) = a^{b-1}$ fat clauses. By induction on b we have $w_R(F_{x=1}) \leq W + (b-1) + k$. On the other hand, since $F_{x=0}$ has one variable fewer, induction on n yields $w_R(F_{x=0}) \leq W + b + k$. The desired upper bound $w_R(F) \leq W + b + k$ now follows from Lemma 18.8. □

PROOF OF THEOREM 18.7. Choose b so that $a^b = S_R(F)$. Then

$$b = \frac{\log S_R(F)}{\log a} \leq \frac{2n \log S_R(F)}{W \ln 2} \leq \frac{4n \log S_R(F)}{W}$$

and, by Lemma 18.9,

$$w_R(F) \leq W + \frac{4n \log S_R(F)}{W} + k.$$

Choosing $W := 2\sqrt{n \log S_R(F)}$ to minimize the right-hand side yields the desired upper bound $w_R(F) \leq 4\sqrt{n \log S_R(F)} + k$. □

REMARK 18.10. That Theorem 18.7 cannot be substantially improved was shown by Bonet and Galesi (1999): there are unsatisfiable c-CNF formulas F (c being a constant) such that $S_R(F) \leq n^{O(1)}$ but $w_R(F) = \Omega(\sqrt{n})$.

18.3.1. Expanding formulas require large refutation width. As such, the measure $w_R(F)$ is difficult to deal with: we must take *all* possible resolution refutations of F into account. Still, there are properties of CNFs F forcing $w_R(F)$ to be large. Namely, it is enough that F has good "expansion" properties. For this, we look at a CNF formula F as a *set* of its clauses. Hence, $|F|$ denotes the number of clauses in F, and $G \subseteq F$ means that the CNF G contains only clauses of F.

Let $v(F)$ denote the number of variables in F, and call a CNF formula F is (r,c)-*expanding* if

$$v(G) \geq (1+c)|G| \text{ for every subset } G \subseteq F \text{ of its } |G| \leq r \text{ clauses.}$$

We can associate with F a bipartite graph, where nodes on the left part are clauses of F, nodes on the right part are variables, and a clause C is joined to a variable x iff x or $\neg x$ belongs to C. Then F is (r,c)-expanding iff every subset of $s \leq r$ nodes on the left part have at least $(1+c)s$ neighbors on the right part.

THEOREM 18.11. *For every unsatisfiable (r,c)-expanding CNF formula F,*

$$w_R(F) \geq \frac{cr}{2}.$$

We first prove three claims relating the number of clauses with the number of variables in unsatisfiable CNF formulas.

CLAIM 18.12. If $|G| \leq \nu(G)$ for every $G \subseteq F$, then F is satisfiable.

PROOF. We will use the well-known Hall's Marriage Theorem. It states that a family $\mathscr{S} = \{S_1, \ldots, S_m\}$ has a system of distinct representatives (that is, a sequence $x_1, \ldots, x_m$ of elements such that $x_i \in S_j$ iff $i = j$) if the union of any number $1 \leq k \leq m$ of members of $\mathscr{S}$ has at least k elements.

Assume now that $|G| \leq \nu(G)$ for all $G \subseteq F$. Then, by Hall's theorem, we can find for each clause C of F a variable $x_C \in \nu(C)$ which appears in none of the remaining clauses of F. Since each variable x_C is "unique" for the corresponding clause C, we can set these variables to 0 or 1 independently to make the corresponding clauses true. Hence, F is satisfiable. □

Say that an unsatisfiable CNF formula is *minimal* unsatisfiable if removing any clause from it makes the remaining CNF satisfiable. The following claim is also known as Tarsi's Lemma.

CLAIM 18.13. If F is minimally unsatisfiable, then $|F| > \nu(F)$.

PROOF. Since F is unsatisfiable, Claim 18.12 implies that there must be a subset of clauses $G \subseteq F$ such that $|G| > \nu(G)$. Let $G \subseteq F$ be a *maximal* subset of clauses with this property. If $G = F$ then we are done, so assume that $G \subset F$ and we will derive a contradiction.

Take an arbitrary sub-formula $H \subseteq F - G$, and let $\mathrm{Vars}(H)$ be the set of its variables. Due to maximality of G, $\mathrm{Vars}(H) - \mathrm{Vars}(G)$ must have at least $|H|$ variables, for otherwise we would have that $\nu(G \cup H) < |G \cup H|$, a contradiction with the maximality of G.

Thus, the CNF formula $F - G$ satisfies the condition of Claim 18.12, and hence, can be satisfied by only setting constants to variables in $\mathrm{Vars}(F) - \mathrm{Vars}(G)$. Since F is minimally unsatisfiable, the CNF formula G must be satisfiable using only the variables in $\mathrm{Vars}(G)$. Altogether this gives us a truth assignment satisfying the entire formula F, a contradiction. □

We say that a CNF formula F *implies* a clause A if any assignment satisfying F also satisfies A. We also say that F *minimally implies* A if the CNF formula F implies A but no its proper subformula (obtained by removing any its clause) does this.

CLAIM 18.14. If F minimally implies A then $|A| > \nu(F) - |F|$.

PROOF. Let $\mathrm{Vars}(F) = \{x_1, \ldots, x_n\}$ and assume w.l.o.g. that $\mathrm{Vars}(A) = \{x_1, \ldots, x_k\}$. Take a (unique) assignment $\alpha \in \{0,1\}^k$ for which $A(\alpha) = 0$. Since F implies A, restricting F to α must yield an unsatisfiable formula F_α on variables $x_{k+1}, \ldots, x_n$. The formula F_α must also be minimally unsatisfiable because F minimally implied A. By

Claim 18.13, F_α must have more than $n-k$ clauses. Hence, $|F| \geq |F_\alpha| > n-k = \nu(F) - |A|$, as desired. □

We now turn to the actual proof of the theorem.

PROOF OF THEOREM 18.11. Let F be an (r,c)-expanding unsatisfiable CNF formula, and let $\mathscr{R}$ be any resolution refutation proof of F. We can assume that both numbers r and c are positive (otherwise there is nothing to prove). With each clause C in $\mathscr{R}$ associate the number

$$\mu(A) = \min\{|G| : G \subseteq F \text{ and } G \text{ implies } A\}.$$

It is clear that $\mu(A) \leq 1$ for all clauses A of F. Furthermore, μ is subbaditive: $\mu(C) \leq \mu(A) + \mu(B)$ if C is a resolvent of A and B. Finally, the expansion property of F implies that $\mu(0) > r$. Indeed, by the definition, $\mu(0)$ is the smallest size $|G|$ of an unsatisfiable subformula $G \subseteq F$, and Claim 18.13 yields $|G| > \nu(G)$. Would we now have $\mu(0) \leq r$, then we would also have $|G| \leq r$ and the expansion property of F would imply $\nu(G) \geq (1+c)|G|$, a contradiction.

Hence, the subadditivity of μ implies that the refutation $\mathscr{R}$ of F must contain a clause C such that $r/2 \leq \mu(C) < r$. Fix some $G \subseteq F$ minimally implying C; hence, $|G| = \mu(C) < r$. By the expansion of F, $\nu(G) \geq (1+c)|G|$. Together with Claim 18.14 this implies $|C| > \nu(G) - |G| \geq c|G| \geq cr/2$, as desired. □

18.3.2. Matching principles for graphs. Given a bipartite $m \times n$ graph $G = ([m],[n],E)$, we may consider the CNF formula $PHP(G)$ which is an AND of the following set of axioms:

- Pigeon Axioms: $C_i = \bigvee_{(i,j)\in E} x_{i,j}$ for $i = 1,\ldots,m$.
- Hole Axioms: $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for $i_1 \neq i_2 \in [m]$ and $j \in [n]$.

That is, the graph dictates what holes are offered to each pigeon, whereas hole axioms forbid (as in the case of PHP_n^m) that two pigeons sit in one hole.

Observe that, if $m > n$ and if the graph G has no isolated vertices, then the CNF formula $PHP(G)$ is unsatisfiable. Indeed, every truth assignment α defines a subgraph G_α of G. Now, if α satisfies all hole axioms then G_α must be a (possibly empty) matching. But we have $m > n$ vertices of the left side. Hence, at least one of these vertices $i \in [m]$ must remain unmatched in G_α, implying that $C_i(\alpha) = 0$.

Observe also that $PHP_n^m = PHP(K_{m,n})$ where $K_{m,n}$ is a complete bipartite $m \times n$ graph. Moreover, if G' is a subgraph of G, then every resolution refutation for $PHP(G)$ can be turned to a resolution refutation of $PHP(G')$ just by setting to 0 all variables corresponding to edges of G that are not present in G'. Thus, to prove a lower bound of the resolution complexity of $PHP(G)$ it is enough to prove such a bound for *any* subgraph of G.

This opens plenty of possibilities to prove large lower bounds for PHP_n^m: just show that the *exists* a graph G (a subgraph of $K_{m,n}$) such that $PHP(G)$ requires large long resolution refutations. By Theorems 18.7 and 18.11, this can be done by showing that the CNF formula $F = PHP(G)$ has large expansion. This, in turn, can be achieved if the underlying graph G itself has good expansion properties.

A bipartite graph is (r,c)-*expander* if every set of $k \leq r$ vertices on the left part has at least $(1+c)k$ neighbors on the right part. It can be easily shown (Exercise 18.4) that if G is an (r,c)-expander then the CNF formula $PHP(G)$ is (r,c)-expanding.

Using a simple probabilistic argument it can be shown that (r,c)-expanders with $c > 0$, $r = \Omega(n)$ and *constant* left-degree exist (Exercise 18.5). Hence, the CNF formula

$F = PHP(G)$ has $N = O(m)$ variables and each its clause has constant width. Theorem 18.11 implies that $w_R(F) = \Omega(n)$. So, by Theorem 18.7, every resolution refutation for F, and hence, for PHP_n^m must have size exponential in $w_R(F)^2/N = \Omega(n^2/m)$.

This lower bound is super-polynomial, as long as we have $m \ll n^2/\log n$ pigeons. However, the larger m is, the more true the pigeonhole principle itself is, and it could be that PHP_n^m could be refuted by much shorter resolution refutation proof. And indeed, all attempts to overcome this "n^2 barrier" for the number of pigeons failed for many years. This was one of most famous open problems in the propositional proof complexity. The "n^2 barrier" for PHP_n^m was first broken by Raz (2001): PHP_n^m requires resolution proofs of exponential size for *any* number $m \geq n+1$ of pigeons. A simpler proof was then found by Razborov (2003), and we will present it in Section 18.6.

18.4. Local search for satisfiability

The 3SAT-problem is, given a 3-CNF F to decide whether it is satisfiable. This is the most famous **NP**-complete problem. Thus, any proof that 3SAT requires a super-polynomial (in the number of clauses) time would imply $\mathbf{P} \neq \mathbf{NP}$. Due to its importance, many algorithms for 3SAT were introduced, resolution being one of them: try to resolve literals one by one until a contradiction (an empty clause) is produced. But if the CNF is satisfiable, such an algorithm will stuck without an answer.

A trivial algorithm, which newer gets stuck, is just to probe all 2^n possible assignments. A less trivial algorithm does a "local search:" it starts with some assignment, and tries to flip its bits one by one in a hope to reach a satisfying assignment, if there is one.

18.4.1. Local search for 2-CNFs works well. Let F be a CNF in n variables, and suppose that we know that it is satisfiable. How quickly can we find a satisfying assignment? If each clause of F has exactly 2 literals, then a satisfying assignment can be found in $O(n^2)$ steps by the following simple randomized procedure.

Suppose we start with an arbitrary assignment of values to the literals. As long as there is a clause that is unsatisfied, we modify the current assignment as follows: we choose an arbitrary unsatisfied clause and pick one of the (two) literals in it uniformly at random; the new assignment is obtained by complementing the value of the chosen literal. After each step we check if there is an unsatisfied clause; if not, the algorithm terminates successfully with a satisfying assignment.

THEOREM 18.15. *Suppose that F is a satisfiable 2-CNF in n variables. Then, with probability at least $1/2$, the above algorithm will find a satisfying assignment in $2n^2$ steps.*

PROOF. Fix an arbitrary satisfying assignment $\alpha \in \{0,1\}^n$ for F, and refer to the values assigned by α to the literals as the "correct values."

The progress of the above algorithm can be represented by a particle moving between the integers $\{0, 1, \ldots, n\}$ on the real line. The position of the particle indicates how many variables in the current solution have "incorrect values," i.e., values different from those in α. At each iteration, we complement the current value of one of the literals of some unsatisfied clause, so that the particle's position changes by 1 at each step. In particular, a particle currently in position i, for $0 < i < n$, can only move to positions $i-1$ or $i+1$ (see Fig. 5).

Let $t(i)$ denote the expected number of steps which a particle, started in position i, makes until it reaches position 0. Our goal is to show that $t(i) \leq n^2$ for all i.

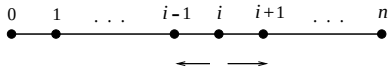


FIGURE 5. Random walk on a line for 2-CNFs. The particle being in position i means that the current assignment differs in i bits from the fixed (but unknown) satisfying assignment.

A particle at location n can only move to $n-1$, and the process terminates when the particle reaches position 0 (although it may terminate earlier at some other position with a satisfying assignment other than a). Hence, $t(n) \leq t(n-1)+1$ and $t(0)=0$. In general, we have that

$$t(i) = p_{i,i-1} \cdot (1 + t(i-1)) + p_{i,i+1} \cdot (1 + t(i+1)) \,,$$

where $p_{i,j}$ is the probability with which the particle moves from position i to position $j \in \{i-1, i+1\}$.

The crucial observation is the following: in an unsatisfied clause at least one of the literals has an incorrect value. Thus, with probability at least $1/2$ we decrease the number of variables having false values. The motion of the particle thus resembles a random walk on the line where the particle moves from the ith position ($0 < i < n$) to position $i-1$ with probability $p_{i,i-1} \geq 1/2$. This implies that

$$t(i) \leq \frac{t(i-1)+t(i+1)}{2} + 1.$$

Replace the obtained inequalities by equations

$$\begin{aligned} x(0) &= 0, \\ x(i) &= \frac{x(i-1)+x(i+1)}{2} + 1, \\ x(n) &= x(n-1) + 1 \,. \end{aligned}$$

This resolves to $x(1) = 2n-1$, $x(2) = 4n-4$ and in general $x(i) = 2in - i^2$. Therefore, $t(i) \leq x(i) \leq x(n) = n^2$, as desired.

By Markov's inequality, a random variable can take a value 2 times larger than its expectation only with probability $< 1/2$. Thus, the probability that the particle will make more than $2 \cdot t(i)$ steps to reach position 0 from position i, is smaller than $1/2$. Hence, with probability at least $1/2$ the process will terminate in at most $2n^2$ steps, as claimed. □

18.4.2. Local search for 3-CNFs fails. The *local search algorithm* for a satisfiable CNF picks an initial assignment in $\{0,1\}^n$ at random, and flips its bits one by one trying to satisfy all clauses. At each step, the decision on what bit of a current assignment α to flip is also random one. The algorithm first constructs a set $I \subseteq [n]$ of bits such that flipping any bit $i \in I$ increases the number of satisfied clauses. Then it chooses one of these bits at random, and flips it. If $I = \emptyset$, then the algorithm chooses one bit at random from the set of bits that do not lead to the decrease of the number of satisfied clauses. If all variables lead to such a decrease, it chooses at random a bit from $[n]$

The algorithm works in iterations, one iteration being a random choice of an initial assignment. We are interested in how many iterations are needed to find a satisfying assignment with a constant probability.

THEOREM 18.16. *There is a satisfiable 3-CNF formula F with n variables and $O(n^3)$ clauses such that the local search algorithm needs $2^{\Omega(n)}$ iterations to find a satisfying assignment with a constant probability.*

PROOF. The desired CNF formula F is an AND of two CNFs G and H. The first CNF G consists of $n+1$ clauses:

$$\neg x_1 \vee x_2\,,\ \neg x_2 \vee x_3\,,\ \ldots,\ \neg x_n \vee x_1 \quad \text{and} \quad \neg x_1 \vee \neg x_2\,.$$

The first n clauses express that in every satisfying assignment for G the values of all its bit must be equal. The last clause of G ensures that all these values must be equal to 0. Hence, $\alpha = \mathbf{0}$ is the only assignment satisfying all the $n+1$ clauses of G.

The second CNF H consists of al $n\binom{n-1}{2}$ clauses of the form

$$\neg x_i \vee x_j \vee x_k\,.$$

Hence, $\alpha = \mathbf{0}$ is the unique satisfying assignment for the entire CNF $F = G \wedge H$. The clauses in H are intended for "misleading" the algorithm.

We will show that there is a threshold t such that the assignments with t 1's form an "insurmountable ring" around the (unique) satisfying assignment $\mathbf{0}$. Namely, if the algorithm encounters an assignment α with[1] $|\alpha| \geq t$, then it chooses a wrong bit for flipping. That is, on such assignments α the algorithms flips some 0-bit to 1-bit, and hence, goes away from the satisfying assignment $\mathbf{0}$. As a threshold t we take $t := n/3 + c$ where c is a sufficiently large constant. Important for us will only be that under this choice, we have for all $k \geq t$:

$$(k-1)(n-k-1) > \binom{n-k}{2} + 4\,.$$

CLAIM 18.17. Let $\alpha \in \{0,1\}^n$ be an assignment with $|\alpha| \geq t$. Then the number of satisfied clauses of F:

a. decreases when flipping a 1-bit of α to 0;
b. increases when flipping a 0-bit of α to 1.

PROOF. Fix an $i \in [n]$ such that $\alpha_i = 1$. Flip the ith bit of α from 1 to 0, and let $\mathrm{Won}(1 \mapsto 0)$ be the set of clauses of F that were unsatisfied by α but are satisfied by the new assignment α'. Similarly, let $\mathrm{Lost}(1 \mapsto 0)$ be the set of clauses of F that were satisfied by α but are satisfied by the new assignment.

Each clause $C \in \mathrm{Won}(1 \mapsto 0)$ has the form $C = \neg x_i \vee x_u \vee x_v$ where u and v are such that $\alpha_u = \alpha_v = 0$. The number of such clauses is therefore

$$|\mathrm{Won}(1 \mapsto 0)| \leq \binom{n-|\alpha|}{2} + 2\,,$$

because flipping one bit can increase/decrease the number of satisfied clauses in the first CNF G by at most 2. On the other hand, each clause $C \in \mathrm{Lost}(1 \mapsto 0)$ has the form $C = \neg x_u \vee x_i \vee x_v$ where u and v are such that $\alpha_u = 1$ and $\alpha_v = 0$. The number of such clauses is

$$|\mathrm{Lost}(1 \mapsto 0)| \geq (|\alpha|-1)(n-|\alpha|) - 2 > \binom{n-|\alpha|}{2} + 2 \geq |\mathrm{Won}(1 \mapsto 0)|\,.$$

This completes the proof of the first claim (a). The proof of the second claim (b) is similar. □

[1] Here $|\alpha|$ stands for the number of 1's in α.

By Claim 18.17, the algorithm can find the (unique) satisfying assignment $\mathbf{0}$ for F only if it starts with an initial assignment with at most $t = n/3 + c$ 1's. Since the fraction of such assignments is $2^{-\Omega(n)}$, the probability that the algorithm finds this satisfying assignment in one iteration is also $2^{-\Omega(n)}$. So, $2^{\Omega(n)}$ iterations are necessary to achieve a constant probability error. □

18.5. Geometric resolution: cutting plane proofs

A proof system strengthening resolution is the *cutting plane proof system*. It originated in works on integer programming by Gomory (1963) and Chvátal (1973). As a proof system it was first considered in Cook, Coullard and Túran (1987). The basic idea is to prove using a few elementary rules that a system of linear inequalities with integer coefficients does not have a 0-1 solution.

Let M be a matrix with integer entries, and $\boldsymbol{b}$ an integer vector. Say that the system $M\boldsymbol{x} \geq \boldsymbol{b}$ of inequalities is *unsatisfiable* if it has no solution $\boldsymbol{x} \in \{0,1\}^n$. That is, instead of unsatisfiable CNF now we have an unsatisfiable system of linear inequalities, and our goal is to prove this (unsatisfiability) using as few applications of rules as possible. Note that a system $M\boldsymbol{x} \geq \boldsymbol{b}$ may be unsatisfiable even if it is feasable, that is, has *real values* solutions $\boldsymbol{x}$: we are interested in 0-1 solutions.

Each formula in such a proof is an inequality of the form $f(x) \geq A$, where $f(x) = \sum_i a_i x_i$, and a_i and A are integers. Since we want to prove nonexistence of 0-1 solutions, besides the axioms arising from $M\boldsymbol{x} \geq \boldsymbol{b}$, there are standard axioms $x_i \geq 0$ and $x_i \leq 1$. In order to keep the same "direction" of inequalities, we use that fact that $f \leq g$ is equivalent to $-f \geq -g$. That is, besides the axioms given by $M\boldsymbol{x} \geq \boldsymbol{b}$ we have *integrity axioms* for each variable: $x_i \geq 0$ and $-x_i \geq -1$. There are three rules of derivation:

a. *Addition* of two inequalities:

$$\frac{f(x) \geq A \quad g(x) \geq B}{f(x) + g(x) \geq A + B}.$$

b. *Multiplication* by a non-negative integer constant:

$$\frac{f(x) \geq A}{cf(x) \geq cA} \qquad (c \text{ non-negative integer}).$$

c. *Division* by a positive integer $c \geq 1$ with rounding:

$$\frac{cf(x) \geq A}{f(x) \geq \left\lceil \frac{A}{c} \right\rceil} \quad \text{and} \quad \frac{cf(x) \leq A}{f(x) \leq \left\lfloor \frac{A}{c} \right\rfloor}$$

where $\lceil \alpha \rceil = \min\{m \in \mathbb{Z} \mid \alpha \leq m\}$ and $\lfloor \alpha \rfloor = \max\{m \in \mathbb{Z} \mid m \leq \alpha\}$.

Given a system $M\boldsymbol{x} \geq \boldsymbol{b}$ with no 0-1 solution $\boldsymbol{x}$, the goal of a CP proof ("CP" stands for "cutting planes") is to derive a contradiction, represented as

$$0 \geq 1\,.$$

The first two rules are "innocent"—the whole power of cutting plane proof system comes from the third rule, because of rounding. Important in this rule is that the coefficients a_i are integers—otherwise this rule would be not valid.

Let us first show that cutting plane proof are not less efficient than resolution. First, we replace each clause by an inequality using the translation $x_i \mapsto x_i$ and $\neg x_i \mapsto 1 - x_i$. For example, the clause $x \vee \neg y$ translates to an inequality $x + (1 - y) \geq 1$, which is the same as $x - y \geq 0$. Then an assignment $a = (a_1, a_2) \in \{0,1\}^2$ satisfies the

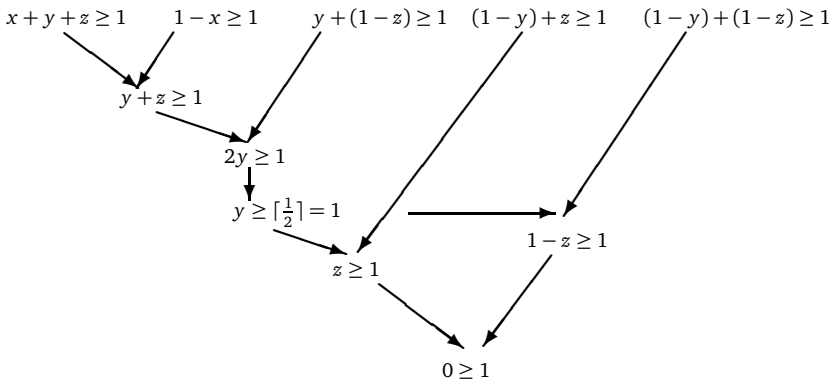


FIGURE 6. A cutting plane proof.

clause iff $a_1 - a_2 \geq 0$, that is, iff either $a_1 = 1$ or $a_2 = 0$. In this way each unsatisfiable CNF translates to an unsatisfiable system of linear inequalities. For example, the CNF

$$(x \vee y)(\neg x \vee y)(x \vee \neg y)(\neg x \vee \neg y)$$

translates to the system

$$x + y \geq 1,\ (1-x) + y \geq 1,\ x + (1-y) \geq 1,\ (1-x) + (1-y) \geq 1$$

or simpler,

$$x + y \geq 1,\ -x + y \geq 0,\ x - y \geq 0,\ -x - y \geq -1.$$

More generally, each clause C translates to the inequality

$$\sum_i a_i x_i \geq 1 - m,$$

where m is the number of negated literals in C, and

$$a_i = \begin{cases} 1 & \text{if } x_i \in C, \\ -1 & \text{if } \neg x_i \in C, \\ 0 & \text{if neither in } C. \end{cases}$$

PROPOSITION 18.18. *The cutting planes proof system can efficiently simulate resolution.*

PROOF. Suppose we have a resolution refutation proof $\mathscr{R}$ of some unsatisfiable CNF. By adding a trivial derivation rule "derive $C \vee z$ from C", we can assume that each resolution inference in this proof has the form

$$\frac{C \vee x_i \qquad C \vee \overline{x}_i}{C}.$$

Let $f = \sum_j a_j x_j \geq 1 - m$ be the inequality corresponding to clause C; here m is the number of negated literals in C. Then the inequality for the clause $C \vee x_i$ is $f + x_i \geq 1 - m$ (x_i comes positive in this clause), and the inequality for the clause $C \vee \neg x_i$ is $f - x_i \geq 1 - m - 1$. Now apply the sum-rule

$$\frac{f + x_i \geq 1 - m \qquad f - x_i \geq 1 - m - 1}{2f \geq 2 - 2m - 1},$$

and then the division rule[2]

$$\frac{2f \geq 2 - 2m - 1}{f \geq 1 - m}$$

to obtain the inequality for the clause C. □

We will now show that, in fact, cutting plane proofs for some CNFs may be even exponentially shorter that resolution proofs. So, proving lower bounds for the former model is a more difficult task.

18.5.1. Tree-like CP proofs. In a general CP proof one derived inequality can be used many times without re-deriving it. That is, the underlying graph of a derivation here may be an arbitrary directed acyclic graph. A *tree-like* CP proof is a special case of a CP proof, where the underlying graph is a tree. That is, every inequality in the proof, except for the initial inequalities, is used at most once as an antecedent of an implication.

Although restricted, tree-like CP proofs are still powerful. So, for example, we already know that the CNF formula PHP_n^{n+1} formalizing the pigeonhole principle has no resolution proof of polynomial length. On the other hand, tree-like CP proof for this CNF is relatively short.

THEOREM 18.19. *PHP_n^m has a tree-like cutting plane proof of polynomial size.*

PROOF. When translated to the language of inequalities, the axioms for the pigeonhole principle PHP_n^m consist of the following inequalities:

a. Pigeon axioms: $x_{i1} + x_{i2} + \cdots + x_{in} \geq 1$ for all $i = 1, \ldots, m$.
b. Hole axioms: $x_{ij} + x_{kj} \leq 1$ for all $1 \leq i < k \leq m$, $j = 1, \ldots, n$.
c. Integrity axioms: $x_{ij} \geq 0$; $x_{ij} \leq 1$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$.

For each j we first derive $x_{1j} + x_{2j} + \cdots + x_{mj} \leq 1$ inductively. The inequality $x_{1j} \leq 1$ is an integrity axiom, and inequality $x_{1j} + x_{2j} \leq 1$ is a hole axiom. For k from 3 to m, suppose we have already derived $x_{1j} + x_{2j} + \cdots + x_{(k-1)j} \leq 1$. We can then derive the inequality with $k-1$ replaced by k as follows.

- Multiply $x_{1j} + x_{2j} + \cdots + x_{(k-1)j} \leq 1$ by $k-2$ and add to the result the hole axioms $x_{ij} + x_{kj} \leq 1$, $i = 1, \ldots, k-1$ to get

$$(k-1)x_{1j} + (k-1)x_{2j} + \cdots + (k-1)x_{kj} \leq 2k - 3\,.$$

a. Apply division rule to get

$$x_{1j} + x_{2j} + \cdots + x_{kj} \leq \left\lfloor \frac{2k-3}{k-1} \right\rfloor = \left\lfloor 2 - \frac{1}{k-1} \right\rfloor = 1\,.$$

Summing these inequalities $x_{1j} + x_{2j} + \cdots + x_{mj} \leq 1$ over all holes j gives that the sum S of all variables is at most n, that is, $-S \geq -n$. On the other hand, summing pigeon inequalities $x_{i1} + x_{i2} + \cdots + x_{in} \geq 1$ over all pigeons i gives that $S \geq m$. Summing these two last inequalities gives $0 \geq m - n \geq 1$, the desired contradiction. □

We are now going to prove an exponential lower bound on the size of tree-like CP proofs using communication complexity arguments.

Let $M\boldsymbol{x} \geq \boldsymbol{b}$ be an unsatisfiable system of inequalities. As in the case of CNFs, every proof of the unsatisfiability of the system $M\boldsymbol{x} \geq \boldsymbol{b}$ can be viewed as an algorithm for the following search problem: Given an assignment $\alpha \in \{0,1\}^n$ find an axiom (an

[2] $\lceil -1/2 \rceil = 0$

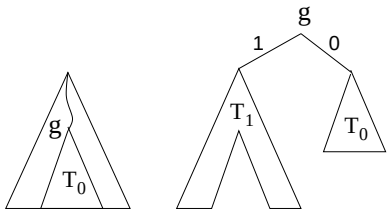


FIGURE 7. The original tree-like CP proof (left), and the obtained threshold tree (right).

inequality in our system) which is not satisfied by α. To see this, just traverse the proof starting from its last inequality $0 \geq 1$ until a "hurt" axiom is found.

Inequalities in a CP proof are not arbitrary: they are derived from axioms using the cutting planes rules. We now relax this and allow *arbitrary* inequalities to be used in the search problem. This gives rise to so-called "threshold trees."

By a *threshold function* we will now mean a boolean function which, on input vector $\boldsymbol{x} \in \{0,1\}^n$, outputs 1 iff the vector $\boldsymbol{x}$ satisfies the inequality $\sum_{i=1}^{n} a_i x_i \geq A$, where the threshold A as well as the weights a_i are arbitrary integers. A *threshold tree* for an unsatisfiable system $M\boldsymbol{x} \geq \boldsymbol{b}$ is a decision tree whose leaves are labeled by inequalities of the system. At each inner node, a decision (where to branch) is made using to an arbitrary threshold function.

LEMMA 18.20. *If an unsatisfiable system $M\boldsymbol{x} \geq \boldsymbol{b}$ has a tree-like cutting planes proof of size S, then the search problem for this system can be solved by a threshold tree of depth at most $1 + \log_2 S$.*

PROOF. Take an arbitrary tree-like cutting plane proof of size S for the system $M\boldsymbol{x} \geq \boldsymbol{b}$, and let T be its underlying tree. We argue by induction on S.

If $S = 1$ then the system consist of a single unsatisfiable inequality, and we can take it as the only inequality in our threshold tree.

For the induction step, assume that the lemma is true for all decision trees of size smaller than S.

By Claim 2.2, there must be a subtree T_0 of T rooted in some node v and such that $S/3 \leq |T_0| \leq 2S/3$. Cut off this subtree T_0 from the entire proof T, assign its root (now a leaf) the inequality $1 \geq 1$, and let T_1 be the resulting decision tree. Let also $f(x) \geq A$ be the linear inequality associated with the root of T_0 in the original proof T. We now can construct a new threshold tree as follows. We first query the function $g(x) = 1$ iff $f(x) \geq A$. If g evaluates to 0, we proceed on the subtree T_0; otherwise we proceed on the subtree T_1 (see Fig. 7). By the induction hypothesis, since both T_0 and T_1 have size at most $2S/3$, the depth of the decision tree obtained will be $1 + \log_2(2S/3) \leq 1 + \log_2 S$.

To see that the new tree solves the search problem for $M\boldsymbol{x} \geq \boldsymbol{b}$, take an assignment $\alpha \in \{0,1\}^n$. If $g(\alpha) = 0$ then we proceed on the subproof corresponding to the subtree T_0. Since the proof is sound, and the root inequality of T_0 is false on α, this implies that one of the leaf inequalities must be falsified by α. If $g(\alpha) = 1$ then we proceed on the subproof corresponding to the subtree T_1. Again since the root inequality $1 \geq 0$ of T_1 is false, one of the leaf inequalities of the original proof T must be falsified by α. This inequality cannot lie in the removed subtree T_0 because its root inequality g as well as

the inequality $1 \geq 1$ are satisfied by α. So, α must falsify some other leaf inequality of T_1. □

Let $M\boldsymbol{x} \geq \boldsymbol{b}$ be an unsatisfiable system of inequalities, and let I, J be a partition of the index set $\{1, \ldots, n\}$ of its variables $x_1, \ldots, x_n$. By a communication game for $M\boldsymbol{x} \geq \boldsymbol{b}$ under this partition we will mean a Karchmer–Wigderson type game where, for an assignment $\alpha \in \{0,1\}^n$, Alice gets its projection onto I, Bob gets the projection onto J, and their goal is to find an inequality falsified by α.

LEMMA 18.21. *If the search problem for $M\boldsymbol{x} \geq \boldsymbol{b}$ has a threshold tree of depth d where all threshold functions have polynomial-sized weight, then there exists a communication protocol for this problem where $O(d \log n)$ bits are sent.*

PROOF. Let $a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq t$ be the first inequality queried at the root of the threshold decision tree for $M\boldsymbol{x} \geq \boldsymbol{b}$, and let $g(x)$ be the corresponding threshold function. Then $g(x)$ can be written as $A(x) \geq B(x)$, where $A(x) = \sum_{i \in I} a_i x_i$ and $B(x) = t - \sum_{i \in J} a_i x_i$. Alice first communicates the value of $A(x)$ to Bob; this only requires $O(\log n)$ many bits (assuming polynomial-sized weights). Bob then completes the computation of $g(x)$, and sends the value $g(x)$ to Alice. The two players then continue on the half of the decision tree which agrees with the value of $g(x)$. The protocol terminates after d rounds, and each round only requires $O(\log n)$ bits of communication. □

In Section 8.4 we have considered the following communication game on a complete graph with $n = 3m$ vertices:

MATCH$_n$: Alice gets a matching p consisting of m edges and Bob gets an $(m-1)$-element set q of vertices. Find an edge e such that $e \in p$ and $e \cap q = \emptyset$.

We proved (Theorem 8.12) that any deterministic communication protocol for this game requires $\Omega(n)$ bits of communication. We now will turn this "search an edge" problem into a search problem for an unsatisfiable CNF formula $Match_n$ with $O(n^2)$ variables and $O(n^4)$ clauses.

Assume for a moment that we already have such a CNF formula $Match_n$. Then the communication complexity of the corresponding to $Match_n$ search problem is $\Omega(n)$. By Lemma 18.21, if a threshold tree T only has threshold functions of polynomial-sized weight and solves the search problem for the corresponding to $Match_n$ system of inequalities $M\boldsymbol{x} \geq \boldsymbol{b}$, then T must have depth $d = \Omega(n/\log n)$. By Lemma 18.20, this means that any tree-like CP proof for $M\boldsymbol{x} \geq \boldsymbol{b}$, all coefficients in which are polynomial in n, must have exponential size $2^{\Omega(n/\log n)}$.

To describe the desired CNF formula $Match_n$, we encode each m-matching $p = \{e_1, \ldots, e_m\}$ by an $m \times (3m)$ matrix of variables $X = (x_{ij})$, where $x_{ij} = 1$ iff $j \in e_i$. Each $(m-1)$-element subset $q = \{v_1, \ldots, v_{m-1}\}$ of $[3m]$ is encoded by an $(m-1) \times (3m)$ matrix $Y = (y_{ij})$, where $y_{ij} = 1$ iff $v_i = j$. That is, the ith row of X specifies the ith pair in the matching p, whereas the ith row of Y specifies the ith vertex in the set q. The CNF formula $Match_n$ consists of three CNFs:

a. $F_1(X)$ is satisfied iff p is an m-matching: every row of X has two 1's, and every column has at most one 1.
b. $F_2(Y)$ is satisfied iff q is an $(m-1)$-subset of $[3m]$: every row of of Y has exactly one 1, and and every column has at most one 1.

c. $F_3(X,Y)$ is satisfied iff every edge in p has at least one endpoint in q:

$$\neg x_{ki} \vee \neg x_{kj} \vee \bigvee_{\ell=1}^{m-1} y_{\ell i} \vee \bigvee_{\ell=1}^{m-1} y_{\ell j} \qquad \text{for all } 1 \le k \le m \text{ and } 1 \le i \ne j \le 3m.$$

By what was said, we have proved the following

THEOREM 18.22. *Any tree-like CP proof for $Match_n$, all coefficients in which are polynomial in n, must have size $2^{\Omega(n/\log n)}$.*

In fact, a similar lower bound $2^{\Omega(n/\log^3 n)}$ for $Match_n$ also holds without any restrictions on the size of coefficients used in a CP proof—being tree-like is the only restriction. For this, it is enough to observe that Theorem 8.12 about the deterministic communication complexity of the game $MATCH_n$ can be extended to *randomized* protocols: $R_{1/n}(MATCH_n) = \Omega(n/\log n)$. It remains then to combine this lower bound with the following "randomized" version of Lemma 18.21.

LEMMA 18.23. *If the search problem for $M\boldsymbol{x} \ge \boldsymbol{b}$ has a threshold tree of depth d, then there exists a randomized communication protocol for this problem where $O(d \log^2 n)$ bits are sent.*

PROOF. It is enough to use two facts about threshold functions. The first (classical) fact is that any threshold function in n variables can be computed as a threshold function with weights at most $2^{O(n \log n)}$ (see [**115**], Theorem 9.3.2.1). The second fact is that $R_{1/n}(GT_n) = O(\log^2 n)$, where $GT_n(x,y)$ is the grater than function on two n-bit integers which outputs 1 iff $x \ge y$ (see Exercise 7.5). The rest is the same as in the proof of Lemma 18.21. □

18.5.2. Arbitrary CP proofs. The lower bound above only holds for *tree-like* CP proofs: we have no analogon of Lemma 18.20 for *general* CP proofs. When trying to prove lower bounds for the length of (=number of inequalities in) general cutting plane proofs, an interesting connection with monotone circuits was detected. The connection is via so-called "interpolation theorem" in logics.

Namely, suppose that our system of inequalities has the form $F = A(x,y) \wedge B(y,z)$, where the inequalities in $A(x,y)$ do not have z-variables, and those in $B(y,z)$ do not have x-variables. If F is unsatisfiable, then an *interpolant* of F is a function $C(y)$ (on the common variables) such that for any truth assignment α to the y-variables,

a. $C(\alpha) = 0$ implies $A(x,\alpha)$ is unsatisfiable, and
b. $C(\alpha) = 1$ implies $B(\alpha,z)$ is unsatisfiable.

That is, given any assignment α to y-variables, the interpolant says which one of $A(x,\alpha)$ and $B(\alpha,z)$ is unsatisfiable. Note that at least one of them must be unsatisfiable, for otherwise the whole system F would be satisfiable: inequalities in $A(x,\alpha)$ and $B(\alpha,z)$ have no variables in common.

This gives us the following *decision* problem for $F = A(x,y) \wedge B(y,z)$: Given a truth assignment α to the y-variables, decide whether $A(x,\alpha)$ is unsatisfiable. We call such an algorithm an *interpolating algorithm* for F.

LEMMA 18.24. *Every cutting plane proof for F of size ℓ gives an interpolating algorithm for F running in time polynomial in ℓ.*

PROOF. Take a cutting plane proof for F. The idea is, given an assignment α to the common y-variables, to split the proof so that we get a refutation either from

x-axioms $A(x,\alpha)$ or from z-axioms $B(\alpha,z)$. The only rule which can mix x-variables and z-variables in the original proof is the addition of two inequalities, yielding an inequality

$$f(x)+g(y)+h(z)\geq D\,.$$

The strategy is (after the assignment $y\mapsto\alpha$) not to perform this rule in such a case and keep two inequalities

$$f(x)\geq D_0 \quad\text{and}\quad h(z)\geq D_1\,,$$

where D_0,D_1 are integers. The sums $f(x)$ and $g(z)$ may be empty, in which case they are treated as 0. What we need is only to ensure that this pair of inequalities is at least as strong as the original inequality after the assignment α, which means that we need to ensure the property:

$$D_0+D_1\geq D-g(\alpha)\,. \tag{18.4}$$

To achieve this, the axioms are replaced (after the assignment $y\mapsto\alpha$) by pairs of inequalities as follows:

$$\begin{array}{lllllll} f(x)+g(y)\geq a & \text{by pair} & f(x)\geq a-g(\alpha) & \text{and} & 0\geq 0\,; \\ g(y)+h(z)\geq b & \text{by pair} & 0\geq 0 & \text{and} & h(z)\geq b-g(\alpha)\,. \end{array}$$

The addition rule is simulated by performing additions of the first inequalities from pairs and the second inequalities in the pairs in parallel. This clearly preserves the property (18.4) we need. The multiplication rule is simulated in a similar way.

But what about the division rule? We perform this rule also in parallel on the two inequalities in the pair. The divisibility condition is clearly satisfied, as we have the same coefficients at variables x and z as in the original inequality. The only "sorrow" is therefore to make sure that the property (18.4) is preserved under rounding. For this, look at an inequality $c\cdot f(x)+c\cdot h(z)\geq D$ in the proof after assignment $y\mapsto\alpha$. By inductive assumption, we have the following inferences:

$$\frac{c\cdot f(x)\geq D_0}{f(x)\geq\lceil D_0/c\rceil} \quad\text{and}\quad \frac{c\cdot h(z)\geq D_1}{h(z)\geq\lceil D_1/c\rceil}\,.$$

Write $D_i/c=d_i+\delta_i$, where $d_i\in\mathbb{Z}$ and $\delta_i\in[0,1)$. Then

$$\left\lceil\frac{D_0}{c}\right\rceil+\left\lceil\frac{D_1}{c}\right\rceil=(d_0+\lceil\delta_0\rceil)+(d_1+\lceil\delta_0\rceil)=d_0+d_1+\lceil\delta_0\rceil+\lceil\delta_1\rceil\geq\left\lceil\frac{D_0+D_1}{c}\right\rceil\,.$$

This implies that for a positive integer c which divides all the coefficients of $g(y)$, we have that

$$D_0+D_1\geq D-g(\alpha)\quad\text{implies}\quad\left\lceil\frac{D_0}{c}\right\rceil+\left\lceil\frac{D_1}{c}\right\rceil\geq\left\lceil\frac{D}{c}\right\rceil-\frac{g(\alpha)}{c}\,.$$

Consider now the pair corresponding to the final inequality $0\geq 1$. It is of the form $0\geq D_0$, $0\geq D_1$ where $D_0+D_1\geq 1$. Since D_0 and D_1 are integers, this implies that either $D_0\geq 1$ or $D_1\geq 1$. Thus we have a proof of contradiction either from $A(x,\alpha)$ or from $B(\alpha,z)$. To know which one is the case, the algorithm may just test whether "$D_0\geq 1$" or not. By only looking at the first inequality in each pair, the CP proof gives us an algorithm $\mathscr{A}$ which, given an assignment α to y-variables, compute a number $\mathscr{A}(\alpha)=D_0$ such that $D_0\geq 1$ implies $A(x,\alpha)$ is unsatisfiable, and $D_0\leq 0$ implies $B(\alpha,z)$ is unsatisfiable. □

Having an interpolating algorithm we can turn it into a sequence of boolean circuits. Thus, if any interpolating circuit for F must be large, then F cannot have small cutting plane proofs. This is the main idea of relating proofs with circuits. Of course, so as it is, this idea is of little help: no nonlinear lower bound for circuits is known. An intriguing aspect, however, is that under some mild conditions on the form of inequalities in F, the circuits can be made *monotone*: we only have to allow monotone real-valued functions as gates.

Namely, say that a system $A(x,y) \wedge B(y,z)$ of linear inequalities is *separated* if all y-variables appear in all inequalities of $A(x,y)$ with nonnegative coefficients, or all appear with nonpositive coefficients in $B(y,z)$.

THEOREM 18.25. *If an unsatisfiable system F of linear inequalities is separated then it has an interpolating monotone real circuit of size polynomial in the minimal cutting plane proof size of F.*

PROOF. It is enough to turn the algorithm from Lemma 18.24 into a monotone real circuit whose size is polynomial in the size of the underlying cutting proof of F. Let us first realize that we only need to compute the constant D_0 (or only D_1) corresponding to the last inequality. We shall assume w.l.o.g. that all y-variables appear in all inequalities of $A(x,y)$ with nonnegative coefficients.

Recall that, in each step, we replace an inequality $f(x) + g(\alpha) \geq a$ by $f(x) \geq D_0$ with $D_0 = a - g(\alpha)$. Since the coefficients of y-variables in $A(x,y)$ are nonnegative, it is more convenient to talk about $-D_0 = g(\alpha) - a$; then we do not need to multiply $g(\alpha)$ by a negative constant -1.

Thus, we only need to compute successively $-D_0$ for each pair. For this, we can use the graph of the circuit for constructing a circuit. Each gate will produce a new $-D_0$ from previous ones. The circuit has 0 and 1 as inputs corresponding to the truth assignment α to y-variables, but computes arbitrary integers in the inner nodes.

If $f(x) + g(y) \geq a$ is an axiom, where $g(y) = \sum c_i y_i$, then the function $\alpha \mapsto -D_0 = g(\alpha) - a$ is nondecreasing because all coefficients c_i are nonnegative, by our assumption. Hence, if $\alpha \leq \beta$ are two 0-1 vectors, then $g(\alpha) \leq g(\beta)$. Thus, operations we need are:

a. addition of an integer constant,
b. multiplication by a non-negative integer constant,
c. addition,
d. division be a positive integer constant with rounding,
e. to get a 0-1 output we add a threshold gate at the output gate, that is, the unary gate t defined by $t(\xi) = 1$ if $\xi \geq 0$ and $t(\xi) = 0$ otherwise.

All the operations are nondecreasing except for multiplication by negative constants. In general they are needed in the initial inequalities, where for $f(x) + g(y) \geq a$ we need to compute $g(\alpha) - a$. However (as we observed above), we do not need multiplication by negative constants there, since we assume that coefficients in $g(y)$ are all nonnegative. The remaining inequalities where we need multiplication by negative constants are $-y_i \geq -1$. These, however, can be treated as inequalities containing z-variables, that is, we can put $D_0 = 0$ for them. Thus we get all gates nondecreasing.

As in the proof of Theorem 18.25, for each assignment α to y-variables, the input to the last gate $t(\xi)$ is an integer $\xi = -D_0$ such that $D_0 \geq 1$ implies $A(x,\alpha)$ is unsatisfiable, and $D_0 \leq 0$ implies $B(\alpha,z)$ is unsatisfiable. Hence, $t(\xi) = 0$ implies $A(x,\alpha)$ is unsatisfiable (because then $\xi = -D_0 \leq -1$, or equivalently, $D_0 \geq 1$), and $t(\xi) = 1$

implies $B(x,\alpha)$ is unsatisfiable. Thus, the obtained circuit is indeed an interpolating circuit for $A(x,y)\wedge B(y,z)$. □

Theorem 18.25 reduces the lower bounds task for cutting planes to that for monotone real circuits. We already know (see Theorem 4.16) that every monotone boolean function f in n variables with the following two properties requires monotone real circuits of size 2^{n^ε}. Inputs of f are graphs G on n vertices, encoded by $\binom{n}{2}$ boolean variables, and

a. $f(G)=1$ if G contains a k-clique,
b. $f(G)=0$ if G is $(k-1)$-colorable.

What we need is a system of linear inequalities $A(x,y)\wedge B(y,z)$ such that any interpolant $f(y)$ for this system satisfies these two conditions. That is, we only need to write the statement

a graph contains a k-clique and is $(k-1)$-colorable

as an unsatisfiable system of linear inequalities. For this we take three groups of variables:

y-variables $y_{i,j}$ encoding the edges: $y_{i,j}=1$ iff the edge $\{i,j\}$ is present;
x-variables x_i, one for each vertex, encoding cliques;
z-variables $z_{i,c}$, one for each vertex i and color c encoding $(k-1)$-colorings: $z_{i,c}=1$ iff vertex i has color c.

We want to impose the conditions:

(i) The set of nodes $\{i \mid x_i=1\}$ forms a clique of size $\geq k$.
(ii) For all $c=1,\ldots,k-1$, the set $\{i \mid z_{i,c}=1\}$ is an independent set.

The underlying graph is given by the values of y-variables. We now describe a system of inequalities Clique(x,y) corresponding to the first condition (i), and a system of inequalities Color(y,z) corresponding to the second condition (ii).

Clique(x,y): For $V=\{i \mid x_i=1\}$ to form a clique, besides the inequality

$$\sum_i x_i \geq k$$

we also need to ensure that all nodes in V are pairwise adjacent. That is, we need that $x_i=1$ and $x_j=1$ implies $y_{i,j}=1$. This can be written as an inequality

$$y_{i,j}-x_i-x_j\geq -1\,.$$

Color(y,z): For the sets $I=\{i \mid z_{i,c}=1\}$ to be independent sets (color classes), we first need that each vertex i receives exactly one color:

$$\sum_c z_{i,c}=1$$

and that no two adjacent vertices $i\neq j$ receive the same color. This last condition means that $z_{i,c}=1$ and $z_{j,c}=1$ must imply $y_{i,j}=0$, and this can be written as an inequality

$$-y_{i,j}-z_{i,c}-z_{j,c}\geq -2\,.$$

Note that the y-variables occur in Clique(x,y) only with positive signs, and occur with negative signs in Color(y,z). Hence, the system of inequalities $F=A(x,y)\wedge B(y,z)$ is separated. By Theorem 18.25, this system has an interpolating monotone real circuit $C(y)$ of size polynomial in the minimal CP proof size of F.

Let us look at what this circuit $C(y)$ does. For every assignment $\alpha \in \{0,1\}^{\binom{n}{2}}$ to y-variables, Clique(x,α) is satisfiable if the graph G_α encoded by α contains a k-clique, whereas Color(α,z) is satisfiable if the graph G_α is colorable by $k-1$ colors. Hence, $C(\alpha)=1$ if G_α has a k-clique, and $C(\alpha)=0$ if the graph G_α is $(k-1)$-colorable. By Theorem 4.13, we know that, for $k=\Theta(\sqrt{n})$, the circuit $C(y)$ must have size 2^{n^ε} for a constant $\varepsilon>0$. This gives us

COROLLARY 18.26. *Any cutting plane derivation of the contradiction* $0 \geq 1$ *from* Clique$(x,y) \wedge$ Color(y,z) *has size at least* 2^{n^ε}.

This proof is not quite satisfying—it is not as "combinatorial" as that for resolution refutations. It would be nice to find a lower bounds argument for cutting plane proofs, *explicitly* showing what properties of inequalities do force long derivations.

RESEARCH PROBLEM 18.27. *Find a combinatorial lower bounds argument for cutting plane proofs.*

18.6. Addendum: Arbitrary number of pigeons

We have already seen how to prove super-polynomial lower bound on the size of resolution refutations of PHP_n^m for up to $m \ll n^2/\log n$ pigeons (see Section 18.3.2). However, the larger m is, the more true the pigeonhole principle itself is, and it could be that PHP_n^m could be refuted by much shorter resolution refutation proof. And indeed, all attempts to overcome this "n^2 barrier" for the number of pigeons failed for many years. This was one of most famous open problems in the propositional proof complexity.

The "n^2 barrier" for PHP_n^m was broken by using a more subtle concept of "pseudo-width" of clauses, tailor made for this particular CNF formula. It turns out that PHP_n^m requires resolution proofs of exponential size for *any* number $m \geq n+1$ of pigeons!

THEOREM 18.28. *For every* $m \geq n+1$*, every resolution refutation proof of* PHP_n^m *has length at least* $2^{\Omega(n^{1/4})}$.

Recall that PHP_n^m denotes the AND of the following clauses (we call them *axioms*):

a. Pigeon Axioms: each of the m pigeon sits in at least one of n holes:
$$x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n} \text{ for all } i=1,\ldots,m.$$
b. Hole Axioms: no two pigeons sit is one hole:
$$\neg x_{i_1,j} \vee \neg x_{i_2,j} \text{ for all } i_1 \neq i_2 \text{ and } j=1,\ldots,n.$$

A special feature of this CNF is that any resolution refutation of the set all its axioms (clauses) can be transformed to a monotone refutation of its pigeon axioms without any increase in size of a derivation. To define monotone refutations, let $X_{i,j}$ be the OR of all but the ith variable in the jth column:
$$X_{i,j} = x_{1,j} \vee \cdots \vee x_{i-1,j} \vee x_{i+1,j} \vee \cdots \vee x_{m,j}\,.$$
By a *monotone refutation* of PHP_n^m we will mean a derivation of an empty clause from pigeon axioms and using the following *monotone resolution rule*:
$$\frac{A \vee x_{i,j} \qquad B \vee X_{i,j}}{A \vee B}\,.$$
Such a derivation can be obtained from the original (non-monotone) derivation by replacing each negated variable $\neg x_{i,j}$ by the OR of variables $X_{i,j}$. Note that, in general,

this rule *is not* sound: there are assignments satisfying both assumptions but falsifying the conclusion. Still, the rule *is* sound if we consider only assignments satisfying all hole axioms; we call such assignments *legal*. That is, an assignment α is legal if it sends no two pigeons to the same hole (no column has more than one 1).

The following fact reduces the lower bounds problem for PHP_n^m to its monotone version.

LEMMA 18.29. *If PHP_n^m has a resolution refutation of size ℓ, then PHP_n^m also has a monotone refutation of size at most ℓ.*

PROOF. Given a resolution refutation proof for PHP_n^m, just replace all occurrences of a negated variable $\neg x_{i,j}$ by the OR $X_{i,j} = \bigvee_{k\neq i} x_{k,j}$. It can be shown (do this!) that the resulting sequence of monotone clauses is a monotone refutation of the pigeon axioms. □

For the proof in the case when m is arbitrarily large, it will be convenient to increase the power of refutations by allowing a larger set of monotone derivation rules:

$$\frac{C_0 \vee X_{I_0,j} \qquad C_1 \vee X_{I_1,j}}{C},$$

where $X_{I,j} = \bigvee_{i\in I} x_{i,j}$, $I_0 \cap I_1 = \emptyset$ and $C_0 \vee C_1 \leq C$. From now on, by a *monotone refutation* of PHP_n^m we will mean a refutation of pigeon axioms using any of these rules. Note that these rules are still sound with respect to all legal truth assignments, that is, assignments sending no two pigeons to one hole: if such an assignment satisfies both clauses $C_0 \vee X_{I_0,j}$ and $C_1 \vee X_{I_1,j}$ then, due to the condition $I_0 \cap I_1 = \emptyset$, it must also satisfy at least one of the clauses C_0 or C_1, and hence, the clause C as well.

18.6.1. Size versus pseudo-width of refutations. Suppose we have a monotone refutation proof $\mathscr{R}$ of the pigeon axioms

$$X_{i,[n]} = \bigvee_{j=1}^{n} x_{i,j}, \quad i = 1,\ldots,m.$$

To analyze the refutation $\mathscr{R}$, we are going to allow much more axioms. For this we fix two parameters. First set

$$\delta := \frac{n}{2\log_2 m}.$$

A *threshold string* is a string $d = (d_1,\ldots,d_m)$ of positive integers with $\delta < d_i \leq n$ for all i. Having such a string d, we will allow all clauses of the form

$$X_{i,J} = \bigvee_{j\in J}^{n} x_{i,j} \quad \text{with} \quad i \in [m] \quad \text{and} \quad |J| \geq d_i$$

be used as axioms; we call such axioms *d-axioms*. Note that every monotone refutation of PHP_n^m is a monotone refutation of the set of d-axioms for the threshold string $d = (n,\ldots,n)$.

Allowing more axioms does not hurt us, since our goal is to prove a *lower* bound on the size of a refutation. The reason for introducing new axioms is that we can then "filter out" from the refutation proof all clauses containing at least one such axiom: we just replace each such clause by the corresponding axiom.

For this purpose we consider the *degree of freedom* $d_i(C)$ of each pigeon i in a clause C: this is the number of holes offered by C to this pigeon, that is, the number of holes j such that the variable $x_{i,j}$ (ith pigeon sits in the jth hole) belongs to C:

$$d_i(C) = \Big|\{j : x_{i,j} \in C\}\Big| .$$

The clause C is "filtered out" from the proof (i.e. can be replaced by an axiom) if $d_i(C) \geq d_i$ for at least one pigeon i.

The main concept of our analysis will be the following very special notion of the "width" of refutation proofs for PHP_n^m, tailor made for this particular set of CNFs Namely, define the *pseudo-width* $w_d(C)$ of a clause as the number

$$w_d(C) = \Big|\{i : d_i(C) \geq d_i - \delta\}\Big|$$

of pigeons whose degree of freedom in this clause is large. The pseudo-width of a refutation $\mathscr{R}$ is the maximum pseudo-width of a clause in it.

Our first task (Lemma 18.30 below) will be to show that if the thresholds d_i are chosen in a clever way, then in every clause $C \in \mathscr{R}$ passing the filter—that is, having $d_i(C) < d_i$ for all pigeons i—almost all, namely, at least $m - O(\log|\mathscr{R}|)$ pigeons pass it *safely*: their "degree of freedom" in C is well below the corresponding threshold d_i, is $\leq d_i - \delta$. Thus, the number of pigeons who *narrowly* (= non-safely) pass the filter $(d_1, \ldots, d_m)$ must be at most $O(\log|\mathscr{R}|)$.

LEMMA 18.30 (Short proofs have small pseudo-width). *If PHP_n^m has a resolution refutation of size S then there exists a threshold string d such that some set of at most S d-axioms has a monotone refutation of size S and pseudo-width $O(\log S)$.*

PROOF. To prove the lemma, we have to somehow "filter out" clauses of large pseudo-width. For this we need the following combinatorial lemma; we will give its proof later in Section 18.6.2.

LEMMA 18.31 (Pigeon filter lemma). *Let $R = \{r_{i,k}\}$ be an $m \times S$ matrix with integer entries. If S is sufficiently large, then there exists a sequence $r_1, \ldots, r_m$ of integers such that $r_i < \lfloor \log m \rfloor$ and for every column k at least one of the following two events happen:*

(i) *$r_{i,k} \leq r_i$ for at least one row i;*
(ii) *$r_{i,k} \leq r_i + 1$ for at most $O(\log S)$ rows i.*

Suppose now that PHP_n^m has a resolution refutation of size S. Then, by Lemma 18.29, the set of all m pigeon axioms has a monotone refutation of size S. Fix such a refutation $\mathscr{R}$ an consider an $m \times S$ matrix $R = \{r_{i,C}\}$ whose rows correspond to pigeons $i \in [m]$ and columns to clauses C of this refutation. Define the entries of this matrix by

$$r_{i,C} := \left\lfloor \frac{n - d_i(C)}{\delta} \right\rfloor + 1.$$

Let $r_1, \ldots, r_m$ be a sequence of integers guaranteed by Lemma 18.31. Set

$$d_i := \lfloor n - \delta r_i \rfloor + 1 ,$$

and note that $d_i > \delta$ because $r_i < \log m$. Hence, $d = (d_1, \ldots, d_m)$ is a threshold string. This special choice of the entries $r_{i,C}$ as well as of the d_i guarantee us two properties (check this!):

(iii) If $r_{i,C} \leq r_i$ then $d_i(C) \geq d_i$.
(v) If $d_i(C) \geq d_i - \delta$ then $r_{i,C} \leq r_i + 1$.

Now take an arbitrary clause $C \in \mathscr{R}$. Our goal is to show that either C contains a d-axiom (and C can be replaced by that axiom which reduces its pseudo-width $w_d(C)$ to 1) or $w_d(C) = O(\log S)$.

If the first case (i) in Lemma 18.31 takes place, then $r_{i,C} \le r_i$ for some pigeon i, and by (iii), $d_i(C) \ge d_i$. Hence, in this case C contains a subclause $X_{i,J}$ which is a d-axiom, and can be replaced by this axiom.

If the second case (ii) in Lemma 18.31 takes place, then the number of pigeons i for which $r_{i,C} \le r_i + 1$, and hence, by (v), the number of pigeons i for which $d_i(C) \ge d_i - \delta$ does not exceed $O(\log S)$. Hence, in this case the pseudo-width $w_d(C)$ of C cannot exceed $O(\log S)$. □

The second task is to show that the number of pigeons who *narrowly* passed the filter must be at least $\Omega\left(n/\log^3|\mathscr{R}|\right)$. This implies $\log|\mathscr{R}| \ge \Omega(n^{1/4})$, as desired.

LEMMA 18.32 (Pigeonhole proofs have large pseudo-width). *For every threshold string d, every monotone refutation $\mathscr{R}$ of a set of S d-axioms requires pseudo-width at least $\Omega\left(n/\log^3 S\right)$.*

This lemma, together with Lemma 18.30 and an observation that in any refutation of PHP_n^m of size S at most $m \le 2S$ pigeons can be used, implies that the minimal size S of a resolution refutation of PHP_n^m must satisfy the inequality $S \ge 2^{n^{1/4}}$ claimed in Theorem 18.28.

PROOF OF LEMMA 18.32. Let $d = (d_1, \dots, d_m)$ be an integer vector with $\delta < d_i \le n$ for all i. Take an arbitrary set $\mathscr{A}$ of $|\mathscr{A}| \le S$ d-axioms, and set

$$w_0 := \frac{\varepsilon\delta^2}{n\log|\mathscr{A}|}$$

where $\varepsilon > 0$ is a sufficiently small constant. Take an arbitrary monotone refutation $\mathscr{R}$ of $\mathscr{A}$. We will show that $w_d(C) > w_0$ for at least one clause C in $\mathscr{R}$.

Suppose the opposite, i.e., that $w_d(C) \le w_0$ for all clauses $C \in \mathscr{R}$. Our goal is to show that then the empty clause $\mathbf{0}$ does not belong to $\mathscr{R}$, i.e., that $\mathscr{R}$ is *not* a refutation of $\mathscr{A}$.

Recall that each axiom in $\mathscr{A}$ has the form $X_{i,J} := \bigvee_{j\in J} x_{i,j}$ for some pigeon i and some set J of $|J| \ge d_i$ holes; $X_{i,J}$ is the axiom for the pigeon i. Let

$$\mathscr{A}_i = \left\{X_{i,J} \in \mathscr{A} : |J| \ge d_i\right\}$$

denote the set of all such axioms in $\mathscr{A}$, and let $\mathscr{A}_I := \bigcup_{i\in I} \mathscr{A}_i$. For a clause C in $\mathscr{R}$ let

$$\mathscr{A}_C = \bigcup_{i:d_i(C)\ge d_i-\delta} \mathscr{A}_i$$

denote the set of all axioms in $\mathscr{A}$ corresponding to pigeons that are "free enough" in the clause C.

As before, truth assignments are $m \times n$ $(0,1)$ matrices α. Such an assignment is legal if it satisfies all hole axioms, that is, if no column has more than one 1. Say that an assignment α is *critical* if it is legal and no row of α has more than

$$\ell := \left\lfloor \frac{\delta}{4w_0} \right\rfloor$$

1-entries. We say that a set $\mathscr{C}$ of clauses *implies* a clause C, and write $\mathscr{C} \models C$, if every critical assignment α satisfying all clauses of $\mathscr{C}$ also satisfies C.

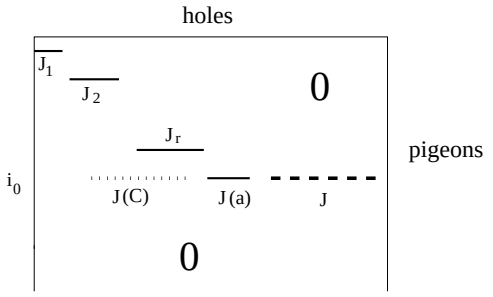


FIGURE 8. $I-\{i_0\}=\{1,\ldots,r\}$, $J_i=\{j \mid a_{i,j}=1\}$, $J(a)=\{j \mid a_{i_0,j}=1\}$ and $J(C)=\{j \mid x_{i_0,j}\in C\}$.

CLAIM 18.33. For every clause C in $\mathscr{R}$ we have that $\mathscr{A}_C \models C$.

This already gives the desired contradiction, because $\delta < d_i$ for all i implies that $\mathscr{A}_{\mathbf{0}}=\emptyset$, and hence, that $\mathscr{A}_{\mathbf{0}} \not\models \mathbf{0}$. Thus, it remains to prove the lemma.

To prove Claim 18.33, we argue by induction on the number of steps in the derivation of C in $\mathscr{R}$. The case $C\in\mathscr{A}$ is obvious since then $C\in\mathscr{A}_C$.

For the inductive step suppose that $\mathscr{A}_A \models A$, $\mathscr{A}_B \models B$ and C is obtained from clauses A,B by a single application of the monotone refutation rule. Since the rule is sound with respect to all legal (and hence, also for all critical) truth assignments, we have that $\{A,B\}\models C$. Hence, if we take the set

$$I=\{i \mid d_i(A)\geq d_i-\delta \text{ or } d_i(B)\geq d_i-\delta\}$$

of pigeons of large degree of freedom in at least one of the clauses A or B, then $|I|\leq 2w_0$ and $\mathscr{A}_I\models C$. Let us choose a minimal $I\subseteq\{1,\ldots,m\}$ such that $\mathscr{A}_I\models C$; then still $|I|\leq 2w_0$. We will show that, in fact,

$$I\subseteq\{i \mid d_i(C)\geq d_i-\delta\}\,;$$

this will obviously imply $\mathscr{A}_I\subseteq\mathscr{A}_C$, and hence, $\mathscr{A}_C\models C$.

Assume the contrary, and pick an arbitrary $i_0\in I$ with $d_i(C)<d_i-\delta$. Since I is minimal, we have that $\mathscr{A}_{I-\{i_0\}}\not\models C$. Hence, there is a critical assignment $\alpha=(a_{i,j})$ which satisfies all clauses in $\mathscr{A}_{I-\{i_0\}}$ but falsifies C. We may assume that $a_{i,j}=0$ for all $i\notin I-\{i_0\}$ and all j, because C is positive and none of such variables $x_{i,j}$ appears in $\mathscr{A}_{I-\{i_0\}}$. Let now

$$J=\left\{j \mid x_{i_0,j}\notin C \text{ and } a_{i,j}=0 \text{ for all } i\in I-\{i_0\}\right\}$$

be the set of holes "permissible" for the pigeon i_0 (see Fig. 8): if we pick an arbitrary subset $J'\subseteq J$ of size $|J'|=\ell$ and change the assignment α by letting $a_{i_0,j}=1$ iff $j\in J'$, then we will get a critical assignment α' which still satisfies all clauses in $\mathscr{A}_{I-\{i_0\}}$ (we have not touched other pigeons) but falsifies C.

We want to show that J' can be chosen in such a way that this new assignment α' will also satisfy all clauses in $\mathscr{A}_{i_0}$; this will give the desired contradiction with $\mathscr{A}_I\models C$.

First, observe that the set J is large enough: since $d_{i_0}(C)<d_{i_0}-\delta$ and each row of α has at most ℓ 1-entries, we have that

$$|J|\geq n-\big(|I|\cdot\ell+d_{i_0}(C)\big)\geq n-\big(2w_0\ell+(d_{i_0}-\delta)\big)\geq n-d_{i_0}+\delta/2\,. \tag{18.5}$$

Now pick $\mathbf{J}$ uniformly and at random among all ℓ-element subsets of J, and let $\boldsymbol{\alpha}$ be the random assignment resulting from the assignment a by setting to 1 all $a_{i_0,j}$ with $j \in \mathbf{J}$. Take an arbitrary axiom $A \in \mathscr{A}_{i_0}$, and let $J_A = \{j \mid x_{i_0,j} \in A\}$ be the set of holes offered by the clause A to the pigeon i_0. Since $|J_A| \geq d_{i_0}$, by (18.5) we have

$$|J_A \cap J| \geq \delta/2\,.$$

Now we can apply Chernoff's inequality and conclude that

$$\Pr[A(\boldsymbol{\alpha}) = 1] = \Pr[J_A \cap \mathbf{J} \neq \emptyset] \geq 1 - e^{-\Omega(p\cdot\delta)} \geq 1 - e^{-\Omega(\delta\ell/n)}.$$

Since

$$\frac{\delta\ell}{n} \geq \frac{\delta^2}{4w_0 n} = \frac{\delta^2}{4n}\cdot\frac{n\cdot\log|\mathscr{A}|}{\varepsilon\delta^2} = \frac{\log|\mathscr{A}|}{4\varepsilon},$$

we obtain that

$$\Pr[A(\boldsymbol{\alpha}) = 1] \geq 1 - |\mathscr{A}|^{-2},$$

if the constant ε is sufficiently small. Since clearly, $|\mathscr{A}_{i_0}| < |\mathscr{A}_{i_0}|^2 \leq |\mathscr{A}|^2$, this implies that, for at least one choice α' of $\boldsymbol{\alpha}$, *all* axioms in $\mathscr{A}_{i_0}$ will be satisfied. Since (as we observed above) the assignment α' also satisfies all axioms in $\mathscr{A}_{I-\{i_0\}}$ but falsifies C, we obtained a contradiction with $\mathscr{A}_I \models C$.

This completes the proof of Claim 18.33, and thus, the proof of Lemma 18.32. □

18.6.2. Proof of the pigeon filter lemma. The lemma is a direct consequence of the following property of randomly chosen numbers. Let m and S be positive integers. Set $t := \lfloor \log m \rfloor - 1$, and let $\boldsymbol{r}$ be a random variable taking its values in $[t] = \{1,\dots,t\}$ with probabilities

$$\Pr[\boldsymbol{r} = t] = 2^{-(t+1)} \quad \text{and} \quad \Pr[\boldsymbol{r} = s] = 2^{-s} \quad \text{for each} \quad s = 1,2,\dots,t-1.$$

CLAIM 18.34. Let S be a positive integer, $x = (x_1,\dots,x_m)$ an integer vector, and Let $\boldsymbol{r}_1,\dots,\boldsymbol{r}_m$ be m independent copies of $\boldsymbol{r}$. Then with probability at least $1 - O(S^{-2})$ at least one of the following two events happens:

A_x: $\boldsymbol{r}_i \geq x_i$ for at least one integer x_i;
B_x: $\boldsymbol{r}_i < x_i - 1$ for all but at most $O(\log S)$ integers x_i.

PROOF. Our goal is to show that at least one of $\Pr[\neg A_x]$ and $\Pr[\neg B_x]$ is at most $O(S^{-2})$, implying that the desired sequence $r_1,\dots,r_m$ satisfying both conditions of Lemma 18.31 exist with probability at least $1 - O(S^{-2})$.

Define the "weight" of x as $W(x) := \sum_{i=1}^{m} 2^{-x_i}$. We consider two cases depending on whether $W(x) \geq 2\ln S$ or not.

Case 1: $W(x) \geq 2\ln S$. Let $I = \{i \mid x_i \leq t\}$ and note that

$$\sum_{i\notin I} 2^{-x_i} \leq m2^{-(t+1)} \leq 2\,.$$

Therefore,

$$\sum_{i\in I} 2^{-x_i} \geq W(x) - 2 \geq 2\ln S - 2\,. \tag{18.6}$$

On the other hand, for every $i \in I$ we have $\Pr[\boldsymbol{r}_i \geq x_i] \geq 2^{-x_i}$, and these events are independent. Since $\Pr[\boldsymbol{r}_i \geq x_i] = 0$ for all $i \notin I$, we have that in this case

$$\Pr[\neg A_x] = \Pr[\forall i :\, \boldsymbol{r}_i < x_i] = \prod_{i\in I}\left(1 - 2^{-x_i}\right) \leq \exp\left(-\sum_{i\in I} 2^{-x_i}\right) \leq e^2 S^{-2}\,,$$

where the last inequality follows from (18.6).

Case 2: $W(x) \leq 2\ln S$. We first show that $\Pr[\boldsymbol{r} \geq x_i - 1] \leq 2^{2-x_i}$ for every i. Indeed, if $x_i > t$ then either $x_i = t+1$ and

$$\Pr[\boldsymbol{r} \geq x_i - 1] = \Pr[\boldsymbol{r} = t] = 2^{1-t} = 2^{2-x_i},$$

or $x_i \geq t+2$ and $\Pr[\boldsymbol{r} \geq x_i - 1] = 0$. If $x_i \leq t$ then

$$\Pr[\boldsymbol{r} \geq x_i - 1] = \sum_{s=x_i-1}^{t} 2^{-s} \leq 2^{1-x_i} \sum_{j=0}^{\infty} 2^{-j} \leq 2^{2-x_i}.$$

Hence, the expected number of i for which $\boldsymbol{r}_i \geq x_i - 1$ does not exceed

$$\sum_{i=1}^{m} 2^{2-x_i} = 4 \sum_{i=1}^{m} 2^{-x_i} = 4W(x) \leq 8\ln S.$$

Since the events "$\boldsymbol{r}_i \geq x_i - 1$" are independent, we may apply Chernoff's inequality and conclude that, for any sufficiently large constant c,

$$\Pr[\neg B_x] = \Pr[|\{i : \boldsymbol{r}_i \geq x_i - 1\}| \geq c\ln S] \leq S^{-2}. \qquad \square$$

Exercises

Ex. 18.1 (Cliques and CNFs). Given a graph $G = (V, E)$, define the CNF formula

$$F_G = \bigwedge_{\{i,j\} \notin E} (\overline{x}_i \vee \overline{x}_j).$$

Each assignment $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0,1\}^n$ can be interpreted as an incidence vector of the set of vertices $S_\alpha = \{i \mid \alpha_i = 1\}$. Show that S_α is a clique in G if and only if α satisfies the formula F_G.

Ex. 18.2. Show that Resolution is complete: every unsatisfiable CNF formula F has a resolution refutation proof. *Hint*: Show that the search problem for F can be solved by a decision tree, and use Theorem 18.1.

Ex. 18.3. Let F be a CNF formula and x a literal. Show that F is unsatisfiable if and only if both CNFs $F_{x=1}$ and $F_{x=0}$ are unsatisfiable.

Ex. 18.4. Let G be a bipartite (r,c)-expander graph. Show that then the induced CNF formula $PHP(G)$ is (r,c)-expanding.

Ex. 18.5. Show that for every constant $d \geq 5$, there exist bipartite $n \times n$ graphs of left degree d that are (r,c)-expanders for $r = n/d$ and $c = d/4 - 1$.

Hint: Construct a random graph with parts L and R, $|L| = |R| = n$, by choosing d neighbors for each vertex in L. For $S \subseteq L$ and $T \subseteq R$, let $E_{S,T}$ be the event that all neighbors of S lie within T. Argue that, $\Pr[E_{S,T}] = (|T|/n)^{d|S|}$. Let E be the event that the graph is not the desired expander, i.e., that all neighbors of some subset $S \subseteq L$ of size $|S| \leq n/d$ lie within some subset $T \subseteq R$ of size $|T| < (d/4)|S|$. Use the union bound for probabilities and the estimate $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$ to show that

$$\Pr[E] \leq \sum_{i=1}^{n/d} \left(\frac{e}{4}\right)^{ds/2}.$$

Use our assumption $d \geq 5$ together with the fact that $\sum_{i=0}^{\infty} x^i = 1/(1-x)$ for any real number x with $|x| < 1$ to conclude that $\Pr[E]$ is strictly smaller than 1.

A CNF formula F is *k-satisfiable* if any subset of its k clauses is satisfiable.

Ex. 18.6 (3-satisfiable CNFs). Given a 3-satisfiable CNF formula F in n variables, define a random assignment $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0,1\}^n$ by the following rule:

$$\Pr[\alpha_i = 1] = \begin{cases} 2/3 & \text{if } F \text{ contains a unary clause } (x_i); \\ 1/3 & \text{if } F \text{ contains a unary clause } (\overline{x}_i); \\ 1/2 & \text{otherwise.} \end{cases}$$

a. Why this definition is consistent? *Hint*: 3-satisfiability.
b. Show that $\Pr[y(\alpha) = 1] \geq 1/3$ for each literal $y \in \{x_i, \overline{x}_i\}$, which appears in the formula F (independent of whether this literal forms a unary clause or not).
c. Show that the expected number of clauses of F satisfied by α is at least a $2/3$ fraction of all clauses.

Hint: Show that each clause if satisfied by α with probability at least $2/3$. The only non-trivial case is when the clause has exactly 2 literals. Treat this case by keeping in mind that our formula is 3-satisfiable, and hence, cannot have three clauses of the form $(y \vee z)$, $(\overline{y})$ and $(\overline{z})$.

Ex. 18.7 (2-satisfiable CNFs). Prove the Lieberher-Specker result for 2-satisfiable CNF formulas: if F is a 2-satisfiable CNF formula then at least γ-fraction of its clauses are simultaneously satisfiable, where $\gamma = (\sqrt{5} - 1)/2 > 0.618$.

Hint: Define the probability of a literal y to be satisfied to be: a ($a > 1/2$) if y occurs in a unary clause, and $1/2$ otherwise. Observe that then the probability that a clause C is satisfied is a if C is a unary clause, and at least $1 - a^2$ otherwise (at worst, a clause will be a disjunction of two literals whose negations appear as unary clauses); verify that $a = 1 - a^2$ for $a = \gamma$.

Ex. 18.8. Write down *explicitely* the CNF formulas $F_1(X)$ and $F_2(Y)$ in the definition of the CNF formula $Match_n$ in Section 18.5.1.

Ex. 18.9. The probability distribution of $\boldsymbol{r}$ in Claim 18.34 is defined somewhat "artificially"? Why we could not take just $\Pr[\boldsymbol{r} = s] = 2^{-s}$ for *all* $s \leq t$? *Hint*: $\sum_{i=1}^{n} a^i = (a^{n+1} - a)/(a - 1)$.

Bibliographic Notes

Theorem 18.1 is a "folklore" observation. The exponential lower bound in Theorem 18.4 was first proved by Haken (1985). A greatly simplified proof was found by Beame and Pitassi (1996). In the proof above we followed the presentation in Beame (2000). Results of Section 18.3 are due to Ben-Sasson and Wigderson (2001). Theorem 18.15 is due to Papadimitriou (1991), and Theorem 18.16 to Hirsch (2000). Theorem 18.19 is due to Cook, Coullard and Turán (1987). Theorem 18.22 as well as its extension to tree-like CP proofs with arbitrary coefficients is due to Impagliazzo, Pitassi and Urquhart (1994). Theorem 18.25 for general CP proofs is due to Pudlák (1997); its proof is based on earlier ideas of Krajiček (1994) and Bonet, Pitassi and Raz (1997). Theorem 18.28 is due to Razborov (2003); a more involved proof was earlier given by Raz (2001).

Epilog

In this book we have learned almost all arsenal of existing lower bounds arguments. They work well for different restricted circuit classes but, so far, have not led to a non-linear lower bound for unrestricted circuits. In this concluding chapter we sketch several general results explaining this failure (the phenomenon of "natural proofs") as well as showing a possible line of further attacks (the "fusion method").

Pseudo-random generators

When trying to prove a lower bound, we try to show that something *cannot* be computed efficiently. It turned out that this task is closely related to proving that something—namely, so-called "pseudorandom generators"—*can* be efficiently computed!

Informally speaking, a pseudorandom generator is an "easy to compute" function which converts a "few" random bits to "many" pseudorandom bits that "look random" to any "small" circuit. Each one of the quoted words is in fact a parameter, and we may get pseudorandom generators of different qualities according to the choice of these parameters. For example, the standard definitions are: "easy to compute" = polynomial time; "few" $= n^{\varepsilon}$; "many" $= n$.

DEFINITION 18.35. A function $G : \{0,1\}^l \to \{0,1\}^n$ with $l < n$ is called an (s,ε)-*secure pseudorandom generator* if for any circuit C of size s on n variables,

$$\Big|\Pr[C(y)=1] - \Pr[C(G(x))=1]\Big| < \varepsilon\,,$$

where y is chosen uniformly at random in $\{0,1\}^n$, and x in $\{0,1\}^l$.

That is, a pseudorandom generator G stretches a short truly random seed x into a long string $G(x)$ which "fools" all circuits of size up to s: no such circuit can distinguish $G(x)$ from a truly random string y.

Note that the definition is only interesting when $l < n$, for otherwise the generator can simply output the first n bits of the input, and satisfy the definition with $\varepsilon = 0$ and arbitrarily large circuit size s. The larger the fraction n/l is, the stronger is the generator. Note also that, if the input x is taken in $\{0,1\}^l$ at random, the output $G(x)$ of a generator is also a random variable in $\{0,1\}^n$. But if $l \ll n$, the random variable $G(x)$ is by no means uniformly distributed over $\{0,1\}^n$ since it can take at most 2^l values with nonzero probability.

Pseudorandom generators have many applications in computer science. It is therefore important to know how to construct them. It turns out that this problem (construction of good pseudorandom generators) is related to proving lower bounds on circuit size.

DEFINITION 18.36. Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. We say that f is (s,ε)-*hard* if for any circuit C of size s,

$$\left|\Pr[C(x) = f(x)] - \frac{1}{2}\right| < \varepsilon\,,$$

where x is chosen uniformly at random in $\{0,1\}^n$.

The meaning of this definition is that hard functions f must be "really hard:" no circuit of size s can even approximate its values, that is, any such circuit can do nothing better then just guess the value. So, the function f looks random for each such circuit.

The idea of how hard boolean functions can be used to construct pseudorandom generators is well demonstrated by the following construction of a generator stretching just one bit.

LEMMA 18.37. *Let f be an $(s+1,\varepsilon)$-hard boolean function in n variables. Then the function $G_f : \{0,1\}^n \to \{0,1\}^{n+1}$ defined by $G_f(x) := \big(x, f(x)\big)$ is a (s,ε)-hard pseudorandom generator.*

PROOF. The intuition is that, since f is hard, no small circuit C should be able to figure out that the last bit $f(x)$ of its input string $\big(x, f(x)\big)$ is not just a random bit. By the definition of a pseudorandom generator, we want the following to hold for any circuit of size at most s on $n+1$ variables:

$$\Big|\Pr[C(y) = 1] - \Pr[C(G_f(x)) = 1]\Big| < \varepsilon\,,$$

where y is chosen uniformly at random in $\{0,1\}^{n+1}$, and x in $\{0,1\}^n$. Assume that this does not hold. Then there is a circuit C that violates this property. Without loss of generality, we may assume that

$$\Pr[C(G_f(x)) = 1] - \Pr[C(y) = 1] \geq \varepsilon\,.$$

This can be done because we can take $\neg C$ if this is not the case. The above is the same as

$$\Pr[C(x, f(x)) = 1] - \Pr[C(x, r) = 1] \geq \varepsilon\,,$$

where x is chosen uniformly at random in $\{0,1\}^n$, and r is a random bit in $\{0,1\}$ with $\Pr[r = 0] = \Pr[r = 1] = 1/2$. A way to interpret this inequality is to observe that when the first n input bits of C are a random string x, the circuit C is more likely to accept if the last bit is $f(x)$ than if the last bit is random. This observation suggests the following strategy in order to use C to predict $f(x)$: given an input x for which we want to compute $f(x)$, we guess a value $r \in \{0,1\}$ and compute $C(x,r)$. If $C(x,r) = 1$ we take it as an evidence that r was a good guess for $f(x)$, and output r. If $C(x,r) = 0$, we take it as evidence that r was the wrong guess for $f(x)$, and we output $1 - r$. Let $C_r(x)$ be a random circuit (with just one random input r) we just described. We claim that

$$\Pr_{x,r}[C_r(x) = f(x)] \geq \frac{1}{2} + \varepsilon\,. \tag{18.7}$$

Since $C_r(x) = r$ iff $C(x,r) = 1$, this can be shown by elementary calculations:

$$\begin{aligned}
&\Pr[C_r(x) = f(x)] \\
&= \Pr[C_r(x) = f(x) | r = f(x)] \cdot \Pr[r = f(x)] \\
&\quad + \Pr[C_r(x) = f(x) | r \neq f(x)] \cdot \Pr[r \neq f(x)] \\
&= \tfrac{1}{2} \cdot \Pr[C_r(x) = f(x) | r = f(x)] + \tfrac{1}{2} \cdot \Pr[C_r(x) = f(x) | r \neq f(x)] \\
&= \tfrac{1}{2} \cdot \Pr[C(x,r) = 1 | r = f(x)] + \tfrac{1}{2} \cdot \Pr[C(x,r) = 0 | r = f(x)] \\
&= \tfrac{1}{2} + \tfrac{1}{2} \cdot \Pr[C(x,r) = 1 | r = f(x)] - \tfrac{1}{2} \cdot \Pr[C(x,r) = 1 | r = f(x)] \\
&= \tfrac{1}{2} + \Pr[C(x,r) = 1 | r = f(x)] \\
&\quad - \tfrac{1}{2} \Big(\Pr[C(x,r) = 1 | r = f(x)] + \Pr[C(x,r) = 1 | r \neq f(x)] \Big) \\
&= \tfrac{1}{2} + \Pr_x[C(x, f(x)) = 1] - \Pr_{x,r}[C(x,r) = 1] \\
&\geq \tfrac{1}{2} + \varepsilon \,.
\end{aligned}$$

From (18.7) we obtain that there must be a constant $r \in \{0,1\}$ such that

$$\Pr_x[C_r(x) = f(x)] \geq 1/2 + \varepsilon \,.$$

Since the size of C_r is at most $s+1$ (plus 1 could come from starting with $\neg C$ instead of C), which is a contradiction with the hardness of f. □

To push this strategy further, what we could do is to beak up the input into k blocks and then apply f to them. This way we get a generator stretching n bits into $n+k$ pseudorandom bits. But this is not too much: for applications we need generators stretching n bits into 2^{n^ε} pseudorandom bits. To achieve this, we need to use intersecting blocks. But we also have to ensure that these blocks do not intersect too much. This is the main motivation for the construction of generators known as *Nisan–Wigderson generators*. The starting point of this construction are combinatorial object known as "partial designs."

A collection of subsets $S_1, \ldots, S_n$ of $[l] = \{1, \ldots, l\}$ is called *partial m-design* if $|S_i| = m$ for all i, and $|S_i \cap S_j| \leq \log n$ for all $i \neq j$.

Given such a design and a boolean function $f : \{0,1\}^m \to \{0,1\}$, the Nisan–Wigderson generator $G_f : \{0,1\}^l \to \{0,1\}^n$ is defined by:

$$G_f(x) = \Big(f(x\restriction_{S_1}), f(x\restriction_{S_2}), \ldots, f(x\restriction_{S_n}) \Big) \,,$$

where $x\restriction_S$ is the substring $(x_i \mid i \in S)$ of x. That is, the ith bit of $G_f(x)$ is just the value of f applied to the substring of x determined by the ith set of the design.

Using a similar argument as for the one-bit generator above, one can prove the following:

THEOREM 18.38. *If the function f is $(n^2, 1/n^2)$-hard, then G_f is an $(n^2/2, 2/n^2)$-secure pseudorandom generator.*

Natural proofs

So far, none of existing lower bounds arguments was able to separate **P** from **NP**. Razborov and Rudich (1997) gave an explanation: all these proof techniques are "natural", and natural proofs cannot prove **P** ≠ **NP**, unless good pseudo-random generators do not exist. Since the existence of such generators is widely believed, it seems very unlikely that natural proofs could do this separation. We are not going to

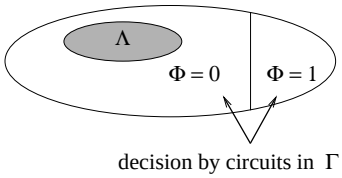


FIGURE 9. A random function $\boldsymbol{f}$ should have a non-negligible chance of having the property Φ. But the value $\Phi(f)$ must be computable by a circuit in Γ taking the 2^n bits of the truth table of f as input.

sink into details of how this is proved; interested reader can just look at a well-written paper of Razborov and Rudich. We just briefly mention what is meant under a "natural proof" and how one could try to avoid this "naturality".

Let $\mathscr{B}_n$ be the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}$, and let Γ and Λ be some classes of boolean functions. We can think of Λ being, say, the class of all boolean functions computable by circuits of size n^2, and Γ being the class $\mathbf{P}/\mathrm{poly}$ of boolean functions computable by circuits of polynomial in n size. Hence, $f \in \Lambda$ iff f can be computed by relatively small circuit (of quadratic size).

Given a specific boolean function $f_0 \in \mathscr{B}_n$, our goal is to show that $f_0 \notin \Lambda$. A possible proof of this fact is a property $\Phi : \mathscr{B}_n \to \{0,1\}$ of boolean functions such that $\Phi(f_0) = 1$ and $\Phi(f) = 0$ for all $f \in \Lambda$. Each such property is a witness for (or a proof of) the fact that "$f_0 \notin \Lambda$."

A Γ-*natural proof against* Λ is a property $\Phi : \mathscr{B}_n \to \{0,1\}$ satisfying the following three conditions:

(1) *Usefulness against* Λ: $\Phi(f) = 1$ implies $f \notin \Lambda$.
(2) *Largeness*: $\Phi(f) = 1$ for at least $2^{-O(n)}$ fraction of all 2^{2^n} functions f in $\mathscr{B}_n$.
(3) *Constructivity*: $\Phi \in \Gamma$, that is, when looked at as a boolean function in $N = 2^n$ variables, the property Φ itself belongs to the class Γ.

The first condition (1) is obvious: after all we want to prove that $f_0 \notin \Lambda$. If $\Lambda \neq \emptyset$, this condition also ensures that Φ cannot be trivial, i.e., take value 1 on all functions. Condition (2) corresponds to our intuition that any reasonable lower bounds argument, designed for a given function f_0, should be also able to show the hardness of the hardest functions—random ones. Thus, a random function $\boldsymbol{f}$ should have a non-negligible chance of having the property Φ. What makes the property 'natural" is the last condition (3). That is, the requirement that the property itself can be tested by not too large circuits.

We emphasize that when property $\Phi(f)$ is computed, the input is the truth table of f, whose size is 2^n, not n. Thus, a property is $\mathbf{P}/\mathrm{poly}$-natural, if it can be computed by circuits of size $2^{O(n)}$, which is more than exponential in n(!)

EXAMPLE 18.39. Let us consider the case when $\Lambda = AC^0$, the class of all boolean functions computable by constant depth circuits with polynomial number of NOT and unbounded fanin AND and OR gates. The proof that Parity $\notin AC^0$ (Section 11.1) involves the following steps: (i) Show that every AC^0 circuit can be simplified to a constant by restricting at most $n - n^\varepsilon$ input variables to constants, and (ii) show that Parity does not have this property. (Here $0 < \varepsilon \leq 1/2$ is a constant depending only on the depth of a circuit, and property (ii) trivially holds, as long as $n - n^\varepsilon \geq 1$.) Thus the natural property lurking in the proof is the following:

$\Phi(f)=1$ iff f cannot be made constant by restricting its all but n^{ε} variables.

Clearly, if $\Phi(f)=1$ then $f\notin AC^0$, so Φ is useful against AC^0. Furthermore, the number of functions that can be made constant by setting its $n-k$ variables does not exceed $2\binom{n}{k}2^{2^{n-k}}\le 2^{n2^{n-k}}$, and this is a negligible fraction of all 2^{2^n} functions. Hence, Φ has largeness property as well. Finally, Φ is constructive in a very strong sense: given a truth table of f, the value $\Phi(f)$ can be computed by a depth-3 circuit of size $2^{O(n)}$ as follows. List all $\binom{n}{k}2^{n-k}=2^{O(n)}$ restrictions of $n-k$ variables. For each one there is a circuit of depth 2 and size $2^{O(n)}$ which outputs 1 iff that restriction does not leave f a constant function, that is, iff the positions in the truth sub-table, corresponding to that restriction, not all are equal. Output AND of all these circuits. The resulting circuit has depth 3 and is polynomial-sized in 2^n.

Thus, property Φ is AC^0-natural against AC^0.

Now we show that natural properties cannot be useful against substantially larger classes of boolean functions, like $\mathbf{P}$/poly, unless good pseudorandom generators do not exist.

A pseudo-random *function generator* is a boolean function $f(x,y):\{0,1\}^{n+n^2}\to\{0,1\}$. By setting the y-variables at random, we obtain its random subfunction $f_{\mathbf{y}}(x)=f(x,\mathbf{y})$. Let $\mathbf{h}:\{0,1\}^n\to\{0,1\}$ be a truly random boolean function. A generator $f(x,y)$ is *secure against* Γ*-attacks* if for every circuit C in Γ,

$$\left|\Pr[C(f_{\mathbf{y}})=1]-\Pr[C(\mathbf{h})=1]\right|<2^{-n^2}. \tag{18.8}$$

That is, no circuit in Γ can distinguish $f_{\mathbf{y}}$ from a truly random function; here again, inputs for circuits are truth tables of boolean functions.

THEOREM 18.40. *If a complexity class Λ contains a pseudo-random function generator that is secure against Γ-attacks, then there is no Γ-natural proof against Λ.*

PROOF. Suppose that a Γ-natural proof Φ against Λ exists. To get a contradiction, we will show that then the proof Φ can be used to distinguish $f_{\mathbf{y}}$ from a random function $\mathbf{h}$.

Since $f(x,y)$ belongs to Γ, every subfunction $f_y(x)$ with $y\in\{0,1\}^{2n}$ belongs to Γ as well. The usefulness of Φ against Λ implies that $\Phi(f_y)=0$ for all y. Hence,

$$\Pr[\Phi(f_{\mathbf{y}})=1]=0.$$

On the other hand, the largeness of Φ implies that $\Pr[\Phi(\mathbf{h})=1]\ge 2^{-O(n)}$. Hence,

$$\left|\Pr[\Phi(f_{\mathbf{y}})=1]-\Pr[\Phi(\mathbf{h})=1]\right|\ge 2^{-O(n)},$$

and thus Φ is a distinguisher. But by constructivity, the boolean function Φ itself belongs to Γ, a contradiction with (18.8). □

It is known that pseudo-random *function* generators may be constructed starting from simpler objects—pseudo-random *number* generators. These are just functions $g_n:\{0,1\}^n\to\{0,1\}^{2n}$. Such a function g_n is secure against Γ-attacks if for every circuit C in Γ,

$$\left|\Pr[C(g_n(\mathbf{x}))=1]-\Pr[C(\mathbf{y})=1]\right|<2^{-n^2}.$$

Here $\mathbf{x}$ is chosen at random from $\{0,1\}^n$ and $\mathbf{y}$ is chosen at random from $\{0,1\}^{2n}$. That is, given a random seed $\mathbf{x}$, g_n produces a random string $\mathbf{y}'=g_n(\mathbf{x})$ in $\{0,1\}^{2n}$, and no circuit in Γ can distinguish this produced string $\mathbf{y}'$ from a truly random string $\mathbf{y}$.

Starting from a pseudo-random number generator $g : \{0,1\}^n \to \{0,1\}^{2n}$, one can construct a pseudo-random function generator $f(x,y) : \{0,1\}^{n+n^2} \to \{0,1\}$ as follows. Associate with g two functions $g_0, g_1 : \{0,1\}^n \to \{0,1\}^n$, where $g_0(x)$ is the first and $g_1(x)$ the second half of the string $g(x)$. Having a vector $y \in \{0,1\}^{n^2}$, we can define a function $S_y : \{0,1\}^n \to \{0,1\}^n$ which is a superposition $S_y = g_{y_n} \circ g_{y_{n-1}} \circ \cdots \circ g_{y_1}$ of these two functions g_0 and g_1 defined by the bits of y. Then just let $f(x,y)$ be the first bit of the superposition S_y applied to input x.

It is widely believed that the class $\Lambda = \mathbf{P}/\text{poly}$ (and even much smaller classes) contain pseudo-random number generators g_n that are secure enough against $\mathbf{P}/\text{poly}$-attacks. It is also known that the pseudo-random function generator $f(x,y)$ constructed from g_n is then also secure enough against $\mathbf{P}/\text{poly}$-attacks. Together with Theorem 18.40, this means that no $\mathbf{P}/\text{poly}$-natural proof can lead to a super-polynomial lower bound on circuit size.

MARGINAL NOTE. This result raised some pessimism among the complexity people: Why try things that are (most probably) impossible? To my opinion, the pessimism is not well founded.

1. The main goal of circuit complexity is *not* to separate **P** from **NP** or some other "uniform" complexity classes—this will probably be done by some cute diagonalization argument (diagonalization is *not* natural). Also, there is a big difference between the classes **P** and $\mathbf{P}/\text{poly}$: the first is "uniform" (requires *one* Turing machine for *all* boolean functions f_n in the sequence $\{f_n \mid n = 1,2,\ldots\}$), whereas the second only requires that for each n a small circuit computing f_n exists. At the beginning of complexity theory, some people (including a great mathematician Kolmogorov) even believed that the whole **NP** is doable with circuits of linear size. Decades passed, and this belief ist still not refuted! The goals of circuit complexity are therefore much more "prosaic:" to prove lower bounds for "simple" explicit functions. In this (pragmatic) respect, even restricted circuit models—like decision trees, bounded-depth circuits, time-restricted branching programs, etc.—are important as they are.

2. The phenomenon of "natural proofs" should be looked at as a guide and a hint that some lower bound arguments cannot be fetched too far. It also answers the question: why proving lower bounds is so difficult? This is difficult because any such proof gives us an algorithm to break down a pseudorandom generator (for the corresponding class of circuits). That is, when proving that something is *not* possible, we actually try to prove that something very strange *is* possible!

3. The discovery of natural proofs phenomenon gives us an additional motivation to search for new arguments. This (striving for better arguments) was always present in circuit complexity. New is that now we know what this "new" means: the arguments must avoid largeness and/or constructivity. So, for example, Chow (2009) has already shown that, if one replaces the largeness condition by "$\Phi(f) = 1$ for at least $|\mathscr{B}_n|/2^{q(n)}$ functions" where $q(n)$ is quasi-polynomial in n, then the resulting "almost-natural" proofs against $\mathbf{P}/\text{poly}$ exist!

4. Mathematics is full with non-constructive proofs—why should we stick on constructive ones?

5. Our intuition that any lower bounds proof should also work for a random function (the largeness condition) might also be wrong. For example, the property of being $K_{2,2}$-free is not shared by random graphs, whereas dense $K_{2,2}$-free graphs have many nice special properties. Thus, the approach based on graph structure of boolean functions—known as "graph complexity"—could also be promising; we sketched this approach in Sections 1.8 and 10.4.

6. We cannot just put the lower bounds problem away—it is too important, both for mathematics and for praxis. My overall opinion is that the idea of natural proofs is just a very nice conceptual frame to classify existing and forthcoming lower bound arguments, as well as an invitation to search for non-constructive, not computable in polynomial time lower bounds criteria. Anyway, this is not a "sentence of depth" for circuit complexity. Just the opposite: thanks to this

discovery, a wild collection of deep and nice lower bound arguments now begins to become a Theory.

The fusion method

In order to show, by contradiction, that a given circuit C is too small for computing a given boolean function f one could try to argue in the following way: try to combine (or "fuse") *correct* rejecting computations of C on inputs in $f^{-1}(0)$ into an *incorrect* rejecting computation on an input in $f^{-1}(1)$.

The idea is to look at circuit $C = (g_1, \ldots, g_t)$ of size t as a *set* Φ_C of local tests on strings $\boldsymbol{r} = (r_1, \ldots, r_t)$ in $\{0,1\}^t$. Namely,

$$\text{each gate } g_i = \varphi(g_{i_1}, \ldots, g_{i_d}) \text{ corresponds to a test } r_i \stackrel{?}{=} \varphi(r_{i_1}, \ldots, r_{i_d}).$$

The first n bits of $\boldsymbol{r}$ correspond to an input vector, and each subsequent bit must pass the corresponding test. If, for example, we consider circuits over the basis $\{\wedge, \vee, \neg\}$, then each of the tests looks at ≤ 3 bits of $\boldsymbol{r}$ and has one of the forms

$$r_i \stackrel{?}{=} \neg r_j \quad r_i \stackrel{?}{=} r_{j_1} \wedge r_{j_2} \quad \text{or} \quad r_i \stackrel{?}{=} r_{j_1} \vee r_{j_2} \quad (j, j_1, j_2 < i).$$

It is clear that each computation $C(\boldsymbol{a}) = (g_1(\boldsymbol{a}), \ldots, g_t(\boldsymbol{a}))$ on a input vector $\boldsymbol{a} \in \{0,1\}^n$ must pass all the tests. It is, however, important to note that also the opposite holds:

$$\text{A string } \boldsymbol{r} \in \{0,1\}^t \text{ is a computation of } C \text{ iff it passes all tests in } \Phi_C.$$

Indeed, if $\boldsymbol{r}$ passes all tests in Φ_C then $\boldsymbol{r}$ is just a computation of C on input $\boldsymbol{a} = (r_1, \ldots, r_n)$, and the result of this computation is the last bit r_t of $\boldsymbol{r}$.

This suggests the following "diagonalization" argument to prove that a given function f cannot be computed by a circuit of size t:

> Show that, for every set Φ of $|\Phi| \leq t$ local tests, there exists a vector $\boldsymbol{r}$ in $\{0,1\}^t$ such that $\boldsymbol{r}$ passes all tests in Φ but $r_t \neq f(r_1, \ldots, r_n)$.

There are two general ideas of how to construct such a "diagonal computation" $\boldsymbol{r}$: the "topological approach" of Sipser (1985) and the "fusion method" first proposed by Razborov (1989a) and then put in a more general frame by Karchmer (1993).

Let $f(x_1, \ldots, x_n)$ be a given boolean function, and let $U = f^{-1}(0)$ be the set of all vectors rejected by f. We look at each gate $g : \{0,1\}^n \to \{0,1\}$ as a (column) vector $\boldsymbol{g} \in \{0,1\}^m$ with $m = |U|$ whose jth position is the value of g when applied to the jth vector in U. In particular, the vector $\boldsymbol{x}_i$ corresponding to an input variable x_i has a 1 in the jth position iff the jth vector of U has 1 the ith position. Put otherwise, the columns $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ form an $m \times n$ matrix A such that $f(\boldsymbol{a}) = 0$ iff $\boldsymbol{a}$ is a row of A.

This way we can look at any circuit $G = (g_1, \ldots, g_t)$ as a boolean m by $2n+t$ matrix M, a *computation matrix*, whose columns are the vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_t$ (see Fig. 18.6.2). This matrix has the following properties:

a. the $(n+i)$-th column is the negation of the i-th column, for $i = 1, \ldots, n$;
b. if $g_i = g_j \wedge g_k$ then the i-th column is the AND of the j-th and the k-th columns;
c. if $g_i = g_j \vee g_k$ then the i-th column is the OR of the j-th and the k-th columns,

where here and throughout, boolean operations on boolean vectors are performed component-wise.

Each boolean function f determines the set $U = f^{-1}(0)$ of its zeroes, as well as the first $2n$ columns $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and $\neg\boldsymbol{x}_1, \ldots, \neg\boldsymbol{x}_n$ of a computation matrix M of any circuit for f. The remaining columns, however, are determined by the gates of a concrete circuit we are considering. To construct a "diagonal" computation we will combine

	x_1	$\cdots$	x_n	$\neg x_1$	$\cdots$	$\neg x_n$	$\cdots$	g_i	$\cdots$	g_t
$\boldsymbol{a}_1$	0	$\cdots$	1	1	$\cdots$	0	$\cdots$	0	$\cdots$	0
$\boldsymbol{a}_2$	1	$\cdots$	1	0	$\cdots$	0	$\cdots$	1	$\cdots$	0
$\vdots$										$\vdots$
$\boldsymbol{a}_m$	1	$\cdots$	0	0	$\cdots$	1	$\cdots$	0	$\cdots$	0
	$F(\boldsymbol{x}_1)$	$\cdots$	$F(\boldsymbol{x}_n)$	$F(\neg\boldsymbol{x}_1)$	$\cdots$	$F(\neg\boldsymbol{x}_n)$	$\cdots$	$F(\boldsymbol{g}_i)$	$\cdots$	0

FIGURE 10. A matrix of a circuit. In a row for vector $\boldsymbol{a}$, the first n positions is the vector $\boldsymbol{a}$ itself, the next n positions is the complemented vector $\boldsymbol{a} \oplus \mathbf{1}$, and the ith further position is the value $g_i(\boldsymbol{a})$ of the ith gate g_i. The last (t-th) position in each row must be 0, since $g_t(\boldsymbol{a}) = f(\boldsymbol{a}) = 0$ for all $\boldsymbol{a} \in U = f^{-1}(0)$.

columns in a new row using boolean functions $F : \{0,1\}^m \to \{0,1\}$ defined on the column space.

We call such a function F a *fusing functional* for f if $F(\mathbf{0}) = 0$ and $F(\neg\boldsymbol{x}_i) = \neg F(\boldsymbol{x}_i)$ for all $i = 1,\ldots,n$, that is, if F respects negations of "basis" columns $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$. Say that a pair $(\boldsymbol{a},\boldsymbol{b})$ of vectors in $\{0,1\}^m$ *covers* a functional F if

$$F(\boldsymbol{a}) \wedge F(\boldsymbol{b}) \neq F(\boldsymbol{a} \wedge \boldsymbol{b}). \tag{18.9}$$

We can now introduce a combinatorial (set-covering) measure characterizing the size of circuits.

Let $\mu(f)$ be the smallest number of pairs of vectors in $\{0,1\}^m$ satisfying the following condition: each monotone fusing functional F for f such that

$$f(F(\boldsymbol{x}_1),\ldots,F(\boldsymbol{x}_n)) = 1 \tag{18.10}$$

is covered by at least one of these pairs. Let $s(f)$ be the smallest number of gates in a DeMorgan circuit computing f.

LEMMA 18.41. *For every boolean function f, $s(f) \geq \mu(f)$.*

PROOF. Let $U = f^{-1}(0)$, $m = |U|$ and let $G = (g_1,\ldots,g_t)$ be a circuit computing f. Take an arbitrary monotone functional $F : \{0,1\}^m \to \{0,1\}$ for f satisfying (18.10). Say that a gate $g_i = g_j * g_k$ with $* \in \{\wedge,\vee\}$ covers F if $F(\boldsymbol{g}_j) * F(\boldsymbol{g}_k) \neq F(\boldsymbol{g}_j * \boldsymbol{g}_k)$. Note that, if none of the gates in G would cover F, then $\boldsymbol{r} = (r_1,\ldots,r_t)$ with $r_i = F(\boldsymbol{g}_i)$ would be a computation $G(\boldsymbol{a}) = (g_1(\boldsymbol{a}),\ldots,g_t(\boldsymbol{a}))$ of our circuit G on the input

$$\boldsymbol{a} := (F(\boldsymbol{x}_1),\ldots,F(\boldsymbol{x}_n)).$$

The fact that $F(\boldsymbol{g}_t) = F(\mathbf{0}) = 0$ would imply that this is a rejecting computation. But (18.10) implies that $f(\boldsymbol{a}) = 1$, and hence, the vector $\boldsymbol{a}$ should be accepted by G, a contradiction.

Thus, the functional F must be covered by at least one gate of G. It suffices therefore to show that if a $\vee$-gate covers F, then F is also covered by an $\wedge$-gate. To show this, let S be the set of all gates in G that cover F. For the sake of contradiction, assume that S contains no $\wedge$-gates.

By the definition of cover we have that for each $g_i = g_j \vee g_k$ in S,

$$F(\boldsymbol{g}_j) \vee F(\boldsymbol{g}_k) \neq F(\boldsymbol{g}_j \vee \boldsymbol{g}_k).$$

Since F is monotone, the only possibility is that

$$F(\boldsymbol{g}_j) = F(\boldsymbol{g}_k) = 0 \quad \text{and} \quad F(\boldsymbol{g}_j \vee \boldsymbol{g}_k) = 1\,. \tag{18.11}$$

Let G' be a circuit identical to G except that each gate $g_i = g_j \vee g_k$ in S is replaced by the instruction $g_i' = 1 \vee 1$. We concentrate on the behavior the both circuits G and G' on the input vector

$$\boldsymbol{a} = (F(\boldsymbol{x}_1), \ldots, F(\boldsymbol{x}_n))$$

defined by the functional F, and make the following two observations:

a. $G'(\boldsymbol{a}) = 1$. This is because we have only changed gates g_i, whose values were 0 on this input (by (18.11)). Since the circuit uses only AND and OR gates, which are monotone, we have that $G'(\boldsymbol{a}) \geq G(\boldsymbol{a}) = f(\boldsymbol{a}) = 1$.
b. The computation of $G'(a) = \big(g_1'(a), \ldots, g_t'(a)\big)$ on input $\boldsymbol{a}$ coincides with the string $F(\boldsymbol{g}_1), \ldots, F(\boldsymbol{g}_t)$, that is, $g_i'(\boldsymbol{a}) = F(\boldsymbol{g}_i)$ for all $i = 1, \ldots, t$. We show this by induction on the position of the gates in G'. Since the first n gates of G' are the same as in G, namely – the variables $x_1, \ldots, x_n$, the claim holds for all $i = 1, \ldots, n$. Take now a gate $g_i = g_j * g_k$ and assume the claim holds for both its inputs, that is $g_j'(\boldsymbol{a}) = F(\boldsymbol{g}_j)$ and $g_k'(\boldsymbol{a}) = F(\boldsymbol{g}_k)$.
   - Case: $g_i = g_j \wedge g_k$. Since, by our assumption, $\wedge$-gates do not cover F, we obtain:

     $$g_i'(\boldsymbol{a}) = g_j'(\boldsymbol{a}) \wedge g_k'(\boldsymbol{a}) = F(\boldsymbol{g}_j) \wedge F(\boldsymbol{g}_k) = F(\boldsymbol{g}_j \wedge g_k) = F(\boldsymbol{g}_i)\,.$$

   - Case: $g_i = g_j \vee g_k$. If $g_i \notin S$, then g_i does not cover F, and the claim follows as in the previous case. If $g_i \in S$, then (18.11) holds, implying that

     $$g_i'(\boldsymbol{a}) = 1 \vee 1 = 1 = F(\boldsymbol{g}_j \vee \boldsymbol{g}_k) = F(\boldsymbol{g}_i)\,.$$

Putting observations (1) and (2) together, we get that $g_t'(\boldsymbol{a}) = g_t(\boldsymbol{a}) = f(\boldsymbol{a}) = 1$, on one side, and $g_t'(\boldsymbol{a}) = F(\boldsymbol{g}_t) = F(\boldsymbol{0}) = 0$, on the other side. Thus, we have a contradiction that S can contain only $\vee$-gates, that is, that only OR gates can cover F. This means that at least one of the pairs $(\boldsymbol{g}_j, \boldsymbol{g}_k)$ of vectors in $\{0,1\}^{|U|}$, corresponding to $\wedge$-gates $g_i = g_j \wedge g_k$ of G, will cover F in the sense of (18.9), as desired. □

It can also be shown (we will not do this) that the lower bound in Lemma 18.41 is tight enough: $s(f) \leq c(\mu(f)+n)^2$ for a constant c. Thus, at least in principle, diagonal computations for (deterministic) circuits *can* be produced by only using monotone functionals. It turned out that different classes of fusing functionals capture different circuit models. A boolean function $F(\boldsymbol{x})$ is *self-dual* if $F(\neg\boldsymbol{x}) = \neg F(\boldsymbol{x})$, and is *affine* if it is a parity (sum modulo 2) of an odd number of variables.

a. Monotone functionals capture deterministic circuits as well as nondeterministic branching programs: classes **P** and **NL**.
b. Monotone self-dual functionals capture nondeterministic circuits: the class **NP**.
c. Affine functionals capture nondeterministic circuits as well as parity branching programs: classes **NP** and $\oplus$**L**.

How does this happen can be found in a nice survey of Wigderson (1993) and in the literature cited therein.

Bibliographic Notes

Strong pseudorandom generators, secure against constant depth circuits, was constructed by Nisan (1991). The concept of natural proofs was invented by Razborov and Rudich (1997). Besides natural proofs and relativization, yet another barrier towards proving $\mathbf{P} \neq \mathbf{NP}$—algebraization—was recently discovered by Aaronson and Wigderson (2008). The fusion method emerged in works of Sipser (1985), Razborov (1989a) and Karchmer (1993).

TABLE 1. In the next two tables we shortly summarize some (not all!) lower bounds arguments as well as some properties of boolean functions making them hard to compute.

Property of f	Circuit model	Meaning
Mixed	General fanin-2 circuits read-once branching programs	For every k-element subset of variables Y, no two assignments to Y lead to the same subfunction.
Robust	DeMorgan formulas, constant depth over $\{\wedge, \vee, \neg\}$	Assigning all but a nontrivial portion of variables to constants leaves a non-constant function.
Hard to cover	DeMorgan formulas	Many rectangles in any monochromatic decomposition of $S_f = f^{-1}(0) \times f^{-1}(1)$.
Hard to approximate	constant depth modular circuits,	$\lvert f \oplus p\rvert \gg 2^{n-n^{\varepsilon}}$ for any degree n^{ε} polynomial p.
	monotone circuits	Any r-CNF C and s-DNF D with $C \leq f$ or $f \leq D$ requires many clauses/monomials.
Clique-free	depth-3 over $\{\oplus, \wedge, \vee, 1\}$	M_f has many 1's but no large all-1 submatrix. Example: $f(x, y) = 1$ iff $x_i \wedge y_i = 0$ for all i.
Large rank	monotone formulas,	Formula size$(f) \geq \mathrm{rk}(D_f)$, where D_f is the disjointness matrix of minterms/maxterms of f.
	monotone span programs	Program size$(f) \geq \mathrm{rk}(D_f)$ if maxterms can be splitted $b = b_0 \cup b_1$ so that $\lvert a \cap b_0\rvert \neq \emptyset \iff \lvert a \cap b_1\rvert = \emptyset$ for all minterms a.
Large rigidity	constant depth over $\{\oplus, \wedge, \vee, 1\}$,	Inputs are any matrices of rank 1.
	log-depth circuits over $\{\oplus, 1\}$	$\mathscr{R}_A\left(n^{\delta}\right) > n^{1+\varepsilon}$ implies super-linear lower bound.
Large entropy	general depth-2 circuits	Setting all but one variable in a subset $Y \subseteq X$ of $\lvert Y\rvert = k$ variables leads to about 2^k different suboperators of $f : \{0,1\}^n \to \{0,1\}^n$. Examples: matrix product, convolution
Rectangle-free	multi-partition communication branching $(1, +R)$-programs	If $r(X) = r_1(X_1) \wedge r_2(X_2)$ with $X_1 \cap X_2 = \emptyset$ and $\lvert X_1\rvert = \lvert X_2\rvert$, and if and $r \leq f$, then $\lvert r^{-1}(1)\rvert$ is small.

TABLE 2

Argument	How it works
Counting/Entropy: formulas, branching programs, span programs, general depth-2 circuits	Split the variables into disjoint blocks $Y_1, \ldots, Y_k$ and, for each block Y_i, show that many different subfunctions of f can be obtained by setting variables outside Y_i to constants.
Gate elimination: general circuits	Show that fixing k inputs to constants eliminates the need of more that k gates.
Random restrictions: DeMorgan formulas, bounded-depth circuits, resolution proofs	Set some variables to constants at random and show that the size of a circuit drops down quickly.
Rank arguments: monotone formulas, monotone span programs, communication protocols	Transform the boolean function f into an appropriate disjointness matrix D_f, or any other matrix whose rank is large. For this, use the cross-intersection property of minterms and maxterms. Show that small circuit for f would allow to reduce the rank of D_f.
Approximations: monotone circuits, bounded-depth circuits, circuits for matrices, circuits with modular gates	Together with random restrictions, this was one of the most fruitful arguments so far! Show that at each gate only a small "progress" towards computing the function f can be made. For this, gradually associate with gates some simpler objects ("approximators"), show that at each gate only few errors can be introduced, and finally, use the properties of the boolean function f computed at the output gate to show that any approximator for f must differ from f on many inputs. So, the number of gates must be large.
Finite limits: depth-3 circuits monotone circuits	Vector $x \in A$ is k-limit for set $B \subseteq \{0,1\}^n$ if, for every $S \subseteq [n]$, $\|S\| = k$ there is $y \in B$ such that $y\restriction_S = x\restriction_S$. Then no depth-2 circuit of bottom fanin $\leq k$, which accepts all vectors in A and rejects all vectors in B, can detect the fact $x \notin B$.
Amplification: communication complexity	If the density $\|S\|/n^k$ of a set $S \subseteq [n]^k$ of "surviving" inputs with respect to the entire universum $[n]^k$ becomes too small, take a projection S' of S onto some $l < k$ coordinates, so that the density $\|S'\|/n^l$ of S' with respect with this smaller universum $[n]^l$ is again large enough.
Adversary arguments: decision tree depth	Fix the bits one by one. Depending on what the algorithm has chosen so far, set the next bit so that the "uncertainty" about the value of f is the largest one.
Spectral arguments: decision tree size	If the sum of absolute values of subsequent Fourier coefficients of f is large, then any decision tree for f needs many nodes.
Cut-and-paste: branching programs	This is a kind of diagonalization. Combine correct accepting computations in a wrong accepting computation.

Bibliography

[1] S. Aaronson and A. Wigderson (2008): Algebrization: A new barrier in complexity theory, in: *Proc. of Ann. ACM Symp. on the Theory of Computing*, 731–740.

[2] A. Aho, J. Ullman and M. Yannakakis (1983): On notions of information transfer in VLSI circuits. In: *Proc. of 15th ACM STOC*, pp. 133–139.

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

[4] M. Ajtai (2002): Determinism versus non-determinism for linear time RAMs with memory restrictions, *J. Comput. Syst. Sci.* 65(1) 2-37.

[5] M. Ajtai (2005): A non-linear time lower bound for boolean branching programs, *Theory of Computing* 1(1) 149-176.

[6] M. Ajtai and Y. Gurevich (1987): Monotone versus positive, *J. of the ACM* **34**, 1004–1015.

[7] E. Allender and M. Koucký (2008): Amplifying lower bounds by means of self-reducibility, in: *Proc. of IEEE Conf. on Computational Complexity*, pp. 31–40.

[8] N. Alon (1986): Covering graphs by the minimum number of equivalence relations, *Combinatorica* **6**, 201–206.

[9] N. Alon and R. Boppana (1987): The monotone circuit complexity of boolean functions, *Combinatorica* **7**:1, 1–22.

[10] N. Alon, M. Karchmer and A. Wigderson (1990): Linear circuits over GF(2), *SIAM. J. Comput.* 19(6),1064-1067.

[11] N. Alon and W. Maass (1988): Meanders and their applications in lower bounds arguments, *J. Comput. Syst. Sci.* **37**(2), 118–129.

[12] K. Amano and A. Maruoka (2005): A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6)\log\log n$ negation gates, *SIAM J. Comput.* 35(1) 201–216.

[13] A. E. Andreev (1985): On a method for obtaining lower bounds for the complexity of individual monotone functions, *Soviet Math. Dokl.* **31**:3, 530–534.

[14] A. E. Andreev (1987): On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes, *Moscow Univ. Math. Bull.* **42**:1,63–66.

[15] L. Babai, P. Frankl and Simon, J. (1986): Complexity classes in communication complexity theory, in *Proc. of 27th Ann. IEEE Symp. on Foundations of Comput. Sci.*,pp. 337–347.

[16] L. Babai, N. Nisan and M. Szegedy (1992): Multiparty protocols, pseudorandom generators for Logspace, and time-space trade-offs, *J. Comput. Syst. Sci.* **45**, 204–232.

[17] L. Babai, T. Hayes and P. Kimmel (1998): The cost of the missing bit: communication complexity with help, in: *Proc. of 30th Ann. ACM Symp. on the Theory of Computing*, 673–682.

[18] L. Babai, A. Gál and A. Wigderson (1999): Superpolynomial lower bounds for monotone span programs, *Combinatorica* **19**:3, 301–319.

[19] L. Babai, A. Gál, P. G. Kimmel and S. V. Lokam (2003): Communication complexity of simultaneous messages, *SIAM J. Comput.* 33(1), 137–166.

[20] D. A. Mix Barrington (1989): Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *J. Comput. Syst. Sci.* 38(1), 150-164.

[21] R. Beals, T. Nishino, and K. Tanaka (1998): On the complexity of negation-limited Boolean networks, *SIAM J. Comput.* **27**:5, 1334–1347.

[22] P. Beame (2000): Proof complexity, in: *Computational Complexity Theory*, S. Rudich and A. Wigderson (eds.), IAS/Park City Math. Series (AMS), vol. 10, 199–246.

[23] P. Beame and T. Pitassi (1996): Simplified and improved resolution lower bounds, in: *Proc. of 37th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 274–282.

[24] P.W. Beame, T.S. Jayram,[3] and M. Saks (2001): Time-space tradeoffs for branching programs, *J. Comput. Syst. Sci.* 63(4), 542–572.
[25] R. Beigel and J. Tarui (1994): On ACC, *Computational Complexity* **4** 350–366.
[26] E. Ben-Sasson and A. Wigderson (2001): Short proofs are narrow - resolution made simple *J. ACM* 48(2), 149–169.
[27] C. Berg and S. Ulfberg (1999): Symmetric approximation arguments for monotone lower bounds without sunflowers *Computat. Complexity* 8(1), 1–20.
[28] S.J. Berkowitz (1982): On some relationships between monotone and non-monotone circuit complexity. Technical Report, University of Toronto.
[29] A. Blake, A. (1937): *Canonical expressions in boolean algebra*, PhD thesis, University of Chicago.
[30] N. Blum (1984): A boolean function requiring 3n network size, *Theoret. Comput. Sci.* 28, 337–345.
[31] M. Blum and R. Impagliazzo (1987): Generic oracles and oracle classes, in: *Proc. of 28th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 118–126.
[32] M.L. Bonet and N. Galesi (1999): A study of proof search algorithms for resolution and polynomial calculu, in: *Proc. of 40th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 422–432.
[33] M. Bonet, T. Pitassi and R. Raz (1997): Lower bounds for cutting planes proofs with small coefficients, *J. Symbolic Logic* 62(3), 708–728.
[34] R B. Boppana and M. Sipser (1990): The complexity of finite functions, in: Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A), pp. 757-804.
[35] A. Borodin, J. von zu Gathen and J. Hopcroft (1982): Fast parallel matrix and GCD computations, in *Proc. of 23rd Ann. IEEE Symp. on Foundations of Comput. Sci.*, pp. 65–71.
[36] A. Borodin, A. Razborov and R. Smolensky (1993), On lower bounds for read-k times branching programs, *Computational Complexity* **3**, 1-18.
[37] S. Buss (1987): Polynomial size proofs of the propositional pigeonhole principle, *J. Symbolic Logic* **52**, 916–927.
[38] S. Buss and T. Pitassi (1998): Resolution and the weak pigeonhole principle, in: *Proc. of Workshop on Comput. Sci. Logic,* Annual Conf. of the EACSL, Lect. Notes in Comput. Sci., Vol. 1414, 149–158, Springer.
[39] A.K. Chandra, M. Furst and R.J. Lipton (1983): Multi-party protocols, in: *Proc. of 15th Ann. ACM Symp. on the Theory of Computing*, 94–99.
[40] D.Y. Cherukhin (2005): The lower estimate of complexity in the class of schemes of depth 2 without restrictions on a basis. *Moscow Univ. Math. Bull.* 60(4), 42–44.
[41] T.Y. Chow (2008): Almost-natural proofs, in *Proc. of 49th Ann. IEEE Symp. on Foundations of Comput. Sci.*, pp. 86-91.
[42] F.R.K. Chung (1990): Quasi-random classes of hypergraphs, *Random Structures and Algorithms* **1**, 363–382.
[43] F.R.K. Chung and P. Tetali (1993): Communication complexity and quasi randomness, *SIAM J. Discrete Math.* **6**, 110–123.
[44] V. Chvátal (1973): Edmonds polytopes and a hierarchy of combinatorial problems, *Disctete Math.* **4**, 305–337.
[45] W. Cook, C.R. Coullard and Gy. Túran (1987): On the complexity of cutting plane proofs, *Discrete Appl. Math.* **18**, 25–38.
[46] M. Davis and H. Putnam (1960): A computing procedure for quantification theory, *J. of the ACM* **7**:3, 210–215.
[47] P.E. Dunne (1988): *The complexity of Boolean networks*, Academic Press Professional, Inc., San Diego, CA.
[48] A. Ehrenfeucht and D. Haussler (1989): Learning decision trees from random examples. *Information and Computation* **82**, 231–246.
[49] P. Erdős and R. Rado(1960): Intersection theorems for systems of sets, *J. London Math. Soc.* **35**, 85–90.
[50] P. Erdös, Some remarks on chromatic graphs (1967): *Colloquium Mathematicum* **16**, 253–256.
[51] M.J. Fischer (1974): The complexity of negation-limited networks–a brief survey, in: Springer Lect. Notes in Comput. Sci., vol. 33, 71–82.
[52] M.J. Fischer(1996): Lectures on network complexity. Technical Report TR-1104, Department of Computer Science, Yale University.
[53] M. Furst, J. Saxe and M. Sipser (1984): Parity, circuits and the polynomial time hierarchy, *Math. Syst. Theory* **17**, 13–27.

[3] Formely Jayram S. Thathachar

[54] A. Gál (1998): A characterization of span program size and improved lower bounds for monotone span programs, in: *Proc. of 30th Ann. ACM Symp. on the Theory of Computing*, 429–437.
[55] A. Gál and P. Pudlák (2003): A note on monotone complexity and the rank of matrices, *Inf. Process. Lett.* 87(6), 321-326.
[56] R. E. Gomory (1963): An algorithm for integer solutions of linear programs, in: Recent advances in mathematical programming, R. L. Graves and P. Wolfe, eds., McGraw-Hill, pp. 269–302.
[57] F. Green, J. Köbler, K. W. Regan, T. Schwentick and J. Torán (1995): The power of the middle bit of a # P function, *J.Comput. Syst. Sci.* 50(3) 456–467.
[58] M. Grigni and M. Sipser (1995): Monotone Separation of Logarithmic Space from Logarithmic Depth, *J. Comput. Syst. Sci.* 50(3), 433-437.
[59] V. Grolmusz (1994): The BNS lower bound for multi-party protocols is nearly optimal, *Information and Computation* **112**:1, 51–54.
[60] M. Grötschel, L. Lovász and A. Schrijver (1981): The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* **1**, 169–197.
[61] A. Haken (1985): The intractability of resolution, *Theor. Comput. Sci.* **39**, 297–308.
[62] A. Haken (1995): Counting bottlenecks to show monotone P≠NP, in: *Proc. of 36th Ann. IEEE Symp. on Foundations of Comput. Sci.*, pp. 36–40.
[63] V. Grolmusz and G. Tardos (2003): A note on non-deterministic communication complexity with few witnesses, *Theory of Comput. Syst.* 36(4), 387–391.
[64] J. Hartmanis and L. A. Hemachandra (1991): One-way functions, robustness and non-isomorphism of NP-complete classes, *Theor. Comput. Sci.* **81**:1, 155–163.
[65] A. Haken, and A. Cook, S. (1999): An exponential lower bound for the size of monotone real circuits, *J. Comput. Syst. Sci.* **58**:2, 326–225.
[66] G. H. Hardy, J. E. Littlewood and Polya, G. (1952): *Inequalities*, Cambridge University Press, 1952.
[67] D. Harnik and R. Raz (2000): Higher lower bounds on monotone size, ı: *Proc. of Ann. ACM Symp. on the Theory of Computing* 2000, pp. 378-387.
[68] J. Hastad(1986): *Computational Limitations for Small Depth Circuits*, MIT Press.
[69] J. Hastad (1989): Almost optimal lower bounds for small depth circuits, in: *Advances in Computing Research*, S. Micali (ed.), Vol. 5, pp. 143–170.
[70] J. Hastad (1993): The shrinkage exponent is 2, in: *Proc. of 34th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 114–123.
[71] J. Hastad and M. Goldmann (1991): On the power of small-depth threshold circuits, *Comput. Complexity* 1(2), 113–129.
[72] J. Hastad, S. Jukna and P. Pudlak (1995): Top-down lower bounds for depth-three circuits, *Computational Complexity* **5**, 99-112.
[73] T. P. Hayes (2001): Separating the k-party communication hierarchy: an application of the Zarankiewicz problem. Manuscript, November 2001.
[74] E. A. Hirsch (2000): SAT local search algorithms: worst-case study, *J. Autom. Reasoning* **24**(1/2), 127–143.
[75] P. Hrubes, S. Jukna, A. Kulikov and P. Pudlák (2009): On convex complexity mesures, ECCC Report Nr. 40.
[76] R. Impagliazzo, T. Pitassi and A. Urquhart (1994): Upper and lower bounds for tree-like cutting planes proofs, in: *Proc. of 9th Ann. IEEE Symp. on Logic in Computer Science (4-7 July 1994, Paris, France*, pp. 220-228.
[77] K. Iwama, O. Lachish, H. Morizumi and R. Raz (2001): An explicit lower bound of 5n - o(n) for boolean circuits, in: *Proc. of 33rd Ann. ACM Symp. on the Theory of Computing*, pp. 399-408.
[78] J. W. Jordan and R. Livné (1997): Ramanujan local systems on graphs, *Topology* 36(5), 1007-1024.
[79] S. Jukna (1999): Combinatorics of monotone computations, *Combinatorica*, **19**:1 (1999), 1–21.
[80] S. Jukna (2004): On the minimum number of negations leading to super-polynomial savings, *Inf. Process. Letters*, 89:2, 71–74.
[81] S. Jukna (2006): On graph complexity, *Combinatorics, Probability and Computing* **15**, 855–876.
[82] S. Jukna (2008): Expanders and time-restricted branching programs, *Theoret. Comput. Sci.* 409 (3), 471-476
[83] S. Jukna (2008): Entropy of operators or why matrix multiplication is hard for depth-two circuits (2008), *Theory of Comput. Syst.* doi 10.1007/s00224-008-9133-y.
[84] S. Jukna (2009): On set intersection representations of graphs. *J. Graph Theory* 61(1), 55–75.
[85] S. Jukna and A. Razborov (1998): Neither reading few bits twice nor reading illegally helps much, *Discrete Appl. Math.* 85:3, 223-238

[86] S. Jukna, A. Razborov, P. Savický and I. Wegener (1999): On P versus NP∩co-NP for decision trees and read-once branching programs, *Computational Complexity* **8**:4, 357–370.

[87] S. Jukna and G. Schnitger (2009): Min-rank conjecture for log-depth circuits, ECCC Report Nr. 8.

[88] B. Kalyanasundaram and G. Schnitger (1992): The probabilistic communication complexity of set intersection, *SIAM J. Discrete Mathematics* **5**:4, 545–557.

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

[90] M. Karchmer and A. Wigderson (1990): Monotone circuits for connectivity require super-logarithmic depth, *SIAM J. on Discrete Math.*, **3**, 255-265.

[91] M. Karchmer and A. Wigderson (1993): On span programs, in: *Proc. of 8th Ann. IEEE Conf. on Structure in Complexity Theory*, 102–111.

[92] M. Karchmer, E. Kushilevitz and N. Nisan (1995): Fractional covers and communication complexity, *SIAM J. on Discrete Math.* 8(1) 76–92.

[93] M. Karchmer, I. Newman, M. Saks and A. Wigderson (1994): Non-deterministic communication complexity with few witnesses, *J. Comput. Syst. Sci.* 49(2), 247-257.

[94] M. V. Khrapchenko (1972): A method of obtaining lower bounds for the complexity of π-schemes, *Math. Notes Acad. Sci.* USSR 10 (1972) 474–479.

[95] J. Krajiček (1994): Lower bounds to the size of constant-depth propositional proofs, *J. Symbolic Logic* 1, 73–86.

[96] E. Kushilevitz and N. Nisan (1997): *Communication complexity*, Cambridge University Press.

[97] L. Lovász, M. Naor, I. Newman and A. Wigderson (1995): Search problems in the decision tree model, *SIAM J. Discrete Math.* 8(1), 119–132.

[98] E. Kushilevitz and Y. Mansour 1991): Learning decision trees using the Fourier spectrum, in: *Proc. of 23rd Ann. ACM Symp. on the Theory of Computing*, 455–464.

[99] E. Kushilevitz and E. Weinreb (2009): On the complexity of communication complexity, in: *Proc. of Ann. ACM Symp. on the Theory of Computing*, 465–473.

[100] N. Linial, Y. Mansour and N. Nisan (1989): Constant depth circuits, Fourier transforms and learnability, in: *Proc. of 30th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 574–579. Journal version in: *J. ACM* **40** (1993), 607–620.

[101] L. Lovász (1979a): On the Shannon capacity of a graph, *IEEE Trans. on Information Theory* 25, 1–7.

[102] L. Lovász (1979b): Determmants, matchings and random algorithms, in L. Budach, ed., Proc. of FCT89. Akademie-Verlag, Berlin, pp. 565-574,

[103] L. Lovász and M. E. Saks (1993): Communication complexity and combinatorial lattice theory, *J. Comput. Syst. Sci.* 47(2), 322–349.

[104] L. Lovász, D. B. Shmoys and E. Tardos (1995): Combinatorics in computer science, in: *Handbook of Combinatorics*, R. Graham, M. Grötschel, and L. Lovász (eds.), Elsevier Science, Vol. 2, 2003–2038.

[105] A. Lubotzky, R. Phillips, P. Sarnak (1988): Ramanujan graphs, *Combinatorica* 8(3), 261-277.

[106] O. B. Lupanov (1958): A method of circuit synthesis, *Izvesitya VUZ, Radiofiz* Vol. 1, pp. 120Ű-140 (in Russina).

[107] F. J. MacWilliams and N. J. A. Sloane (1997): *The theory of error-correcting codes*, Elsevier, North-Holl.

[108] G. A. Margulis (1973): Explicit constructions of concentrators, Probl. Peredachi Inf. 9 (1973) 71-80 (in Rusian). Translation: Problems of Inf. Transm. (1975), 323-332.

[109] A.A. Markov (1957): On the inversion complexity of systems of Boolean functions, *Doklady Academii Nauk SSSR*, **116**, 917–919 (in Russian). English translation in: *J. of ACM*, **5**:4 (1958), 331–334, and in: *Soviet Math. Doklady* **4** (1963), 694–696.

[110] G. Midrijanis (2005): Three lines proof of the lower bound for the matrix rigidity, arXiv:cs/0506081v2.

[111] M. Morgenstern (1994): Existence and explicit constructions of $q+1$ regular Ramanujan graphs for every prime power q, *J. Comb. Theory Ser. B*, 62(1), 44-62.

[112] H. Morizumi (2008): A note on the inversion complexity of boolean functions in boolean formulas, CoRR abs/0811.0699, http://arxiv.org/abs/0811.0699.

[113] F. Mayer auf der Heide (1984): A polynomila linear search algorithm for the n-dimensional knapsack problem, *J. of the ACM* , **31**(3), 668–676.

[114] K. Mulmuley, U. Vazirani and V. Vazirani (1987): Matching is as easy as matrix inversion, *Combinatorica* **7**, 105–114.

[115] S. Muroga (1971): *Threshold Logic and Its Applications*, Wiley-Interscience.

[116] E.I. Nechiporuk (1966): On a Boolean function, *Doklady Akademii Nauk SSSR*, **169**:4, 765–766 (in Russian). English translation in: *Soviet Mathematics Doklady* **7**:4, 999–1000.

[117] I Newman (1991): Private vs. common random bits in communication complexity, *Inf. Process. Lett.* 39(2), 67–71.

[118] R. G. Nigmatullin (1983): *The Complexity of Boolean Functions*, Kazan University Press (in Russian).

[119] N. Nisan (1989): CREW PRAM's and decision trees, in: *Proc. of 21st Ann. ACM Symp. on the Theory of Computing*, 327–335.

[120] N. Nisan (1991): Pseudorandom bits for constant depth circuits, *Combinatorica* 11(1), 63–70.

[121] R. J. Lipton and R. Sedgewick (1981): Lower bounds for VLSI, in *Proc. of 13th Ann. ACM Symp. on the Theory of Computing*, 300–307.

[122] N. Nisan (2002): The communication complexity of approximate set packing and covering, in: P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, R. Conejo (Eds.), *29th Int. Colloq. on Automata, Languages and Programming*, ICALP 2002 (Malaga, Spain, July 8-13, 2002), Lect. Notes in Comput. Sci.2380, 868-875.

[123] E.A. Okolnishnikova (1982): On the influence of negation on the complexity of realization of monotone Boolean functions by formulas of bounded depth, in: *Metody Diskretnogo Analiza* **38**, 74–80 (in Russian)

[124] C. H. Papadimitriou (1991): On selecting a satisfying truth assignment, in: *Proc. of 32nd Ann. IEEE Symp. on Foundations of Comput. Sci.*, 163–169.

[125] Ch. H. Papadimitriou and M. Sipser (1984): Communication complexity, *J. Comput. Syst. Sci.*, **28**:2 (1984), 260–269.

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

[127] P. Pudlák, Communication in bounded depth circuits (1994): Combinatorica 14(2), 203–216.

[128] P. Pudlák (1997): Lower bounds for resolution and cutting plane proofs and monotone computations, *J. Symbol. Logic* **62**(3), 981–998.

[129] P. Pudlák, V. Rdl and P. Savický (1988): Graph Complexity, *Acta Inf.* 25(5): 515-535.

[130] P. Pudlák and J. Sgall (1998): Algebraic models of computation and interpolation for algebraic proof systems, in: *Proof Complexity and Feasible Arithmetics*, P.W. Beame and S.R. Buss (eds.), DIMACS Series in Discrete Math. and Theor. Comput. Sci., Vol. 39, 279–295.

[131] P. Pudlák, V. Rödl, J. Sgall (1997): Boolean circuits, tensor ranks, and communication complexity, SIAM J. Comput. 26(3) 605–633.

[132] P. Pudlák and V. Rödl (2004): Pseudorandom sets and explicit constructions of Ramsey graphs, in: J. Krajicek (ed.) *Quad. Mat.* 13, pp. 327–346.

[133] R. Raz (2001): Resolution lower bounds for the weak pigeonhole principle, *J. ACM* 51(2), 115-138

[134] R. Raz (2000): The BNS–Chung criterion for multi-party communication complexity, *Computational Complexity* **9**, 113–122.

[135] R. Raz and A. Wigderson (1989): Probabilistic communication complexity of Boolean relations, in: Proc. of 30th Ann. IEEE Symp. on Foundations of Comput. Sci., 562–567.

[136] R. Raz and A. Wigderson (1992): Monotone circuits for matching require linear depth, *J. ACM* 39(3), 736-744.

[137] A. A. Razborov (1985a): Lower bounds for the monotone complexity of some boolean functions, *Soviet Math. Dokl.* **31**, 354–357.

[138] A. A. Razborov (1985b): A lower bound on the monotone network complexity of the logical permanent, *Matematicheskie Zametki*, **37**:6, 887–990 (in Russian). English translation in: Math. Notes Acad. of Sci. USSR, **37**:6 (1985), 485–493.

[139] A. A. Razborov (1987): Lower bounds on the size of bounded-depth networks over a complete basis with logical addition, *Math. Notes of the Acad. of Sci. of the USSR* **41**:4, 333–338.

[140] A. A. Razborov (1988): Bounded-depth formulae over the basis $\{\&, \oplus\}$ and some combinatorial problem, in: S.I. Adian (ed.), *Problems of Cybernetics, Complexity Theory and Applied Mathematical Logic* (VINITI, Moscow) 149–166. (in Russian)

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

[142] A. A. Razborov (1989b): On rigid matrices. Preprint, Steklov Mathematical Institute, 19 pp. (in Russian)

[143] A. A. Razborov (1990): Applications of matrix methods to the theory of lower bounds in computational complexity, *Combinatorica,* **10**:1, 81–93.

[144] A. A. Razborov (1992a): On the distributional complexity of disjointness, *Theoret. Comput. Sci.* **106**:2, 385–390.

[145] A. A. Razborov (1992b): On submodular complexity measures, in: Boolean Function Complexity, London Math. Soc. Lecture Note Series 169, 76–83.

[146] A. A. Razborov (1995): Bounded arithmetics and lower bounds in boolean complexity, in: *Feasible Mathematics II*, P. Clote and J. Remmel (eds.), Proc. of Workshop (Cornell University, Ithaca, NY, USA, May 28-30, 1992), Birkhäuser, Boston, MA.

[147] A. A. Razborov (1996): On small size approximation models, in: *The Mathematics of Paul Erdos I. Algorithms and Combinatorics*, pp. 385–392.

[148] A. Razborov and A. Wigderson (1993): $n^{\Omega(\log n)}$ Lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom, *Inf. Process. Letters* 45, 303–307.

[149] A. A. Razborov and S. Rudich (1997): Natural proofs, *J. Comput. Syst. Sci.* 55(1), 24-35.

[150] A .A. Razborov and A .A. Sherstov (2008): The sign-rank of AC^O, in: *Proc. of 49th Ann. IEEE Symp. on Foundations of Comput. Sci.* 57-66

[151] N. P. Redkin (1973): Proof of minimality of circuits consisting of funcitonal elements, *Syst. Th. Res.* 23, 85–110.

[152] R. L. Rivest and J. Vuillemin (1976): On recognizing graph properties from adjacency matrices *Theor. Comput. Sci.* 3(3), 371-384.

[153] A. Rosenbloom (1997): Monotone circuits are more powerful than monotone boolean circuits, *Inf. Process. Letters* **61**:3, pp. 161–164.

[154] M. Santha and Ch. Wilson (1993): Limiting negations in constant depth circuits, *SIAM J. Comput.* **22**:2, 294–302.

[155] J. E. Savage (1976): *The Complexity of Computing*, Wiley, New York.

[156] P. Savický and S. Zák (1997): A lower bound on branching programs reading some bits twice, *Theor. Comput. Sci.* **172**, 293–301.

[157] C. Schnorr (1974): Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen, *Computing* 13, 155-171.

[158] C. E. Shannon (1949): The synthesis of two-terminal switching circuits, *Bell Systems Technical Journal* **28**, 59–98.

[159] M. Sipser (1985): A topological view of some problems in complexity theory, in *Colloquia Mathematica Societatis János Bolyai* **44**, pp. 387-391.

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

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

[162] J. Spencer (1995): Probabilistic methods, in: *Handbook of Combinatorics*, R. Graham, M. Grötschel and L. Lovász (eds.), Elsevier-Science, Vol. 2, 1786–1817.

[163] P.M. Spira (1971), On time–hardware complexity tradeoffs for Boolean functions, *Proceedings of 4th Hawaii Symposium on System Sciences*, Western Periodicals Company, North Hollywood, 525–527.

[164] L. Stockmeyer (1977): On the combinational complexity of certain symmetric Boolean functions, *Math. System Theory* **10**, 323–336.

[165] B. A. Subbotovskaya (1961), Realizations of linear functions by formulas using $+, ., -$, *Doklady Akademii Nauk SSSR*, **136**:3, pp. 553–555 (in Russian). English translation in: *Soviet Mathematics Doklady* **2**, 110–112.

[166] K. Tanaka and T. Nishino (1994): On the complexity of negation-limited Boolean networks, in *Proc. 26th Annual ACM Symposium on the Theory of Computing* (ACM, New-York), 38–47.

[167] É. Tardos (1987): The gap between monotone and non-monotone circuit complexity is exponential, *Combinatorica*, **7**:4, 141–142.

[168] G. Tardos (1989): Query complexity, or why is it difficult to separate $NP^A \cap co - NP^A$ from P^A by a random oracle *A*? *Combinatorica* **8**:4, 385–392.

[169] Shi-Chun Tsai (2001): A depth 3 circuit lower bound for the parity function *J. Inf. Sci. Eng.* 17(5), 857–860.

[170] G. C. Tseitin (1968): On the complexity of derivations in propositional calculus, in: *Studies in mathematics and mathematical logic*, Part II, Slisenko, A.O. (ed.), 115–125.

[171] L. G. Valiant (1977): Graph-theoretic methods in low-level complexity, in *Proc. of 6th Conf. on Mathematical Foundations of Computer Science*, *Springer Lect. Notes in Comput. Sci.* **53** 162–176. Springer-Verlag.

[172] L. G. Valiant (1986): Negation is powerless for Boolean slice functions, *SIAM J. Comput.* **15**, 531–535.

[173] L. G. Valiant and V. V. Vazirani (1986): NP is as easy as detecting unique solutions, *Theor. Comput. Sci.* **47**, 85–93.

[174] I. Wegener (1985): On the complexity of slice functions, *Theor. Comut. Sci.* 38, 55–68.
[175] I. Wegener (1987): *The complexity of Boolean functions*", Wiley-Teubner.
[176] A. Wigderson (1993): The fusion method for lower bounds in circuit complexity, in: *Combinatorics, Paul Erdös is Eighty*, Vol. 1, 453–468.
[177] A. Wigderson (1994): NL/poly $\subseteq \oplus$L/poly, in: *Proc. of 9th Ann. IEEE Conf. on Structure in Complexity Theory*, 59–62.
[178] M. Yannakakis (1991): Expressing combinatorial optimization problems by linear programs, *J. Comput. Syst. Sci.*, **43**, 441–466.
[179] A. C. Yao (1979): Some complexity questions related to distributed computing, in: *Proc. of 11th Ann. ACM Symp. on the Theory of Computing*, 209–213.
[180] A. C. Yao(1981): The entropic limitations of VLSI computations, in: *Proc. of 13th Ann. ACM Symp. on the Theory of Computing*, 308–311.
[181] A. C. Yao (1985): Separating the polynomial time hierarchy by oracles, in: *Proc. of 26th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 1–10.
[182] A. C. Yao (1990): On ACC and threshold circuits, in: *Proc. of 31th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 619–627.
[183] S. Zák (1995): A superpolynomial lower bound for $(1, +k(n))$-branching programs, in: *Proc. MFCS'95*, Springer Lect. Notes in Comput. Sci, vol. 969, 319–325.
[184] U. Zwick (1991), An extension of Khrapchenko's theorem, *Information Processing Letters*, **37**, 215–217.
[185] U. Zwick (1991b): A $4n$ lower bound on the combinatorial complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions *SIAM J. Comput.* 20, 499-505.

Index