On Some Most Probable Separations of Complexity Classes

Jin-yi Cai

TR 86-778
(Ph.D. Thesis)
August 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853

On Some Most Probable Separations of Complexity Classes

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jin-yi Cai

August, 1986

On Some Most Probable Separations of

Complexity Classes

Jin-yi Cai, Ph.D.

Cornell University 1986

This thesis is a study of separations of some complexity classes which take place in almost all relativized worlds. We achieve probability one separations of $PSPACE$ from the Polynomial-time Hierarchy PH. Also we separate with probability one all levels of the Boolean Hierarchy BH.

The study on the Boolean Hierarchy is a continuation of the work by Bennet and Gill in [BG81] and the joint work in [CH86], where we introduced the "sawing" argument. This "sawing" technique is adapted here to yield probability one separation.

The study on $PSPACE$ versus the Polynomial-time Hierarchy is more intriguing. Several novel techniques are employed here. The connection with Boolean circuit is expoited to reduce the problem to a Boolean circuit computation problem.

The fixed depth unbounded fan-in Boolean circuit model is considered in connection with the parity function. We show that with an exponential bound of the form $exp(n^\lambda)$ on the size of the circuits, they make asymptotically 50% error on all possible inputs, uniformly. Certain probabilistic and game theoretic methods are applied extensively to conclude the result.

Biographical Sketch

Jin-yi Cai was born in Shanghai, China on January 23, 1961.

Starting January 1978 he attended Fudan University in Shanghai as a mathematics major. He continued his mathematics study at Temple University as a graduate student, and received an M.A. in mathematics in 1983.

The scenic beauty of Ithaca surrounded him for the ensuing 3 years of his study under Prof. Juris Hartmanis.

He was awarded an M.S. in computer science by Cornell University in 1985.

Acknowledgements

From China to America, I am grateful to many people who have shaped and helped me on my study; indeed too many to mention them all.

During my undergraduate years at Fudan University, I was introduced to the rigor and beauty of modern mathematics. Two professors, Dao-xing Xia and Shao-zong Yan, guided me most during my study at Fudan. To them I owe my sincere gratitude.

After my arrival at Temple, I was deeply moved by the warmth and encouragement from all the professors I knew at the Mathematics Department. I learned most from Donald Newman, whose brilliance and intellectual clarity impressed me greatly; I feel fortunate to have studied with him. Arthur Poe taught me the first course in computer science. They were the guiding force for my interests and decision to enter the field of computer science. I am very thankful to them, as well as many others at Temple, who had helped to make my first two years in America a wonderful remembrance.

Many professors at Cornell have my gratitude. Tom Coleman, John Gilbert, Greg Johnson, Anil Nerode, Richard Shore, Charlie Van Loan and many more have made my Cornell years a rich experience. It was Juris Hartmanis, my thesis advisor, who guided me most in my Ph.D. study. He showed me the right way to do research. A master—his intuitive feeling on the entire subject of feasible computation, his approach to problems and his intellectual power have influenced and guided me very much. I could not have found a better advisor. I want to thank him for his generosity of his time and ideas, for his concern and encouragement.

I would like to thank Anita Affeldt, the graduate secretary, who is always so helpful. I would also like to thank many fellow students, Lane Hemachandra, Jim Hook, John Johnstone, Ming Li, Peter Mark, San-zheng Qiao and Scott Smith, and others; I am grateful to have these wonderful friends. Scott and John read my early draft of the main result in this thesis. Jim and Anita spent a great deal of time helping me using the type-setting system. I am very grateful to them.

To my parents and Gabriele.

Contents

Chapter 1

Introduction

One way to live forever for mathematicians, so the saying goes, is to choose a problem that is so hard and important that you can't die nor solve it. If there is any such problem in computer science today, it is the separation of various time-space complexity classes, represented by the famous $\boldsymbol{P} \stackrel{?}{=} \boldsymbol{NP}$ question [HU79].

Since the time of Euclid, mathematicians have had a notion of an algorithm. It was one of the most elementary yet intriguing concept man ever came to know. Many beautiful theorems were proved with this notion from antiquity. One knows a particular algorithm does constitute an effective procedure when one sees it. However, it is reasonable to say that this very concept of an effective computational procedure was only made precise in this century. The notion from ancient times was imprecise because one did not know what it meant nor how to prove the *nonexistance* of an algorithm.

A famous example of this inadequacy of a merely intuitive notion of an algorithm is Hilbert's 10th problem. In his address to the International Congress of Mathematicians at the turn of the century, he asked for an algorithm to solve a general Diophantine equation. Some 70 years later, contrary to any imagination of the old time, it was proven that such a general algorithm did not exist [Mat70].

In the 1930's, mathematicians made a sensational leap forward in our understanding of what were and what were not possibly computable. The research on computability emerged as an important branch of mathematics known as recursion theory.

However, in the real world, we should not be content with the knowledge that a problem is ultimately computable. When a problem *is* computable, it is also extremely important to know that a computation terminates with an answer using reasonable amount of time and space. With the advent of modern computers the question becomes ever more pressing.

It was during the 60's, we began to systematically investigate the efficiency of problem solving, in what became known today as computational complexity theory [HS65].

Just as the recursive sets are identified with the intuitive notion of computable sets, the class P of all problems computable in time polynomial of the problem size is regarded as feasibly computable problems, or easy problems [Cob64] [Edm65]. (Precise definitions for this and others will be given in later chapters.)

Are there problems computable but not *feasibly* computable? Hartmanis, Lewis and Stearns' hierarchy theorem [HU79] tells us yes. Furthermore, for many natural problems no efficient algorithm (i.e. in polynomial time) is known. Of course being unable to find one is different from proving there is none, which is usually exceedingly difficult. However, for literally 1,000 plus interesting problems found in computer science and mathematics [GJ79] no such easy algorithm is expected to exist. This is supported by the strong evidence of NP-completeness theory introduced by Cook and Karp [Coo71] [Kar72].

Let us consider for example the problem of coloring a map. Suppose you are given a map of the newly discovered alien world. You want to color this map so that no two adjacent countries are painted same color. It is easy if you have many colors. It turns out that four colors is all you need, thanks to the famous Four Color Theorem [AH77a] [AH77b]. But how about three colors? It is *not* always possible. Moreover, from a computational point of view, we have an interesting case here. If such a coloring does not exist, we know of no systematic way to demonstrate this fact without spending exponential amount of time to examine essentially all the possibilities. On the other hand, if such a coloring exists for that map, one can simply guess a coloring scheme and then verify that it is indeed a valid coloring using three colors.

We observe an essential feature of this problem from a computational standpoint: when a solution exists a clever guess will lead to an easy verification, but if no solution exists then no guess will succeed and it is not easy to verify that. Many other problems share this asymmetry.

For example, consider the following problems:

- ***SAT***: Given a Boolean formula f, is the formula satisfiable, i.e. is there an assignment to the Boolean variables that satisfies f?

- ***HAM***: Given a graph, is there a Hamiltonian circuit, i.e. is there a tour of the graph that visits every node exactly once?

- ***IntegerLinearProgramming***: Given an $m \times n$ matrix of integers A and an integer vector b, does there exist an integer vector x such that $Ax \geq b$?

We refer to [GJ79] for hundreds more such interesting problems. All these problems are hard; no easy algorithm is known to produce a solution, but it is easy to verify that a guessed solution is indeed a solution.

NP, nondeterministic polynomial time, is the class intended to capture these problems. By nondeterminism, an algorithm is allowed to "guess".

For a given instance of the problem, we say the nondeterministic algorithm solves the problem if some guess leads to a solution.

The class $\boldsymbol{NP}$ is a very rich class. It includes $\boldsymbol{P}$, probably much more. The typical $\boldsymbol{NP}$ problems are, however, not known to be in $\boldsymbol{P}$. The problems listed earlier are certainly in $\boldsymbol{NP}$. In fact, they are among the "hardest" problems in $\boldsymbol{NP}$, in a precise sense. We say an $\boldsymbol{NP}$ problem is complete, if every other $\boldsymbol{NP}$ problem can be easily computed from it. The fact that literally over a thousand problems (including those listed earlier) are known to be $\boldsymbol{NP}$-complete, and none of these admits an easy algorithm despite persistent effort by researchers, is a strong evidence in support of the conjecture $\boldsymbol{P} \neq \boldsymbol{NP}$, which implies that no easy algorithm for those problems exists. However, the proof of this conjecture seems still beyond current capabilities.

An important technique in the study of the structure of complexity theory is to compare the relative complexities of different problems. In what is called relativization techniques, a complexity class is endowed with an additional "black box" that solves some problem such as $\boldsymbol{SAT}$. As we have indicated before, the $\boldsymbol{NP}$-complete problems are the hardest in $\boldsymbol{NP}$ in the sense that every $\boldsymbol{NP}$ problem can be easily (i.e. in $\boldsymbol{P}$) computed from it.

Suppose we have all the answers to an $\boldsymbol{NP}$-complete problem, say $\boldsymbol{SAT}$, what can we do with the machinery of nondeterminism in polynomial time?

Let's consider the following problem:

- ***RBF***: Given a Boolean formula f, is it reducible to a shorter one, i.e. is there an equivalent Boolean formula g that is shorter and tautologically equivalent to f?

It is not known how to solve ***RBF*** in ***NP***. In fact it is conjectured that such a nondeterministic algorithm does not exist. On the other hand using an ***NP*** oracle ***SAT***, one can solve ***RBF*** in nondeterministic polynomial time as follows: Guess a shorter formula g, then asks the oracle ***SAT*** if $\neg[f \leftrightarrow g]$ is satisfiable. f can be reduced to a shorter formula *iff* some guess g will lead to an unsatisfiable $\neg[f \leftrightarrow g]$.

What we have discussed is the class Σ_2^P, languages for which there is a nondeterministic polynomial time algorithm given the answers to ***SAT***. This process can be extended infinitely, resulting what's known as the Meyer-Stockmeyer Polynomial-time Hierarchy, ***PH***. The conjecture that $\boldsymbol{RBF} \notin \boldsymbol{NP}$ implies $\Sigma_2^P \neq \boldsymbol{NP}$, which in turn implies $\boldsymbol{P} \neq \boldsymbol{NP}$.

There is however a direct way to simulate all the computations in the Polynomial-time Hierarchy with repeated search and verification. This simulation may take a long time, but it can be done without using excessive amount of space. Let ***PSPACE*** be the class of problems solvable in polynomial space. We know that $\boldsymbol{PH} \subseteq \boldsymbol{PSPACE}$. Whether ***PH*** is properly contained in $\boldsymbol{PSPACE}$ is another major open problem in complexity theory.

Relativization techniques have been studied from a different perspective. Instead of restricting ourselves to a fixed "black box" of information such as $\boldsymbol{SAT}$, we may want to endow a complexity class with an abstract set of "information". It is then natural to compare two complexity classes such as $\boldsymbol{P}$ and $\boldsymbol{NP}$, relativized to an arbitrary set $\boldsymbol{A}$.

The approach has a parallel in recursion theory, where the analogs of $\boldsymbol{P}$ and $\boldsymbol{NP}$ are the recursive sets and recursive enumerable sets. In the domain of recursion theory, many deep results were proved using sophisticated arguments. Furthermore, the following ***relativization principle*** is observed: Proofs are valid relativized to any oracle set.

With this in mind, the following theorem by Baker, Gill and Solovay is truly surprising:

Theorem 1.1 *[BGS75] There are oracle sets* $\boldsymbol{A}$ *and* $\boldsymbol{B}$*, such that*

$$\boldsymbol{P^A \neq NP^A} \text{ and } \boldsymbol{P^B = NP^B}.$$

This result is generally taken as evidence that known mathematical proof techniques will not settle the $\boldsymbol{P}$ vs. $\boldsymbol{NP}$ question.

Relativized separation of the second level of the Polynomial-time Hierarchy was carried out by Baker and Selman [BS79]. It remained a major open problem until last year when finally came the breakthrough of Yao.

Theorem 1.2 *(Yao) There exists an oracle* A *that separates all levels of* PH *and* $PSPACE$.

In this thesis, we show that the separation of $PSPACE$ from the Polynomial time Hierarchy can be achieved with probability one. Thus, with overwhelming probability in all relativized worlds, $PSPACE$ is different from PH. This result gives us renewed confidence that these complexity classes are indeed different.

Another hierarchy of complexity classes has attracted much attention recently [CH86] [Wag85]. This class of problems, the Boolean Hierarchy, is the Boolean closure of the class NP.

There are many interesting properties of this class BH, as discussed extensively in [CH86]. In this thesis we show that it is also more likely that the Boolean Hierarchy extends into an infinite hierarchy.

We will start in the next Chapter making formal definitions. In Chapter 3 we prove that the Boolean Hierarchy is infinite in almost all relativized world. In Chapter 4 the main result of this thesis is proved: with probability one, a random oracle separates $PSPACE$ from the entire Polynomial-time Hierarchy.

Chapter 2

Definitions and Preliminaries

The intuitive notion of an algorithm has been formalized by the model of a Turing machine (TM). A Turing machine is an 8-tuple

$$M = \langle Q, B, \Sigma, \Gamma, \delta, q_0, F, A \rangle,$$

where

Q is a finite set of *states*,

B is a special blank symbol,

Σ is the finite set of *input symbols*, not including B,

Γ is the finite set of allowed *tape symbols*, including Σ and B,

δ is the transition function from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$,

$q_0 \in Q$ is the starting state,

$F \subseteq Q$ is the set of final states,

$A \subseteq F$ is the set of accepting states.

A Turing machine has a finite control, an infinite tape that is divided into cells and a tape head that scans one cell at a time. It works as follows: On an input $x \in \Sigma^*$, M starts in state q_0 with x on its tape and its tape head scanning the first symbol of x. M proceeds by applying δ to the state and scanned symbol. Then M enters the state specified by δ, replacing the scanned tape symbol and moves its head left (L) or right (R).

The process is repeated until a final state is entered. If this state belongs to A, we say the computation accepts x. Otherwise the computation rejects. The machine may never enter a final state, in which case it also rejects.

It is remarkable that this simple model captures all effective computations.

When we are dealing with resource bounded computations, we may always assume that the computation halts. Two types of Turing machines are most interesting to us. If the transition function is single-valued, then we call the machine deterministic. Otherwise it is multi-valued, we call the machine nondeterministic. They are called deterministic and nondeterministic respectively, because upon an input x, the deterministic machine will produce exactly the same computation, while a nondeterministic machine may not. Instead it produces a computation tree where each path represents a valid sequence of moves. A nondeterministic Turing machine accepts input

x *iff* there exists a computation path that leads to accepting x. Intuitively, nondeterminism mirrors the guessing ability of a computation. We comment that nondeterminism is only significant when we consider resource bounded computations, since it may speed up the computation considerably. The set of input that a Turing machine M accepts is denoted by $L(M)$, the language accepted by M. Without loss of generality, we assume that our Turing machines work over a fixed alphabet, e.g. $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, B\}$. Denote Σ^* the set of all finite strings over Σ, then $L(M) \subseteq \Sigma^*$, for all M.

An oracle Turing machine is a Turing machine with a special oracle tape and three distinguished states $q_?$, q_{yes} and q_{no}. When computing with an oracle set a Turing machine may write a string x on its oracle tape, and when it enters the query state $q_?$, depending on $x \in A$ on not, it enters q_{yes} or q_{no}. The language accepted by machine M using oracle A is denoted by $L(M^A)$. If C is a class of languages accepted by a family of machines $\{M_i\}$, the relativized class C^A is defined to be $\{L(M_i^A)\}$. If D is a collection of sets, define $C^D = \bigcup_{A \in D} C^A$.

For resource bounded computations, we define time bounded classes and space bounded classes. We say that a TM M runs in time $T(n)$ if for all input x of length n, M on x takes no more than $T(n)$ many steps before entering a final state.

A nondeterministic TM runs in time $T(n)$ if every computation path on every input x of length n takes no more than $T(n)$ steps.

Similarly we define space bounded computation; here the number of tape cells used in the computation replaces the number of steps. Note that if a TM M runs in time $T(n)$, it also runs in space $T(n)$, since the tape head of the machine can only move one cell at a time.

Define

$$TIME[f(n)] = \{L(M) | M \text{ is a deterministic TM running in time } f(n)\},$$

$$NTIME[f(n)] = \{L(M) | M \text{ is a nondeterministic TM running in time } f(n)\}$$

and

$$SPACE[f(n)] = \{L(M) | M \text{ is a deterministic TM running in space } f(n)\};$$

and define

$$P = \bigcup_{k>0} TIME[n^k + k],$$
$$NP = \bigcup_{k>0} NTIME[n^k + k],$$
$$coNP = \{\overline{L} | L \in NP\},$$

where the overbar denotes set complementation.

$$\Sigma_0^P = P$$

$$\Sigma_1^P = NP$$

$$\Sigma_{k+1}^P = NP^{\Sigma_k^P}$$

$$PH = \bigcup_{k=0}^{\infty} \Sigma_k^P$$

$$PSPACE = \bigcup_{k>0} SPACE[n^k + k]$$

Theorem 2.1 *(Meyer-Stockmeyer) [Sto77]*

$$P \subseteq NP \subseteq \Sigma_2^P \subseteq \cdots \subseteq PH \subseteq PSPACE.$$

If $\Sigma_k^P = \Sigma_{k+1}^P$ *then* $PH = \Sigma_k^P$.

None of the above inclusions is known to be proper, in fact the most outstanding conjecture today in computer science is that all of these inclusions are proper.

Naturally, one may consider introducing nondeterminism in space bounded computations.

Theorem 2.2 *(Savitch) [HU79] Nondeterministic polynomial space bounded computation can be simulated in deterministic polynomial space, i.e.*

$$NPSPACE = PSPACE.$$

Thus nondeterminism does not play a major role in space bounded computations. On the other hand, we believe that nondeterminism in time bounded computation generates a proper hierarchy PH.

One can show that [GJ79]

$L \in \Sigma_k^P \Longleftrightarrow$ For some polynomial time computable predicate P and some polynomial p,

$$L = \{x | \exists^P x_1 \forall^P x_2 \cdots Q^P x_k P(x, x_1, x_2, \ldots, x_k)\},$$

where Q is the quantifier $\exists$ if k is odd, and $\forall$ if k is even; $\exists^P x_i$ and $\forall^P x_i$ denote, respectively, $\exists x_i(|x_i| \leq p(|x|))$ and $\forall x_i(|x_i| \leq p(|x|))$.

Another well known characterization of the classes Σ_k^P is by alternation, introduced by Chandra, Kozen and Stockmeyer[CKS81].

An alternating Turing machine is essentially an nondeterministic TM that has existential stages and universal stages. For languages in the Polynomial time Hierarchy PH, the machines are of a fixed number of alternations and are polynomial time bounded. For example, a Σ_2^P machine is a polynomial time bounded machine that works as follows: Upon an input x, the machine is in the existential stage, where the computation on each nondeterministic path leads to a universal stage. The computation on input x is defined to be accepting if there is a computation path in the existential stage such that all of its following computation paths in the universal stage accepts.

The following Theorem is due to Chandra, Kozen and Stockmeyer:

Theorem 2.3 *For each $k \geq 1$, a language L is in Σ_k^P* iff *it is accepted by a (polynomial time) alternating Turing machine of k alternations.*

Each level in the Polynomial time Hierarchy as well as $PSPACE$ contain problems that are hardest within the class according to the following definitions.

Definition 2.1 *A set* $A \subseteq \Sigma^*$ *is* polynomial time Turing reducible *to a set* $B \subseteq \Sigma^*$, *denoted as* $A \leq_T^P B$, iff $A \in P^B$.

A set $A \subseteq \Sigma^*$ *is* polynomial time many-one reducible *to a set* $B \subseteq \Sigma^*$, *denoted as* $A \leq_m^P B$, iff *there is a polynomial time computable function* f *such that*

$$x \in A \Leftrightarrow f(x) \in B.$$

A set $S \subseteq \Sigma^*$ *in class* C *is* C*-complete with respect to Turing reduction or many-one reduction* iff *for all* $A \subseteq \Sigma^*$, *if* $A \in C$, *then* $A \leq_T^P S$ *or* $A \leq_m^P S$, *respectively.*

The languages mentioned earlier such as SAT, HAM are all NP-complete. The following problem is complete for $PSPACE$:

- QBF: Given a quantified Boolean formula f with no free variables, determine whether f has the value T.

A complete language of a class C is the typical problem the class represents.

Another class of languages has come under intense study recently, the class of Boolean hierarchy [CH86] [Wag85].

Define

$$NP(0) = P$$

$$NP(1) = NP$$

$$NP(2) = D^P = \{L_1 \cap \overline{L_2} | L_1, L_2 \in NP\}$$

$$NP(3) = \{(L_1 \cap \overline{L_2}) \cup L_3 | L_1, L_2, L_3 \in NP\}$$

$$NP(4) = \{L_1 \cap \overline{L_2}) \cup L_3) \cap \overline{L_4} | L_1, L_2, L_3, L_4 \in NP\}$$

$$\vdots$$

$$BH = \bigcup_i NP(i).$$

This hierarchy sits between the first and second levels of the Polynomial time Hierarchy. We have the following characterizations [CH86].

Theorem 2.4 *Any of the following classes is precisely $NP(n)$, where the L_i's are in NP.*

1. *For n even,* $\{(((L_1 \cap \overline{L_2}) \cup L_3) \cdots \cup L_{n-1}) \cap \overline{L_n}\}$,

   and for n odd, $\{((L_1 \cap \overline{L_2}) \cup L_3) \cdots \cup L_n\}$.

2. *For n even,* $\{(((L_1 \cap \overline{L_2}) \cup L_3) \cdots \cup L_{n-1}) \cap \overline{L_n} | L_1 \supseteq L_2 \supseteq L_3 \cdots \supseteq L_n\}$,

   and for n odd, $\{((L_1 \cap \overline{L_2}) \cup L_3) \cdots \cup L_n | L_1 \supseteq L_2 \supseteq L_3 \cdots \supseteq L_n\}$.

3. *For n even,* $\{(L_1 - L_2) \cup \cdots \cup (L_{n-1} - L_n)\}$,

   and for n odd, $\{(L_1 - L_2) \cup \cdots \cup (L_{n-2} - L_{n-1}) \cup L_n\}$.

4. *For n even,* $\{(L_1 - L_2) \cup \cdots \cup (L_{n-1} - L_n) | L_1 \supseteq L_2 \cdots \supseteq L_n\}$,

and for n odd, $\{(L_1 - L_2) \cup \cdots \cup (L_{n-2} - L_{n-1}) \cup L_n | L_1 \supseteq L_2 \cdots \supseteq L_n\}$.

5. $\{L_1 - (L_2 - (\cdots (L_{n-1} - L_n) \cdots))\}$.

6. $\{L_1 - (L_2 - (\cdots (L_{n-1} - L_n) \cdots)) | L_1 \supseteq L_2 \cdots \supseteq L_n\}$.

Theorem 2.5 *The Boolean hierarchy BH is the closure of NP under Boolean operations.*

Theorem 2.6 *The Boolean hierarchy is the closure of NP under polynomial time bounded truth-table reductions [Yes83]:* $BH = \{L | L \leq_{btt}^{P} SAT\}$.

Each level of the Boolean hierarchy has complete languages. In [CH86], we extensively studied many interesting properties of BH. We showed for example that there exists a recursive oracle that separates BH into an infinite hierarchy.

We have seen that relativization can be used to define new and interesting complexity classes. Initiated by Baker, Gill and Solovay in 1975, relativization techniques have been widely used in another quite different setting.

Here is a key observation from recursive function theory: *Almost all proof techniques there relativize!* That is to say that just about all proofs in separating classes found in the rich literature of recursive function theory remain valid with respect to any given oracle set. For example, one can prove that the class of recursive sets $\boldsymbol{REC}$ is different from the class of recursive enumerable sets $\boldsymbol{RE}$. i.e. there are $\boldsymbol{RE}$ sets that are not recursive. (An analogous result in the Polynomial-time Hierarchy would separate $\boldsymbol{P}$ from $\boldsymbol{NP}$.) But for better or worse one observe that the proof remains valid relativized to any given oracle set $\boldsymbol{A}$, i.e. $\boldsymbol{REC}^{\boldsymbol{A}} \neq \boldsymbol{RE}^{\boldsymbol{A}}$.

In view of these observations, the following theorem by Baker, Gill and Solovay is rather surprising.

Theorem 2.7 *[BGS75] There exists an oracle* $\boldsymbol{A}$ *for which* $\boldsymbol{P}^{\boldsymbol{A}} \neq \boldsymbol{NP}^{\boldsymbol{A}}$ *and there exists an oracle* $\boldsymbol{B}$ *for which* $\boldsymbol{P}^{\boldsymbol{B}} = \boldsymbol{NP}^{\boldsymbol{B}}$.

This result is generally taken as evidence that the resolution of the $\boldsymbol{P}$ vs. $\boldsymbol{NP}$ question is hard, and substantially new mathematical tools are needed.

Since then structural complexity theorists have discovered many other relativized separation or collapsing results on various complexity classes. Whenever a relativized result is found, it proves little about the base case, but signifies the contrary to the relativized assertion will be exceedingly hard to prove, if it is at all possible.

We cite another major result on this line by Baker and Selman.

Theorem 2.8 *[BS79] There exists an oracle* $\boldsymbol{A}$ *for which* $\boldsymbol{P}^{\boldsymbol{A}} \neq \boldsymbol{NP}^{\boldsymbol{A}} \neq \Sigma_2^{\boldsymbol{P},\boldsymbol{A}} \neq \Sigma_3^{\boldsymbol{P},\boldsymbol{A}}$.

In this thesis, we study separations of complexity classes that occur in almost all relativized world.

Intuitively, a random oracle set A is generated as follows. For each string $x \in \{0,1\}^*$, we flip a fair coin, and depending on the outcome, we put x in A or not. Formally, we may represent each A by its characteristic function, and then map to a real number in binary expansion $\in [0,1]$. Now we define the probability measure μ on the oracle space to be the Lebesque measure on [0, 1]. The readers may easily verify that the formal definition does represent our intuitive notion of a random set as described.

The study of separation by random oracles was initiated by Bennet and Gill; they showed that

Theorem 2.9 *[BG81] With probability one,* $P^A \neq NP^A \neq \Sigma_2^{P,A}$.

This result is intuitively very appealing. In fact probability one separation was such a strong separation that the two authors went on to propose their *Random Oracle Hypothesis*, which asserts, roughly speaking, that everything happens with probability one about complexity classes does happen in the real world. Unfortunately the Random Oracle Hypothesis was proved to be false in its stated generality [Kur82] [Har85]. Nonetheless, probability one separation remained a favorite topic. Recently, Ambos-Spies has provided a thread of positive evidence in the spirit of the Random Oracle Hypothesis [Amb86].

The following approach is generally taken to prove such separation results. Let L^A be a test language belonging to one class relativized to A, one shows that for all i, $L^A \neq L(M_i^A)$ with probability 1, where $\{M_1^A, M_2^A, \ldots\}$ is an enumeration for the other class.

Lemma 2.1 *[BG81] Let $\mathcal{M}^A = \{M_1^A, M_2^A, \ldots\}$ be an enumeration of some polynomial time bounded relativized language class. Let L^A be a test language such that membership of x in L^A depends only on finitely many strings that lie between $f(|x|)$ and $g(|x|)$, where $f(n)$ and $g(n)$ are two monotone functions that tend to infinity. Then*

$$\exists \varepsilon_0 > 0, \forall i, \mu\{A : L(M_i^A) \neq L^A\} > \varepsilon_0 \Longrightarrow \mu\{A : L^A \notin \mathcal{M}^A\} = 1.$$

Proof Since $\{A : L^A \notin \mathcal{M}^A\} = \bigcap_i \{A : L^A \neq L(M_i^A)\}$ it is sufficient to show that $\mu\{A : L^A \neq L(M_i^A)\} = 1$ for all i. Fix such an i, define E_m to be the set of all A's for which L^A still agrees with M_i^A, up to length m, i.e.

$$E_m = \{A : \forall x[|x| \leq m \Rightarrow L^A(x) = M_i^A(x)\}].$$

It suffices to show that $\mu(E_m) \to 0$, since

$$\lim_{m \to \infty} \mu(E_m) = \mu(\bigcap E_m) = \mu\{A : L^A = L(M_i^A)\}.$$

Clearly $\mu(E_m)$ is monotonically non-increasing. Without loss of generality, assume all $\mu(E_m) \neq 0$.

Fix an m, there is a u, for all $|x| \leq m$, M_i^A and L^A can only use strings of length less than u.

Denote $A^{<u}$ the set of strings in A with length less than u. Clearly if $A^{<u} = B^{<u}$, then

$$A \in E_m \Longleftrightarrow B \in E_m.$$

An initial segment of an oracle set can be represented by a (finite) characteristic string, just as before. There are $U = 2^{2^u - 1}$ many such segments s altogether. Let G_u be the set of all "good" initials :

$$G_u = \{s | \exists A[A^{<u} = s \wedge A \in E_m]\}.$$

Hence,

$$\mu(E_m) = \frac{|G_u|}{U}.$$

Now fix any $s_0 \in G_u$, we show that

$$\mu\{A : L(M_i^A) = L^A \wedge A^{<u} = s_0\} < \frac{1 - \varepsilon_0}{U}.$$

Intuitively, this says a "good" initial segment s_0 will not "help".

Denote $s_0 \circ A$ to be the set obtained from A by replacing its initial segment $A^{<u}$ with s_0. By our assumption on L^A, for u, there exists some m' so large that for all y with $|y| > m'$, membership of y in L^A does not depend on $A^{<u}$. Define an oracle Turing machine M_k in the family $\mathcal{M}$ such

that

$$M_k^A(y) = \begin{cases} L^A(y) & \text{if } |y| \le m' \\ M_i^{s_0 \circ A}(y) & \text{otherwise.} \end{cases}$$

We note that

$$L(M_k^A) = L^A \iff L(M_k^{s_0 \circ A}) = L^{s_0 \circ A}.$$

Hence $\forall s, s'$,

$$\mu\{A : L(M_k^A) = L^A \wedge A^{<u} = s\} = \mu\{A : L(M_k^A) = L^A \wedge A^{<u} = s'\}.$$

On the other hand, for $A^{<u} = s_0$,

$$L(M_i^A) = L^A \Longrightarrow L(M_k^A) = L^A.$$

Hence,

$$\begin{aligned} \mu\{A : L(M_i^A) = L^A \wedge A^{<u} = s_0\} &\le \mu\{A : L(M_k^A) = L^A \wedge A^{<u} = s_0\} \\ &= \frac{\mu\{A : L(M_k^A) = L^A\}}{U} \\ &< \frac{(1-\varepsilon_0)}{U}. \end{aligned}$$

Thus,

$$\begin{aligned} \mu\{A : L(M_i^A) = L^A\} &= \sum_{s \in G_u} \mu\{A : L(M_i^A) = L^A \wedge A^{<u} = s\} \\ &< |G_u| \cdot \frac{(1-\varepsilon_0)}{U} \\ &= (1-\varepsilon_0) \cdot \mu(E_m). \end{aligned}$$

But $\{A : L(M_i^A) = L^A\} = \lim E_m$, therefore, $\exists n > m$, s.t. $\mu(E_n) < (1-\varepsilon_0) \cdot \mu(E_m)$. Thus $\lim \mu(E_n) = 0$. ∎

Chapter 3

Probability One Separation of the Boolean Hierarchy

In this Chapter we prove the following

Theorem 3.1 *A random oracle separates the Boolean hierarchy with probability one.*

The proof idea is a combination of the "sawing" argument discussed in [CH86] and the techniques introduced in [BG81] to handle the NP vs. $coNP$ case.

To illustrate the flavor, let's assume a test language in $coNP^A(3)$ is given,

$$L^A = \{1^n | [(\not\exists x P_1^A(x)) \vee (\exists y P_2^A(y))] \wedge (\not\exists z P_3^A(z)), |x| = |y| = |z| = n\},$$

where P_1^A, P_2^A and P_3^A are some easy oracle dependent properties.

To show that an $NP^A(3)$ language $[L(N_1^A) \cap \overline{L(N_2^A)}] \cup L(N_3^A)$ differs from L^A on a "nontrivial" amount of A, we proceed as follows:

Take a large n. Consider the computation $N_3^A(1^n)$ in two cases:

- Case 1. N_3^A accepts 1^n with a "nontrivial" amount of A. Then *either* it includes a "nontrivial" amount of A with $\exists z P_3^A(z)$, in which case we are done; *or* it includes a "nontrivial" amount of A with $\nexists z P_3^A(z)$. Since N_3 is a nondeterministic polynomial time machine, a single accepting path does not query too many strings. Thus for many z one may alter A so that N_3^A still accepts 1^n but $P_3^A(z)$ holds for some z of length n, resulting a "nontrivial" amount of error.

- Case 2. N_3^A rejects 1^n for all but a "trivial" amount of A. Now we concentrate on those A with the property that $\nexists z P_3^A(z)$, and the previous argument is repeated—we have essentially reduced the case to that of $L(N_1^A) \cap \overline{L(N_2^A)}$ vs. a coD^P property $(\nexists x P_1^A(x)) \vee (\exists y P_2^A(y))$.

So let's consider the computation $N_2^A(1^n)$ in two cases:

- Case 1. N_2^A accepts 1^n with a "nontrivial" amount of A with $\nexists z P_3^A(z)$. Then *either* it includes many A with $\exists y P_2^A(y)$, in which case we are done, since for these A, $1^n \in L^A$ but the $NP^A(3)$ language said no; *or* it includes many A with $\nexists y P_2^A(y)$, in which case we can demonstrate as before a "nontrivial" amount of error.

- Case 2. N_2^A rejects 1^n for all but a "trivial" amount of A with $\nexists z P_3^A(z)$. Now we concentrate on those A with the property that $\nexists z P_3^A(z) \wedge \nexists y P_2^A(y)$. Now in essence we are dealing with the case NP vs. $coNP$. Apply the argument once more and we are done.

Next we will give the formal proof.

Define the characteristic function

$$\chi^A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise.} \end{cases}$$

Define $\chi_k^A(x) = \chi^A(x0101^k) \cdots \chi^A(x01^{|x|}01^k)$. We note that if $x01^i01^k = y01^j01^l$, then $k = l$, $i = j$ and $x = y$.

Thus for A random,

$$\{\chi_k^A(x) | k = 1, 2, 3, \ldots, x \in \Sigma^*\}$$

is a family of independent random variables, with the probability distribution

$$\forall k, x, Prob.(\chi_k^A(x) = z) = \frac{1}{2^{|x|}}$$

for all z, $|z| = |x|$.

Fix any k and length n, we define

$$C_{k,n}^0 \quad = \quad \{A | (\forall x)[\chi_k^A(x) \neq 1^n]\},$$

$$C^1_{k,n} = \{A|(\exists x)[x \neq 1^n \wedge \chi^A_k(x) = 1^n \wedge (\forall y \neq x)[\chi^A_k(y) \neq 1^n]\}.$$

We will abbreviate $C^0_{k,n}$ as C^0_k or C^0 when context permits. Similarly we abbreviate $C^1_{k,n}$ as C^1_k or C^1.

Let $m = 2^n$, we note that

$$\mu(C^0_{k,n}) = \begin{pmatrix} m \\ 0 \end{pmatrix} \left(\frac{1}{m}\right)^0 \left(\frac{m-1}{m}\right)^m = \left(1 - \frac{1}{m}\right)^m,$$

$$\mu(C^1_{k,n}) = \begin{pmatrix} m-1 \\ 1 \end{pmatrix} \left(\frac{1}{m}\right)^1 \left(\frac{m-1}{m}\right)^{m-1} = \left(1 - \frac{1}{m}\right)^m.$$

Thus $\mu(C^0_{k,n}) = \mu(C^1_{k,n}) \rightarrow 1/e$, as $n \rightarrow \infty$.

We also define $Z = \{z||z| = n \wedge z \neq 1^n\}$ for any fixed n, which we endow with the discrete measure ν: $\nu(\{z\}) = 1$ for any $z \in Z$.

Thus the Cartesian product $C^0 \times Z$ and $C^1 \times Z$ are equipped with the a product measure $\mu \times \nu$.

We define

$$\begin{aligned} \varphi : C^0 \times Z &\longrightarrow C^1 \times Z \\ (A, z) &\mapsto (A_z, u) \end{aligned}$$

where $A_z = A \cup \{z01^i01^k | 1 \leq i \leq |x|\}$ and $u = \chi^A_k(z)$.

Note that if $A \in C^0$, then $\forall z \in Z$, $A_z \in C^1$ with the unique z such that $\chi^A_k(z) = 1^n$, and $u = \chi^A_k(z) \neq 1^n$.

We show that φ is a measure isomorphism (one-to-one, onto and measure preserving).

Denote $C_k^0[z \to u] = \{A \in C_k^0 | \chi_k^A(z) = u\}$, and similarly $C_k^1[z \to 1^n] = \{A \in C_k^1 | \chi_k^A(z) = 1^n\}$. We have the following disjoint unions:

$$\begin{aligned}
C^0 \times Z &= \bigcup_{z \in Z} (C^0 \times \{z\}) \\
&= \bigcup_{z \in Z} (\bigcup_{u \in Z} (C^0[z \to u] \times \{z\})), \\
C^1 \times Z &= \bigcup_{z \in Z} (C^1[z \to 1^n] \times Z) \\
&= \bigcup_{z \in Z} (\bigcup_{u \in Z} (C^1[z \to 1^n] \times \{u\})).
\end{aligned}$$

Since the Lebesque measure is shift-invariant, φ maps $C_{k,n}^0[z \to u] \times \{z\}$ 1-1 onto $C_{k,n}^1[z \to 1^n] \times \{u\}$ preserving the measure $\mu \times \nu$. Thus φ is a measure isomorphism from $C^0 \times Z$ to $C^1 \times Z$.

Fix any oracle NP Turing machine N^A, its computation path can query at most polynomially many strings.

$$\begin{aligned}
\text{Let} \quad C_{k,n}^{0,+} &= \{A \in C_{k,n}^0 | N^A(1^n) \text{ accepts } \}, \\
C_{k,n}^{1,+} &= \{A \in C_{k,n}^1 | N^A(1^n) \text{ accepts } \}.
\end{aligned}$$

We have the following fact [BG81].

Lemma 3.1 *For some sequence $\varepsilon_n \to 0$,*

$$\mu(C_{k,n}^{1,+}) \geq (1 - \varepsilon_n) \cdot \mu(C_{k,n}^{0,+})$$

Proof

$$
\begin{aligned}
\mu(C^{1,+}) &= \frac{\mu \times \nu(C^{1,+} \times Z)}{\nu(Z)} \\
&= \frac{1}{\nu(Z)} \cdot \int_{C^1 \times Z} \chi^{C^{1,+} \times Z} d(\mu \times \nu) \\
&= \frac{1}{\nu(Z)} \cdot \int_{C^0 \times Z} \chi^{C^{1,+} \times Z} (\varphi(A, z)) d(\mu \times \nu) \\
&\geq \frac{1}{\nu(Z)} \cdot \int_{C^{0,+} \times Z} \chi^{C^{1,+} \times Z} (\varphi(A, z)) d(\mu \times \nu) \\
&= \frac{1}{\nu(Z)} \cdot \int_{C^{0,+}} d\mu \int_Z \chi^{C^{1,+} \times Z} (\varphi(A, z)) d\nu
\end{aligned}
$$

where the third equality follows from φ being measure preserving, and the fifth equality is by *Fubini's Theorem.*

Since any accepting path of N^A can only be dependent on polynomially many strings of A, it follows that uniformly for all $A \in C^{0,+}_{k,n}$, for asymptotically all $z \in Z$, A_z given by $\varphi(A, z)$ does not alter the accepting computation, i.e. $A_z \in C^{1,+}_{k,n}$.

Thus for some $\varepsilon_n \to 0$,

$$\forall A \in C^{0,+}, \int_Z \chi^{C^{1,+} \times Z} (\varphi(A, z)) d\nu \geq (1 - \varepsilon_n) \cdot \nu(Z).$$

(Here the ε_n is uniform for A.)

Hence, $\mu(C^{1,+}) \geq (1 - \varepsilon_n) \cdot \mu(C^{0,+})$. ∎

Since the family $\{\chi^A_k(x) | k = 1, 2, 3, \ldots, x \in \Sigma^*\}$ is independent, we have the following Corollary.

Corollary 3.1 *For any property $\mathcal{P}$ and any set of indices $\{i_1, \ldots, i_j\} \not\ni k$,*

$$\mu(C_{k,n}^{1,+}(\mathcal{P})) \geq (1 - \varepsilon_n) \cdot \mu(C_{k,n}^{0,+}(\mathcal{P}))$$

where for $C = C_{k,n}^{0,+}$ or $C_{k,n}^{1,+}$, $C(\mathcal{P}) = \{A \in C | \mathcal{P}(\chi_{i_1}^A, \ldots, \chi_{i_j}^A)\}$.

We are ready to prove Theorem 3.1. We will prove for every k, $NP^A(k) \neq coNP^A(k)$ with probability one. Thus

$$P^A \neq NP^A \neq NP^A(2) \neq \cdots \neq NP^A(k) \neq \cdots \neq BH^A$$

with probability one.

Fix any k. Without loss of generality, assume k is an odd number. (By our downward separation result in [CH86], $NP^A(k) \neq coNP^A(k) \Longrightarrow \forall l \leq k, NP^A(l) \neq coNP^A(l)$.) Let's define a test language:

$$\begin{aligned} L^A &= \{1^n | \{\cdots\{(\nexists x_1)[\chi_1^A(x_1) = 1^n] \vee (\exists x_2)[\chi_2^A(x_2) = 1^n]\} \wedge \cdots\} \\ &\quad \wedge (\nexists x_k)[\chi_k^A(x_k) = 1^n]\}. \end{aligned}$$

Clearly $L^A \in coNP^A(k)$, for all A.

In light of Lemma 2.1, we only need to prove for any oracle NP machines $N_1, \ldots, N_k$,

$$\mu\{A | [[L(N_1^A) \cap \overline{L(N_2^A)}] \cup \cdots] \cup L(N_k^A) \neq L^A\} > \frac{1}{12^{k+1}}.$$

For the given machines, we choose a large n so that $\varepsilon_n < 1/4$. Consider the following procedure.

$\mathcal{R}es = \mathcal{P}(\Sigma^*)$, the power set of Σ^*

$s_0 = 0$

For $s = k$ to 1 do

Let $\quad C^{0,+} = \{A \in C^0_{s,n} | N_s^A(1^n) \text{ accepts }\} \cap \mathcal{R}es$

$\quad\quad C^{1,+} = \{A \in C^1_{s,n} | N_s^A(1^n) \text{ accepts }\} \cap \mathcal{R}es$

If $\quad \mu\{A \in \mathcal{R}es | N_s^A(1^n) \text{ accepts }\} > \frac{1}{2^{2s}} \cdot \frac{1}{3^k}$

then $\quad s_0 = s$

If $\quad \mu(C^{0,+}) \leq \frac{1}{2^{2s+1}} \cdot \frac{1}{3^k}$

then quit

else $\quad \mu(C^{1,+}) \geq (1 - \varepsilon_n) \cdot \mu(C^{0,+}) > \frac{3}{4} \cdot \frac{1}{2^{2s+1}} \cdot \frac{1}{3^k}$, quit

else $\quad \mathcal{R}es = \mathcal{R}es \cap \{A | (\not\exists x_s)[\chi_s^A(x_s) = 1^n]\}$

end. {For}

We Observe that when the procedure terminates, either $s_0 > 0$, indicating a breaking point, or $s_0 = 0$, the initial value.

Suppose $s_0 > 0$, then $\mathcal{R}es = \{A | (\forall i > s_0)[(\not\exists x_i)[\chi_i^A(x_i) = 1^n]]\}$. We have been assured that

$$\mu\{A \in \mathcal{R}es | (\exists x_{s_0})[\chi_{s_0}^A(x_{s_0}) = 1^n] \wedge (N_{s_0}^A(1^n) \text{ accepts })\} > \frac{3}{4} \cdot \frac{1}{2^{2s_0+1}} \cdot \frac{1}{3^k},$$

while

$$\mu\{A \in \mathcal{R}es | (\exists s > s_0)[N_s^A(1^n) \text{ accepts }]\} \leq \sum_{s > s_0} \frac{1}{2^{2s}} \cdot \frac{1}{3^k}.$$

A simple calculation shows that

$$\mu\{A \in \mathcal{R}es|\mathcal{E}\} > \frac{1}{12^{k+1}},$$

where $\mathcal{E}$ stands for the event

$$(\exists x_{s_0})[\chi^A_{s_0}(x_{s_0}) = 1^n] \wedge (N^A_{s_0}(1^n) \text{ accepts }) \wedge (\nexists s > s_0)[N^A_s(1^n) \text{ accepts }].$$

But on those A, the $NP^A(k)$ machine differs from L^A.

Now let's assume $s_0 = 0$, then $\mathcal{R}es = \{A|(\forall i, 1 \le i \le k)[(\nexists x_i)[\chi^A_i(x_i) = 1^n]]\}$, and we have

$$\mu(\mathcal{R}es) \approx \left(\frac{1}{e}\right)^k,$$

$$\mu\{A \in \mathcal{R}es|(\exists s, 1 \le s \le k)[N^A_s(1^n) \text{ accepts }]\} \le \sum_{s=1}^{k} \frac{1}{2^{2s}} \cdot \frac{1}{3^k} < \frac{1}{3^k}.$$

Hence, the $NP^A(k)$ machine again differs from L^A for a set of A of measure greater than

$$\mu(\mathcal{R}es) - \frac{1}{3^k} > \frac{1}{12^k}.$$

The proof is completed.

Chapter 4

Probability One Separation of PH from PSPACE

As complexity measures, the relationship between time and space has been one of the primary concerns in complexity theory research. Despite convincing heuristic evidence and persistent effort, no proof is in sight for separating the Polynomial time Hierarchy from Polynomial Space.

We noted in Chapter 2 that the entire Polynomial time Hierarchy PH is contained in $PSPACE$. Savitch's theorem tells us that for polynomial space bounded computation, nondeterminism does not generate anything new. However, for polynomial time bounded computation we have a whole hierarchy PH sitting on top of P using repeated nondeterminism.

A proof that $PH \neq PSPACE$ would be an extremely strong separation of time and space. We show that ***PH*** is properly contained in ***PSPACE*** in almost all relativized worlds .

Theorem 4.1 *(Main) With probability one, a random oracle separates* ***PSPACE*** *from the entire Polynomial time Hierarchy.*

The proof is organized as follows:

1. Use the alternating Turing machine model [CKS81] to reduce the problem to a Boolean circuit computation problem.
2. Employ certain probabilistic and game theoretic techniques to crack a shallow circuit.
3. Inductively prove a theorem on the general case which is then adapted to resolve the problem on circuit computation in 1.

The present work is a continuation of the work pioneered by Furst, Saxe, Sipser and Yao. [FSS84] [Sip83] [Yao85].

In 1978 Furst, Saxe and Sipser showed that the Boolean function ***Parity*** (see definition below) cannot be computed in fix depth polynomial size Boolean circuit. They also observed that an exponential poly-logarithmic lower bound would establish the existence of an oracle separating ***PSPACE*** from the Polynomial Hierarchy. Later Sipser extended this work in [Sip83]. In 1985 finally came the breakthrough with Yao's following theorem, which influenced our research immensely.

Theorem 4.2 *(Yao) There exists an oracle A such that*

$$P^A \neq NP^A \neq \Sigma_2^{P,A} \neq \cdots \neq PH^A \neq PSPACE^A$$

Our strong separation result is obtained by looking at *how much* error is present in the supposed circuit computation (instead of the existence of a *single* error). This question is interesting in its own right in the theory of circuit computation. As a nice byproduct of the proof of this strong separation we have the following corollary.

Corollary 4.1 *Fixed depth Boolean circuits with a bound of the form $exp(n^\lambda)$ on the size, for some λ, make asymptotically 50% error computing the Boolean function Parity.*

4.1 Initial Reductions and Techniques

We proceed with some definitions. Let X be the set of n Boolean variables $\{x_1, x_2, \ldots, x_n\}$. A $\Sigma_{1,n}$-formula (circuit) is a sum of the form $\sum_k \overline{x_{i_k}} + \sum_k x_{j_k}$, where $x_{i_k}, x_{j_k} \in X$. Without loss of generality, we assume that the variables are distinct. The number of literals is its size. A $\Pi_{1,n}$-formula (circuit) is the negation of a $\Sigma_{1,n}$-formula, with the same size, i.e., a product of the form $\prod_k \overline{x_{i_k}} \cdot \prod_k x_{j_k}$.

For $k > 1$, a $\Sigma_{k,n}$-formula (circuit) H is a sum of $\Pi_{k-1,n}$-formulae, $\sum_i G_i$, with $size(H) = \sum_i size(G_i)$. A $\Pi_{k,n}$-formula is the negation of a

$\Sigma_{k,n}$ -formula, with the same size.

The bottom fan-in (bfi) of a Boolean circuit is the largest of sizes of depth 1 subcircuits.

For any $\Pi_{2,n}$-formula G, $G = \prod_{i=1}^{l} C_i$, where

$$C_i = \overline{x_{i_1}} + \overline{x_{i_2}} + \cdots + \overline{x_{i_s}} + x_{i_{s+1}} + x_{i_{s+2}} + \cdots + x_{i_{s+t}} \text{ and } s, t \geq 0.$$

We denote $J_{i-} = \{i_1, i_2, \ldots, i_s\}, J_{i+} = \{i_{s+1}, i_{s+2}, \ldots, i_{s+t}\}$ and $J_i = J_{i-} \cup J_{i+}$.

A (partial) assignment of X is an n-tuple $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in \{0, 1, *\}^n$. If $\sigma \in \{0, 1\}^n$, then σ is a total assignment. Let F be a Boolean function on X, then $F|_\sigma$ denotes the Boolean function after the assignment σ, i.e., assign $x_i = 0, 1$ or unassigned, if $\sigma_i = 0, 1$ or $*$.

To generalize a bit, we also consider random assignments of X. For $0 \leq p \leq 1$, let $\mathcal{R}_p$ denote the probability space $\{0, 1, *\}^\omega$ with a product measure ν, where (independently) for each coordinate i, $1 \leq i < \omega, \nu\{(a_1, \ldots, a_i, \ldots) | a_i = \alpha\} = (1 - p)/2$, if $\alpha = 0$ or 1; and p if $\alpha = *$. A random assignment is denoted as $A = (a_1, \ldots, a_i, \ldots)$.

If F is a Boolean function on free variables $\{x_{i_1}, \ldots, x_{i_s}\} \subseteq X$, then a random assignment A taken from $\mathcal{R}_p$ (denoted as $A \in \mathcal{R}_p$), assigns the variables x_{i_j} to 0, 1 or unassigned, according to a_j of A.

We denote $F \|_A$ the Boolean function resulted from the assignment.

Similarly we define one-sided random assignments. A random B taken from $\mathcal{R}_p^+$ (denoted as $B \in \mathcal{R}_p^+$) assigns independently to each x_{i_j} in F to 1 with probability p , and unassigned with probability $1-p$, respectively. $\mathcal{R}_p^-$ is defined in the same way. Note that all random assignments only affect free variables, when they are applied to a formula.

Consider a sequence of random assignments $R_1, \ldots, R_s$. $F \|_{R_1,\ldots,R_s}$ is defined to be $(F \|_{R_1,\ldots,R_{s-1}}) \|_{R_s}$. The successive random assignments only act on the variables left untouched by previous assignments. We also denote $R_1 \cdots R_s$ as $A = (a_1, \ldots, a_j, \ldots)$, where $a_j = 0, 1$ or $*$, depending on $R_1 \cdots R_s$ assigns x_j to 0, 1 or unassigned, respectively.

Note however, a partial assignment σ is applicable to $F \|_{R_1,\ldots,R_s}$ *iff* σ assigns 0 or 1 to only those variables that are $*$-valued by $R_1 \cdots R_s$. In which case, we denote $(F \|_{R_1,\ldots,R_s})|_\sigma$ as the resulted function.

Fix an alphabet {0, 1}. Define the parity function $Parity_n$

$$Parity_n(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i (\mathrm{mod}\ 2).$$

We will consider circuit computation in relation to the parity function $Parity_n$. The parity function is chosen for the following property: the value of $Parity_n$ is **vitally** dependent on every single variable x_i.

For $A \subseteq \{0,1\}^*$, define the parity language

$$Parity^A = \{1^n |\ \text{there are odd number of strings of length } n \text{ in } A\ \}.$$

Clearly, we have $Parity^A \in PSPACE^A$, for all A.

We aim to prove that $\mu\{A|Parity^A \notin PH^A\} = 1$. Surely this implies that a random oracle separates $PSPACE$ from the entire Polynomial time Hierarchy, with probability one.

There are only countably many levels $\Sigma_k^{P,A}$ in PH^A. Each level Σ_k^P has a recursive enumeration as the class of languages accepted by polynomial time alternating machines at that level [CKS81]. Let $M_1, M_2, \ldots$ be an enumeration of Σ_k^P alternating machines, then it is sufficient to show that

$$\forall i, \mu\{A|Parity^A \neq L(M_i^A)\} = 1.$$

According to Lemma 2.1, we only need to show that for each level k,

$$\exists \varepsilon_0 > 0, \forall i, \mu\{A : Parity^A \neq L(M_i^A)\} > \varepsilon_0.$$

Now we reduce alternating machines to Boolean circuits. This is essentially the same as in [FSS84].

For a fixed alternating machine with oracle M_i^A at level Σ_k^P, consider its computation on 1^m. We claim that it is always possible to postpone the queries of strings. The trick is to guess the answer and verify them in a later stage. For example, in an existential stage, whenever a query is needed, we instead guess the answer and proceed until the succeeding universal stage. At the beginning of this universal stage we verify with the oracle the guesses in the previous existential stage; and if any of the guesses is wrong, we abort this path, otherwise we proceed. Similarly we may delay the queries of a universal stage and verify in the succeeding existential stage. This time if any guess was wrong we simply accept. It is easily shown that this transformation does preserve the notion of acceptance by alternating Turing machine. By adding one more level of the alternation using the same trick, we may obtain an equivalent polynomial time bounded alternating machine, that only queries at the bottom level and only queries once on every computation path. Thus, the computation tree structure is *independent* of the oracle, and therefore, can be frozen to yield a Σ_{k+1}-circuit, G.

The set of input Boolean variables of G corresponds to those strings that are queried by the modified alternating machine, on 1^m. (A queried string is in A *iff* the corresponding Boolean variable is set to be true.) Empirically, there should be precisely 2^m input variables, corresponding to 2^m strings of length m. This is because $1^m \in^? Parity^A$ is independent of any string of length unequal to m; furthermore, fix any string x of length m, it does depend on x whether 1^m belongs to the parity language, $Parity^A$. Does the machine have to query precisely those strings of length m, no more and no less? We show that this is indeed the case, without loss of generality.

Suppose then for some x, $|x| = m$, and x is never queried. Then clearly the machine is making error precisely half the probability (under μ) at length m. That is enough.

Now suppose all x's with $|x| = m$ are queried, but so are $y_1, \ldots, y_l$ of length unequal to m.

Consider all 2^l many possible assignments σ for the y_i's. For any σ, consider the $\Sigma_{k+1,2^m}$-circuit resulted $G|_\sigma$. Pick the best σ_0, in the sense that $G|_{\sigma_0}$ makes least error for parity. Clearly the original G makes no less error than $G|_{\sigma_0}$, percentage wise. Formally speaking,

$$\begin{aligned}\mu\{A|Parity^A(1^m) \neq M_i^A(1^m)\} &\geq \text{ error rate of } G|_{\sigma_0}\text{for parity} \\ &\equiv \frac{card.\{\tau \in \{0,1\}^n : G|_{\sigma_0}|_\tau \neq Parity_n|_\tau\}}{2^n}.\end{aligned}$$

where $n = 2^m$.

A remark on the size. Since $M_i(1^m)$ runs at most $p(m)$ steps, for some polynomial $p(\cdot)$, the size of the circuit is bounded by exponential polylog in n, $\exp(O(p(\log(n))))$, where n is the input size to the circuit.

We have shown that in order to prove our main theorem, the following theorem on Boolean circuit computation would suffice.

Theorem 4.3 *For all $k \geq 2$, there is a sequence α_n, where $\alpha_n \to 1/2$ as $n \to \infty$, such that all depth k Boolean circuits with n input and size bounded by $exp(n^{1/4(k+1)})$, when computing $Parity_n$, make error on more than $\alpha_n \cdot 2^n$ of all 2^n many input.*

We now define a notion that is central to our exposition.

A Boolean function is given, G. Consider the following class of 2-man games, played between a master and a player: the general mode of the game is a cycle; the master gives a Boolean variable (unassigned so far), and asks the player to assign it. The player may assign it either 0 or 1. The master may repeat the cycle zero or more times, until he declares the end of the game. The rule dictates that when the master declares the end of the game, the assignment made by the player so far must make G a constant.

A Boolean function G is *k-monochromatic iff* there is a 2-man game of the defined class, in which the master has a winning strategy in the following sense: the master can declare the end after no more than $\lceil k \rceil$ many variables are assigned.

To put it differently, it is guaranteed that, no matter how the player plays, the master can force the function G to be constant, after at most $\lceil k \rceil$ variables are assigned.

Here we emphasize two points:

1. The k variables are *not* given out in a batch; rather the master makes up his mind as to which variable to give next, depending on how the player assigned so far.

2. Even in a play following a winning strategy, the master is (technically) *not* required to declare the end of the game at the earliest possible moment.

We finish the section with the following Lemma that essentially states that under a *monochromaticity* condition, an *and* of *or*'s and an *or* of *and*'s are interchangeable.

Lemma 4.1 *If G is k-monochromatic, then G is a constant, or equivalent to a $\Sigma_{2,n}$-circuit (as well as a $\Pi_{2,n}$-circuit) with $bfi \leq k$.*

Proof The proof is by induction on k. The case $k = 0$ is trivial. Suppose $k > 0$, and the Lemma is true for all values less than k. Let G be k-monochromatic, but not $(k-1)$-monochromatic. Let's play the game; suppose x_i is the first variable the master puts out. Then $G = [x_i \wedge G|_{x_i=T}] \vee [\overline{x_i} \wedge G|_{x_i=F}]$, where T stands for *True* and F stands for *False*. Now both $G|_{x_i=T}$ and $G|_{x_i=F}$ are $(k-1)$-monochromatic; we apply the inductive hypothesis once more, the result follows. ∎

The rest of this chapter is devoted to the proof of Theorem 4.3. We will start with depth 2 circuits in the next section.

4.2 Depth Two Circuits

We wish to prove a theorem concerning depth 2 Boolean circuits.

Theorem 4.4 *Fix $0 < \varepsilon < 1/5$, then there exists a constant C, such that, for any $G \in \Pi_{2,n}$ with $bfi \leq n^{\varepsilon}$, take a random $Q_1 \in \mathcal{R}^{-}_{1-q}$ and a random*

$Q_2 \in \mathcal{R}^+_{1-q}$, where $0 \leq q \leq n^{-1.05\epsilon}$, then $G\,||_{Q_1Q_2}$ is n^ϵ-monochromatic, with probability $1-\varepsilon_n$, where $\varepsilon_n \leq Ce^{-n^\epsilon}$.

The idea of the proof is as follows: We will define a 2-man game, with respect to the circuit $G\,||_{Q_1Q_2}$, which we claim the master most probably has a winning strategy (cf. Section4.1). The game is designed so that each play creates a *record* as to how the game was played. In the rare case $G\,||_{Q_1Q_2}$ is not n^ϵ-monochromatic, the *record* will be "large". Now we define another procedure, Recording (Q_1Q_2, *record*), which will reproduce the game play. On the other hand, given a "large" *record*, a random assignment R *survives* the procedure Recording (R, *record*) is so unlikely, that even summing up the probability over all "large" *records*, it is still of measure near 0.

4.2.1 The Game and the Recording

We denote an assignment Q_1Q_2 as $A = (a_1, \ldots, a_i, \ldots)$, as in Section 4.1. Suppose $G = C_1 \wedge \ldots \wedge C_l$. Let $N = \{1, \ldots, \lceil n^\epsilon \rceil\}$, $\mathcal{X} = \{\langle Z, s\rangle | Z \subseteq N, s \in \{0,1\}^{|Z|}\}$. Define $\|\ \langle Z, s\rangle\ \| = card.(Z)$. A record S is a finite sequence $\langle X_1, \ldots, X_l\rangle$, where $X_i \in \mathcal{X}$. Define the norm $\|\ S\ \| = \sum_{i=1}^{l} \|\ X_i\ \|$.

Intuitively, when a record element $X_i = \langle Z, s\rangle$ is generated in certain round of the game, Z codes the variables to be assigned and s codes the assignment made by the player, in that round.

The coding scheme in Z is an indirect addressing. Specifically, if $J =$

$\{\boldsymbol{i}_1, \boldsymbol{i}_2, \ldots, \boldsymbol{i}_\alpha\}$, and $\boldsymbol{Z} = \{z_1, z_2, \ldots, z_\beta\}$, where $\alpha, \beta \geq 0, 1 \leq \boldsymbol{i}_1 < \ldots < \boldsymbol{i}_\alpha, 1 \leq z_1 < \ldots < z_\beta$, then $\boldsymbol{Z}$ codes the subset of $\boldsymbol{J}$:

$$\{\boldsymbol{i}_{z_1}, \ldots, \boldsymbol{i}_{z_\beta}\}, \text{ if } z_\beta \leq \alpha.$$

We denote this set as $\boldsymbol{J} \oplus \boldsymbol{Z}$. If $\boldsymbol{Z} = \emptyset$, then $\boldsymbol{J} \oplus \boldsymbol{Z} = \emptyset$. If $z_\beta > \alpha$, $\boldsymbol{J} \oplus \boldsymbol{Z}$ is undefined.

Conversely for $\Delta = \{\boldsymbol{i}_{z_1}, \ldots, \boldsymbol{i}_{z_\beta}\} \subseteq \boldsymbol{J}$, we denote,

$$\boldsymbol{J} \ominus \Delta = \{z_1, \ldots, z_\beta\}.$$

Note that $\boldsymbol{J} \ominus \emptyset = \emptyset$. Clearly for $\Delta \subseteq \boldsymbol{J}, \boldsymbol{J} \oplus (\boldsymbol{J} \ominus \Delta) = \Delta$.

Our game is played in rounds. The master executes the program, and in certain rounds, he asks the player to assign a few Boolean variables. Then the master continues, until the program halts. When the program halts, it halts in "result" or in "abort".

The procedure Recording is similar; for technical reasons, we first present Recording. A record $S = \langle \boldsymbol{X}_1, \ldots, \boldsymbol{X}_l \rangle$ is given. There are four essential variables $\Theta, \boldsymbol{Y}^+, \boldsymbol{Y}^*$ and $\boldsymbol{N}^*$, respectively representing our knowledge about the assignment $\boldsymbol{A}$ at any given point in the execution. (More accurately, our knowledge about $\boldsymbol{A}$ for which we can be *forced* to acknowledge.)

Procedure Recording $(\boldsymbol{A}, S)$

0 $\quad \Theta, \boldsymbol{Y}^+, \boldsymbol{Y}^*, \boldsymbol{N}^* := \emptyset$; $\boldsymbol{t} := 0$; $\boldsymbol{List} := [C_1, \ldots, C_l]$;

Repeat

1 **if** $List = \emptyset$ **then** case 1 : $t < l \Rightarrow$"abort";

case 2 : $t \geq l \Rightarrow$"result"

fi

2 let C_i be on the top of $List$

3 **if** $(\exists u \in J_{i-}, a_u = 0)$ **then** $\Theta := \Theta \cup \{J_{i-}\}$, delete C_i from $List$

4 **else** [critical round]

5 $t := t + 1$

6 $F := J_i - (Y^+ \cup Y^* \cup N^*)$

7 get $X_t = \langle Z, s \rangle$ from S, "abort" if nonexistent

8 $D := F \oplus Z$, "abort" if undefined

9 **if** $(D \neq \{j \in F | a_j = *\})$ **then** "abort"

10 **else** $Y^+ := Y^+ \cup \{u \in F | a_u = 1\}$

$\Theta := \Theta \cup \{\{u\} | u \in F, a_u = 0\}$

$Y^* := Y^* \cup \{u \in D | s$ assigns x_u to $1\}$

$N^* := N^* \cup \{u \in D | s$ assigns x_u to $0\}$

fi

11 **if** $(D = \emptyset)$ **then**

12 **if** $(\forall u \in J_{i+} - N^*, a_u = 0)$ **then** "abort" **fi**

fi

13 Delete any C_k from $List$ with

$$J_{k-} \cap N^* \neq \emptyset \text{ or } J_{k+} \cap (Y^+ \cup Y^*) \neq \emptyset$$

fi

End [Repeat]

Some properties of Recording are easily verified. Define $List_{out}$ to be the set of j such that C_j has been deleted from $List$. (In the following, c.r. is a short hand for "critical round"). We have:

Lemma 4.2 **(1)** *Θ, Y^+, Y^*, N^* and $List_{out}$ are monotonically non-decreasing.*

Every time the **Repeat** *loop is entered, the following are true:*

(2) $\forall K \in \Theta, \exists u \in K, a_u = 0$.

(3) $\forall u \in Y^+, a_u = 1$.

(4) $\forall u \in Y^* \cup N^*, a_u = *$.

(5) *$t = \#$ of c.r. completed so far.*

(6) *$\forall j \in List, J_{j-} \cap N^* = \emptyset$ or $J_{j+} \cap (Y^+ \cup Y^*) = \emptyset$.*

(7) *$\forall j \in List_{out}, J_{j-} \in \Theta$ or $J_{j-} \cap N^* \neq \emptyset$ or $J_{j+} \cap (Y^+ \cup Y^*) \neq \emptyset$.*

(8) $Y^* \cap N^* = \emptyset$.

Proof A straightforward check. ∎

Next we define our Game. The Game is very much like Recording, except the record S is being produced as we go along, one slot per critical round. Specifically,

- In the initialization part (line 0), add $S := \emptyset$.
- Change line 1 to: **if** $\boldsymbol{List} = \emptyset$ **then** "result" S.
- Change lines 7 and 8 to:

  Create $\boldsymbol{X_t} : \boldsymbol{Z} := \boldsymbol{F} \ominus \{j \in \boldsymbol{F} | \boldsymbol{a_j} = *\}$;

  $\boldsymbol{D} := \boldsymbol{F} \oplus \boldsymbol{Z} (= \{j \in \boldsymbol{F} | \boldsymbol{a_j} = *\})$;

  for $j \in D$ **do** let the player assign x_j, and record the assignment in S with a binary string s of length $|D|$ (in the obvious way).

  $\boldsymbol{X_t} := \langle \boldsymbol{Z}, \boldsymbol{s} \rangle$.

Let ρ be the assignment made by the player: $\rho_d = 0, 1$ or $*$; if $\boldsymbol{d} \in \boldsymbol{N^*}, \boldsymbol{Y^*}$ or otherwise.

We wish to prove the following:

Lemma 4.3 *The Game will eventually halt. When the Game halts,* $\boldsymbol{G} \,||_A\, |_\rho \equiv 1$ *(at line 1), or* $\equiv 0$ *(at line 12). Furthermore, let* S *be the record it created, then Recording* $(\boldsymbol{A}, S)$ *will run precisely the same as Game* $(\boldsymbol{A}, S)$*, until halting.*

Proof We claim that every completed round of the Game either deletes a clause or assigns a variable. The only nontrivial case is in a c.r. with $D = \emptyset$. In the case of $D = \emptyset$ and the round is completed, the condition at line 12 must be false. Thus $\exists t \in J_{i+} - N^*, a_t = 1$, (there is no $*$ in F), where i is the index of the current clause C_i. But then C_i must be deleted at line 13.

Therefore the Game will eventually halt. Let S_0 be the record created when the Game halts.

We prove by induction, that Recording (A, S_0) will reproduce the game play. Suppose they both enter a new round with all the variables having the same value. (This is certainly true initially.) Also assume $t =$ the length of S_0 constructed so far in the game.

If $List = \emptyset$, then the game halts in result. Since $t = l$, the length of S_0, Recording (A, S_0) will also halt in result.

Suppose $List \neq \emptyset$. Then they get the same C_i, same condition at line 3 (same A!). If the condition is satisfied, then the induction is completed. Suppose not. They come to line 7. The Game creates the next X_t. Since it is never altered later in the Game, it is what Recording obtains from S_0. From the way X_t is created in the Game, Recording will not halt at line 7, 8 and 9. Now for the rest of this round they have the same code. The induction is completed.

We have proved that Recording will reproduce the game play, and hence Lemma 4.2. applies to the procedure Game. In fact we proved something more, namely, the Game can only halt at line 1 or line 12.

Followed from Lemma 4.2, (4) and (8), ρ is a valid assignment to $G\ ||_A$. If the Game halts at line 1, then $G\ ||_A\ |_\rho \equiv 1$, by Lemma 4.2, (2), (3) and (7). If the Game halts at line 12, then we claim that $C_i\ ||_A\ |_\rho \equiv 0$, hence $G\ ||_A\ |_\rho \equiv 0$.

By (6) of Lemma4.2, $F = [J_{i-} - (Y^+ \cup Y^*)] \cup [J_{i+} - N^*]$, at line 6. Since $D = \emptyset$, Y^* and N^* are unchanged at line 10. Clearly the only way to satisfy C_i is in F. But if $t \in J_{i-} - Y^*, a_t = 1$, by lines 3,9 and 11; and if $t \in J_{i+} - N^*, a_t = 0$, by line 12. So $C_i\ ||_A\ |_\rho \equiv 0$. ∎

Let $\mathcal{A} = \{A : G\ ||_A$ is *not* n^ϵ-monochromatic$\}$. For a record S, let $\mathcal{A}[S] = \{A :$ Recording(A, S)"results"$\}$.

If $A \in \mathcal{A}$, then for any game, in particular for our Game, the master has no winning strategy. Hence there is a play in which the player assigned $\lceil n^\epsilon \rceil$ many variables and still, the circuit is not constant.

Being not constant 0, there is a satisfying assignment σ. Now for the rest of the Game, the player adopts the following strategy: assign any new variable according to σ. Since this strategy keeps the circuit satisfiable and the game eventually halts, the Game must halt with the circuit equal to constant 1. Hence, the Game "results" with some S, where $\|\ S\ \| > \lceil n^\epsilon \rceil$.

Therefore

$$\mathcal{A} \subseteq \bigcup \mathcal{A}[S],$$

where the union is over all S, with $\| S \| > \lceil n^{\epsilon} \rceil$.

4.2.2 A Probability Analysis

In this section, we will focus on Recording (A, S). For a fixed S with $\| S \| > \lceil n^{\epsilon} \rceil$, we consider the probability that Recording (A, S) results, where $A = Q_1 Q_2$, $Q_1 \in \mathcal{R}^{-}_{1-q}$ and $Q_2 \in \mathcal{R}^{+}_{1-q}$.

Define $\mathcal{A}^{m\gamma} = \{A :$ Recording(A, S) will come to its mth c.r. with $(\Theta,$ Y^{+}, Y^{*}, $N^{*}) = \gamma\}$, $\Gamma^{m} = \{\gamma : \mathcal{A}^{m\gamma} \neq \emptyset\}$, $\mathcal{A}^{m} = \bigcup_{\gamma \in \Gamma^{m}} \mathcal{A}^{m\gamma}$.

We first derive a condition for $A \in \mathcal{A}^{m\gamma}$.

Lemma 4.4 *For any $\gamma = (\gamma^{\Theta}, \gamma^{+}, \gamma^{Y}, \gamma^{N}) \in \Gamma^{m}$, there exists i_{γ}, such that $A \in \mathcal{A}^{m\gamma} \Longleftrightarrow A$ satisfies the following conditions:*

(I) $\forall K \in \gamma^{\Theta}$, $\exists u \in K$, $a_u = 0$.

(II) $\forall u \in \gamma^{+}, a_u = 1$.

(III) $\forall u \in \gamma^{Y} \cup \gamma^{N}, a_u = *$.

(IV) $\forall u \in J_{i_{\gamma}-}, a_u \neq 0$.

Let's prove the following Lemma first: Pick any $A^0 \in \mathcal{A}^{m\gamma}$.

Lemma 4.5 *A satisfies Conditions (I), (II) and (III)* $\Longrightarrow$ *Recording for* A^0 *and* A *will run precisely the same (with all the variables* $\Theta, Y^+, Y^*, N^*, List$*, and* s *being the same at corresponding moments) up to line 1 of the* m*th c.r. of* A^0.

Proof By induction. Suppose they are at line 1 of the m_0th round of A^0 (including m' c.r. and $m' < m$), and so far they are all the same. (Trivially true for the base case $m_0 = 1$.)

If this is the m th c.r. for A^0, then the induction is completed. Suppose it is not. Hence A^0 will complete this round without halting. Thus $List \neq \emptyset$ and they pick the same C_i.

If A^0 satisfies the condition at line 3, then $J_{i-} \in \gamma^{\Theta}$, since the mth c.r. of A^0 is yet to come. By (I), A satisfies the same condition at line 3.

If A^0 fails the condition at line 3, this is a c.r. of A^0, but not the mth yet. A^0 will successfully record all $u \in J_{i-} \cup J_{i+}$ in (Θ, Y^+, Y^*, N^*) , which will later become γ. In particular, $\forall u \in J_{i-}$, $u \in \gamma^+ \cup \gamma^Y \cup \gamma^N$. By (II) and (III), A must also fail the condition at line 3.

Hence either they both finish the current round at line 3, in which case the induction is completed; or they both advance to line 4. Suppose then this is a c.r. for both. They must find (the same) D well-defined. As we noted, A^0 will record all $u \in J_{i-} \cup J_{i+}$ which will be shown in γ.

In particular, by (I), (II) and (III), A must also find D to be precisely the set of $*$'s in F, and thus update Y^* and N^* exactly the same. Similarly, A must update Θ and Y^+ the same way as A^0 does, by (I) and (II).

Now if $D \neq \emptyset$, we are done. If $D = \emptyset$, then A^0 will find the condition at line 12 to be false. i.e., $\exists t_0 \in J_{i+} - N^*, a^0_{t_0} \neq 0$. But $D = \emptyset \Rightarrow a^0_{t_0} = 1$. Hence $t_0 \in Y^+$, which is the same for both A and A_0. Hence $a_{t_0} = 1$ as well. Therefore A won't halt there. The induction is completed. ∎

Now we prove Lemma 4.4 . Pick $A^0 \in \mathcal{A}^{m\gamma}$ and run Recording (A^0, S), let C_{i_γ} be the clause under question in its mth c.r.

$\Longrightarrow$ Since $A \in \mathcal{A}^{m\gamma}$, A satisfies (I), (II)and (III). By Lemma 4.5, A and A^0 will reach line 1 of the mth c.r. of A_0, with all the variables the same. Since A^0 comes to line 4, $\boldsymbol{List} \neq \emptyset$, which is the same for A; so they both pick up C_{i_γ}. Since this is the m th c.r. for A^0, A^0 will fail the condition at line 3, and enter its c.r. with Θ unchanged. Thus γ^Θ is the common value for Θ when A and A^0 entered the current round at line 1.

If A were to satisfy the condition at line 3, then this is not a c.r. for A, and its mth c.r. is yet to come. Since $J_{i_\gamma -}$ is now added to Θ by A, $J_{i_\gamma -} \in \gamma^\Theta$. In other words, $J_{i_\gamma -} \in \Theta$ when they entered at line 1. But then A^0 must have satisfied the condition at line 3. A contradiction.

Therefore A satisfies (IV).

$\Longleftarrow$ Again by Lemma 4.5 , we can assume they arrive at line 1 of the m th c.r. of A^0, with exactly the same history.

$A^0 \in \mathcal{A}^{m\gamma} \Rightarrow$ ***List*** $\neq \emptyset$ and A picks up C_{i_γ}. Then (IV) says that this is also a c.r. for A. Since this is the m th c.r. for A^0, $A^0 \in \mathcal{A}^{m\gamma}$, and so far they are the same, this is also the m th c.r. for A, with $(\Theta, Y^+, Y^*, N^*) = \gamma$. Hence $A \in \mathcal{A}^{m\gamma}$. ▮

Now we are ready to estimate the probability $Pr.(\mathcal{A}[S])$. Let E^m denote the event that Recording (A, S) completes its m th c.r. without halting.

$$\begin{aligned} Pr.(\mathcal{A}[S]) &\leq Pr.(\mathcal{A}^1) \cdot \Pi_{1 \leq m < l} Pr.(A \in \mathcal{A}^{m+1} | A \in \mathcal{A}^m) \cdot Pr.(E^l | A \in \mathcal{A}^l) \\ &\leq [\Pi_{1 \leq m < l} Pr.(E^m | A \in \mathcal{A}^m)] \cdot Pr.(E^l | A \in \mathcal{A}^l) \\ &= \Pi_{1 \leq m \leq l} Pr.(E^m | A \in \mathcal{A}^m). \end{aligned}$$

We show the following:

Lemma 4.6 $\forall \gamma \in \Gamma^m$,

$$\begin{aligned} Pr.(E^m | A \in \mathcal{A}^{m\gamma}) &\leq q^{\|X_m\|} \ \textit{if} \ \| X_m \| \neq 0 \\ &\leq q \cdot n^{\varepsilon} \ \textit{otherwise.} \end{aligned}$$

Clearly, Lemma 4.6 implies the same bound for $Pr.(E^m | A \in \mathcal{A}^m)$, since it can be estimated as:

$$\sum_{\gamma \in \Gamma^m} Pr.(E^m | A \in \mathcal{A}^{m\gamma}) \cdot Pr.(A \in \mathcal{A}^{m\gamma} | A \in \mathcal{A}^m),$$

and

$$\sum_{\gamma \in \Gamma^m} Pr.(A \in \mathcal{A}^{m\gamma} | A \in \mathcal{A}^m) = 1.$$

Hence,

Corollary 4.2 *$Pr.(\mathcal{A}[S]) \leq q^{\|S\|} \cdot (qn^{\epsilon})^{l'}$, where, $l' = \#$ of X_i in S with $\| X_i \| = 0$.*

We use Lemma 4.4 to prove Lemma 4.6.

Proof (of Lemma 4.6) Assume $\| X_m \| \neq 0$. Consider a random assignment taken from $\mathcal{R}^-_{1-q}$, followed by one from $\mathcal{R}^+_{1-q}$, on the variables in $F = J_{i_\gamma} - (\gamma^+ \cup \gamma^Y \cup \gamma^N)$. We refer to the procedure Recording. In order to survive the m th critical round, we must have $D = F \oplus X_m = \{j \in F | a_j = *\}$. Clearly the conditions on the random assignment by Lemma 4.4 can be most favorably strengthened to that all variables in F are assigned $*$ by the first round $\mathcal{R}^-_{1-q}$. (Since in order to remain $*$ after two sweeps, it must remain $*$ after the first). Note that originally the conditions from Lemma 4.4 on F were with $\mathcal{R}^-_{1-q}$ only. For $\mathcal{R}^+_{1-q}$, a given $u \in F$ is assigned $*$ only with probability q. Thus we have the upper bound $q^{\|X_m\|}$.

In the case $\| X_m \| = 0$, we estimate

$$\begin{aligned} & Pr.(D = \emptyset \text{ is all the } *' \text{s in } F \wedge \exists t \in J_{i_\gamma+} - \gamma^N, a_t \neq 0 | A \in \mathcal{A}^{m\gamma}) \\ \leq \; & Pr.(\exists t \in J_{i_\gamma+} - \gamma^N, a_t = 1 | A \in \mathcal{A}^{m\gamma}). \end{aligned}$$

We consider two sweeps from $\mathcal{R}^-_{1-q}$ followed by one from $\mathcal{R}^+_{1-q}$, on $J_{i_\gamma+} - \gamma^N$. The conditions (II), (III) and (IV) are irrelevant now, (using independence). And condition (I) would only reduce the probability for a given $u \in J_{i_\gamma+} - \gamma^N$ to be assigned 1. Unconditionally, a given u is assigned $*$ by $\mathcal{R}^-_{1-q}$ with probability q, hence the upper bound qn^ε, where n^ε comes from the bfi condition $|J_{i_\gamma}| \leq n^\varepsilon$. ∎

Now we can finally estimate $Pr.(\mathcal{A})$. It is bounded above by

$$\sum_{\|S\|>\lceil n^\varepsilon\rceil} Pr.(\mathcal{A}[S]),$$

which is bounded by

$$\sum_{N>\lceil n^\varepsilon\rceil} \sum_{l=1}^{N} \binom{N-1}{l-1} \sum_{l'\geq 0} \binom{l'+l}{l} (2\lceil n^\varepsilon\rceil)^N (qn^\varepsilon)^{l'} q^N,$$

where N stands for norm, $l = \#$ of nonempty X_i in S, and $l' = \#$ of empty X_i in S.

Recall that $q \leq n^{-1.05\varepsilon}$. For a fixed ε, $\exists N_\varepsilon$, such that $\forall n > N_\varepsilon$, $16n^{-0.05\varepsilon} < 1/(2e)$. We get, for $n > N_\varepsilon$,

$$\begin{aligned}
Pr.(\mathcal{A}) &\leq \sum_{N>\lceil n^\varepsilon\rceil} 2^{N-1}(2\lceil n^\varepsilon\rceil)^N q^N \sum_{l=1}^{N} 2^l \sum_{l'\geq 0} 2^{l'}(qn^\varepsilon)^{l'} \\
&\leq 2 \sum_{N>\lceil n^\varepsilon\rceil} 2^N(2\lceil n^\varepsilon\rceil)^N q^N 2^N \\
&\leq 2 \sum_{N>\lceil n^\varepsilon\rceil} (16n^{-0.05\varepsilon})^N \\
&\leq e^{-n^\varepsilon}
\end{aligned}$$

Hence $Pr.(\mathcal{A}) \leq Ce^{-n^\varepsilon}, \forall n$; where C only depends on ε.

Theorem 4.4 is proven.

4.3 Depth k Circuits

Theorem 4.4 is proved under a "skewed" probabilistic assignment. We first "unskew" it:

Theorem 4.5 *Fix $0 < \varepsilon < 1/5$. Then there exists a constant C , such that for any circuit $G \in \Pi_{2,n}$ (or $\Sigma_{2,n}$) with $bfi \leq n^{\varepsilon}$, let $0 \leq p \leq n^{-2.2\varepsilon}$, take $Q \in \mathcal{R}_p$, then $G \|_Q$ is n^{ε}-monochromatic, with probability $1 - \varepsilon_n$ where $\varepsilon_n \leq Ce^{-n^{\varepsilon}}$.*

Proof Clearly we only need to show for the $\Pi_{2,n}$ case.

For $0 \leq p \leq n^{-2.2\varepsilon}$, let

$$p' = 1 - \frac{1}{2}\frac{(1-p)^2}{1-(p(2-p))^{1/2}},$$

$$q = \frac{(p(2-p))^{1/2} - p}{1-p}.$$

It is easy to verify that, $0 \leq p' \approx 1/2 \leq 1$, $0 \leq q = O(p^{1/2}) \leq n^{-1.05\varepsilon}$. Take random $R \in \mathcal{R}^+_{p'}, Q_1 \in \mathcal{R}^-_{1-q}$ and $Q_2 \in \mathcal{R}^+_{1-q}$. It is straightforward to show that RQ_1Q_2 has the same distribution as $Q \in \mathcal{R}_p$.

Now we apply Theorem4.4 to each $G \|_R$, the result follows. ∎

Theorem 4.6 *Let $k \geq 2$, $1 \leq j \leq k-1$. Let $p = n^{-1/k}$, and*

$$0 < \frac{1}{4k} = \varepsilon_k < \varepsilon_{k-1} < \ldots < \varepsilon_1 = \frac{1}{3k}$$

be equally spaced.

For any $G \in \Pi_{j+1,n}$ *(or* $\Sigma_{j+1,n}$ *) with* $bfi \leq n^{1/3k}$ *and* $size(G) \leq e^{n^{1/4k}}$, *take random* $A_1, \ldots, A_j$ *from* $\mathcal{R}_p$, *then with probability* $1 - O(exp(-n^{\varepsilon_j}))$, $G \|_{A_1,\ldots,A_j}$ *is* $n^{1/3k}$*-monochromatic.*

Note: The constant in the O-notation only depends on k.

Proof Fix $k \geq 2$, we prove the theorem by induction on j.

Base case $j = 1$. $G \in \Pi_{2,n}$, with $bfi \leq n^{1/3k}$. Take $\varepsilon = 1/3k < 1/5$, $p = n^{-1/k} \leq n^{-2.2\varepsilon}$ in Theorem 4.5, we have $G \|_{A_1}$ is $n^{1/3k}$-monochromatic with probability $1 - O(exp(-n^{1/3k}))$.

Now suppose $j > 1$, and the Theorem is true for $j - 1$. We prove for $G \in \Sigma_{j+1,n}$ case. The $\Pi_{j+1,n}$ case is dual.

Let $G = \sum_{i=1}^{l} K_i$, where $K_i \in \Pi_{j,n}$. Since G has $bfi \leq n^{1/3k}$, and $size(G) \leq e^{n^{1/4k}}$, $l \leq e^{n^{1/4k}}$, and every K_i inherits the condition on bfi and $size$. Let $B_i = K_i \|_{A_1,\ldots,A_{j-1}}$, then $G \|_{A_1,\ldots,A_{j-1}} = \sum_{i=1}^{l} B_i$.

By induction hypothesis, with probability $1 - O(exp(-n^{\varepsilon_{j-1}}))$, B_i is $n^{1/3k}$-monochromatic, for any i fixed, where the constant is independent of B_i. Hence, with probability $1 - O(exp(-n^{(\varepsilon_{j-1}+\varepsilon_j)/2}))$ all B_i are simultaneously $n^{1/3k}$-monochromatic. Again, the constant here only depends on k.

By Lemma 4.1, all B_i are equivalent to $\Sigma_{2,n}$-formulae, and thus with probability $1 - O(exp(-n^{(\varepsilon_{j-1}+\varepsilon_j)/2}))$, $G \|_{A_1,\ldots,A_{j-1}}$ is equivalent to a Σ_2-

formula with $bfi \leq n^{1/3k}$. Apply Theorem 4.5 once more, we get that $G \|_{A_1,\ldots,A_j}$ is $n^{1/3k}$-monochromatic with probability $1 - O(exp(-n^{\epsilon_j}))$. ∎

Take $j = k - 1$ in Theorem 4.6. we obtain:

Corollary 4.3 *Let $k \geq 2, p = n^{-(k-1)/k}$. For any $G \in \Pi_{k,n}$ (or $\Sigma_{k,n}$) with $bfi \leq n^{1/3k}$ and $size(G) \leq e^{n^{1/4k}}$, take random $R \in \mathcal{R}_p$, then $G \|_R$ is $n^{1/3k}$-monochromatic with probability $1 - o(1)$, uniformly.*

We note that the restriction on bfi is only technical; one may always extend one more level of alternation to have $bfi \leq 1$.

4.4 Circuits vs. Parity

In this section we complete the proof of Theorem 4.3. By the remark we made at the end of last section, we need only to prove:

Theorem 4.7 . *Let $k \geq 2$. There exists a sequence $\{\alpha_n\}$, $\alpha_n \rightarrow 1/2$, such that, all depth k Boolean circuits, with n input, size $\leq exp(n^{1/4k})$, and $bfi \leq n^{1/3k}$, when computing $Parity_n$, make error on $\geq \alpha_n$ of all 2^n possible input.*

The strategy to prove Theorem 4.7 is the following: Fix $k \geq 2$. Consider any depth k circuit G satisfying the conditions. Take randomly a total assignment σ (all 2^n many assignments from $\{0,1\}^n$ are equally likely). We wish to prove that $G|_\sigma \neq \boldsymbol{Parity_n}|_\sigma$, with probability $1/2 - o(1)$. (Where o(1) may depend on k, but it is independent of G.)

Now we choose to pick σ in two stages: First randomly pick a "single $*$" σ^*, so that all $\sigma^* \in A^* \equiv \{\sigma \in \{0,1,*\}^n | \exists \text{ a unique } d, \sigma_d = *\}$ are equally likely. Then assign the unique $*$ in σ^* to 0 or 1 with equal probability, to obtain our random σ.

Theorem 4.7 will be proved if we can show that $G|_{\sigma^*} \equiv$ constant, with probability $1 - o(1)$, since for any σ^*, the conditional probability for failure is

$$\boldsymbol{Pr.}(G|_\sigma \neq \boldsymbol{Parity_n}|_\sigma \mid G|_{\sigma^*} \equiv \text{ constant}) = 50\%.$$

Now our strategy to generate a random $\sigma^* \in A^*$ is the following: Let $p = n^{-(k-1)/k}$ and $q = n^{-1/2.5k}$. For a nonempty finite set of variables S, an "A^*-uniform assignment" on S is a random assignment that randomly picks one variable in S as $*$ and unformly assigns the others to 0 or 1.

Procedure Generate 1 (σ^*)

Take a random $A \in \mathcal{R}_p$

if (A leaves $\leq n^{1/2k}$ variables in X unassigned)

then take a random σ^*

else take a random $B \in \mathcal{R}_q$

if (AB assigned every variable in X)

then take a random σ^*

else let $S = \{x_i \in X | AB$ assigns x_i to $*\}$,

A^*-uniformly assign S,

let σ^* be the result.

fi

fi Return (σ^*)

Clearly Generate 1 does generate every $\sigma^* \in A^*$ equally likely.

Now we "realize" Generate 1 by the following procedure, which will complete our proof:

Procedure Generate 2 (σ^*, tag)

Take a random $A \in \mathcal{R}_p$

if (A leaves $\leq n^{1/2k}$ variables in X unassigned)

then $tag :=$ failure, take a random σ^*

else **if** ($G \|_A$ is not $n^{1/3k}$-monochromatic)

then $tag :=$ failure, take a random $B \in \mathcal{R}_q$

if (AB assigned every variable in X)

then take a random σ^*

else let $S = \{x_i \in X | AB$ assigns x_i to $*\}$,

A^*-uniformly assign S,

let σ^* be the result.

fi

else play the game (as the player),

assign any given variable with distribution $\mathcal{R}_q$

if (the master ever gets a $*$)

then stop the game, $tag :=$ failure,

run through $\mathcal{R}_q$ for the remaining variables,

let $S = \{x_i \in X | x_i$ is unassigned so far$\}$,

A^*-uniformly assign S,

Let σ^* be the result.

else when the game is finished, run through $\mathcal{R}_q$ for the rest,

if (no variable is assinged $*$)

then $tag :=$ failure, take a random σ^*

else $tag :=$ success,

let $S = \{x_i \in X | AB$ assigns x_i to $*\}$,

A^*-uniformly assign S,

Let σ^* be the result.

fi

fi

fi

fi Return (σ^*, tag)

Clearly if we ignore the tag, Generate 2 is the same as Generate 1. If Generate 2 returns (σ^*, success), then $G|_{\sigma^*} \equiv$ constant. Let F denote the event that Generate 2 returns with $tag =$ failure. We claim:

$$Pr.(F) = o(1).$$

We only need to verify:

1. $Pr.(A \in \mathcal{R}_p$ leaves $\leq n^{1/2k}$ variables in X unassigned $)= o(1)$.

   This follows from Chebechev's Inequality.

2. $Pr.(G \;||_A$ is not $n^{1/3k}$-monochromatic $) = o(1)$.

   This is Corollary 4.3.

3. $Pr.$ (the master ever gets a $*$ under $\mathcal{R}_q \;|G \;||_A$ is $n^{1/3k}$-monochromatic).

   $= o(1)$.

   This is because $n^{1/3k} \cdot n^{-1/2.5k} \rightarrow 0$.

4. $Pr.$ (AB leaves no $*$ in $X \;|A$ leaves $\geq n^{1/2k}$ variables unassigned in X)$= o(1)$.

This is trivial.

4.5 Final Remarks

The result concerning circuit and parity is of independent interests other than relativization. After all one cannot do any worse than 50% error for parity.

The following Corollary is evident:

Corollary 4.4 *(Yao) There is a recursive oracle A separating $PSPACE$ from the Polynomial-Time Hierarchy.*

The proof is simple. Observe that with probability one the Parity Language $\boldsymbol{Parity^A}$ is not in PH^A. Hence for those A, $L^A(M_i)$ differs from $\boldsymbol{Parity^A}$ infinitely often for any PH machine M_i. By the definition of measure μ, any initial segment of A corresponds to a small interval of [0,1]. Now suppose we are given an initial segment of A, the oracle constructed so far, and we want to diagonalize over M_i. What we do is simply look for an extension that kicks M_i out. The "brute force" method must succeed due to our probability one separation.

Shortly after this work, Hastad [Has86] has obtained a simplification of Yao's proof, improving the bound on the circuit size from $\Omega(e^{n^{1/4k}})$ to $\Omega(e^{cn^{1/k}})$. Later Babai [Bab86] has obtained the result in Theorem 4.1 by a short proof assuming Yao's theorem and a result by Ajtai [Ajt83].

Chapter 5

Conclusions

In this thesis we considered probability one separation of complexity classes, the Boolean Hierarchy BH and the Polynomial-time Hierarchy PH from $PSPACE$. The following extension is natural and still open:

- Is it true that with probability one, a random oracle separates the Polynomial-time Hierarchy PH into an infinite hierarchy?

In [CH86], we achieved various collapsing and separation results on the Boolean Hierarchy BH. It is natural to ask the same questions on the Polynomial-time Hierarchy PH. e.g.

- Is there an oracle that separates the first k levels of the Polynomial-time Hierarchy PH and yet collapsing PH to Σ_k^P, with or without collapsing $PSPACE$ to PH?

Relativized results are interesting, for they open up worlds of possibilities. More interesting than that, of course, is the world of unrelativized classes, $\boldsymbol{P}$, $\boldsymbol{NP}$, $\boldsymbol{PH}$, $\boldsymbol{PSPACE}$ etc. It is hoped that our techniques in considering circuit computation with the game theoretic and probabilistic approach, could shed some light on future research in the unrelativized world. It is well known that if $\boldsymbol{P} = \boldsymbol{NP}$ then $\boldsymbol{NP}$ has polynomial size circuit (not of bounded depth, in general). Thus proving some $\boldsymbol{NP}$ function not computable by any polynomial size circuit will settle $\boldsymbol{P} \neq \boldsymbol{NP}$. Recently Razborov [Raz85] has shown that certain $\boldsymbol{NP}$ function can not be computed by any polynomial size *monotone* circuit. (A circuit is *monotone*, if the *not* ($\neg$) gate does not appear.) On the other hand, the existence of polynomial size circuit for $\boldsymbol{NP}$ is equivalent to the existence of a sparse oracle for $\boldsymbol{NP}$. Mahaney [Mah82] showed that the existence of a sparse $\boldsymbol{NP}$-complete set (in $\boldsymbol{NP}$) implies $\boldsymbol{P} = \boldsymbol{NP}$. Karp and Lipton [KL80] showed that the existence of a sparse oracle for $\boldsymbol{NP}$ implies $\boldsymbol{PH} = \Sigma_2^{\boldsymbol{P}}$. Mahaney later improved this result to $\boldsymbol{PH} = \Delta_2^{\boldsymbol{P}} (= \boldsymbol{P}^{\boldsymbol{SAT}})$ [Mah82]. Recently Kadin further improved it to $\boldsymbol{PH} = \boldsymbol{P}^{\boldsymbol{SAT}[log]}$, $\boldsymbol{P}$ with logarithmic many queries to a $\boldsymbol{SAT}$ oracle [Kad86]. Furthermore, it is optimal among all proofs that relativizes.

In short, if a sparse oracle for $\boldsymbol{NP}$ exists, then the Polynomial-time Hierarchy collapses to the second level.

Thus, one way or the other, circuit computation is closely tied to the central questions in this field.

We believe that circuit computation will continue to play an interesting role in the world of feasible computation.

Bibliography

[AH77a] K. Appel and W. Haken. Every planar map is 4-colorable – 1: discharging. *Ill. J. Math*, 21:429–490, 1977.

[AH77b] K. Appel and W. Haken. Every planar map is 4-colorable – 2: reducibility. *Ill. J. Math*, 21:491–567, 1977.

[Ajt83] M. Ajtai. Σ_1^1 -formulae on finite structures. *Ann. Pure and Applied Logic*, 24:1–48, 1983.

[Amb86] Klaus Ambos-Spies. Randomness, relativization, and polynomial reducibilities. In *Structure in Complexity Theory*, pages 23–34, Springer-Verlag *Lecture Notes in Computer Science #223*, 1986.

[Bab86] L. Babai. A random oracle separates $PSPACE$ from Polynomial Hierarchy. *Notes*, 1986. To appear.

[BG81] C. Bennet and J. Gill. Relative to a random oracle A, $P^A \neq NP^A$ with probability 1. *SIAM J. on Computing*, 10:96–113, 1981.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativization of $P = ?NP$ question. *SIAM J. on Computing*, 4:431–442, 1975.

[BS79] T. Baker and A. Selman. A second step toward the Polynomial hierarchy. *Theoretical Computer Science*, 8:177–187, 1979.

[Cai86] J. Cai. With probability one, a random oracle separates $PSPACE$ from the Polynomial-time hierarchy. In *ACM Sym. on Theory of Computation*, pages 21–29, 1986.

[CH86] J. Cai and Lane A. Hemachandra. The Boolean hierarchy: hardware over NP. In *Structure in Complexity Theory*, pages 105–124, Springer-Verlag *Lecture Notes in Computer Science #223*, 1986.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of ACM*, 26(1), 1981.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In *Proc. 1964 International Congress for Logic Methodology and Philosophy of Science*, pages 24–30, North Holland, 1964.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *ACM Sym. on Theory of Computation*, pages 151–158, 1971.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the Polynomial-time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[Har85] J. Hartmanis. Solvable problems with conflicting relativizations. *Bulletin of the European Association for Theoretical Computer Science*, 27:40–49, Oct 1985.

[Has86] J. Hastad. Almost optimal lower bounds for small depth circuits. In *ACM Sym. on Theory of Computation*, pages 6–20, 1986.

[HS65] J. Hartmanis and R. Sterns. On the computational complexity of algorithms. *Trans. AMS*, 117:285–306, 1965.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[Kad86] Jim Kadin. *Deterministic Polynomial Time with $O(log(n))$ Queries.* Technical Report TR-86-771, Cornell University, August 1986.

[Kar72] R. Karp. Reducibilities among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *ACM Sym. on Theory of Computation*, pages 302–309, 1980.

[Kur82] S. A. Kurtz. On the random oracle hypothesis. In *ACM Sym. on Theory of Computation*, pages 224–230, 1982.

[Mah82] S. Mahaney. Sparse complete sets for $\boldsymbol{NP}$: solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[Mat70] Y. V. Matijasevic. Enumerable sets are diophantine. *Soviet Math. Dokl.*, 11:354–357, 1970.

[Raz85] A. A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Dokl. Ak. Nauk.*, 281:798–801, 1985.

[Sip83] M. Sipser. Borel sets and circuit complexity. In *ACM Sym. on Theory of Computation*, pages 61–69, 1983.

[Sto77] L. Stockmeyer. The Polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Wag85] K. Wagner. On the Boolean closure of ***NP***. In *Proc. of the 1985 International Conference on Fundamentals of Computation Theory*, pages 485–493, Springer-Verlag *Lecture Notes in Computer Science* , 1985.

[Yao85] A. Yao. Separating the Polynomial-time hierarchy by oracles. In *IEEE Annual Symposium on Foundations of Computer Sciece*, pages 1–10, 1985.

[Yes83] Y. Yesha. On certain polynomial-time truth-table reducibilities of complete sets to sparse sets. *SIAM Journal on Computing*, 12(3):411–425, 1983.