DO WE KNOW IT ALL?

Whenever our ignorance is revealed,
We say now the vacuum is filled,
And then we claim, we know,
All that is there to know.
What new soul there can be?
Now that so much has been.
Everything and everyone must be,
A product of all that has been.

And yet, here is a simple fact,
We forget, we already know.
Rare they may get,
The further we go.
But there will always be,
Some odd ones,
Which are not products of,
What has come before.

They will always be,
Some less, or some more,
Than the products of,
What has been before.
And here is,
The most amazing part,
There will never be an easy way,
To definitely tell them apart.

Sanjay Gupta

Determinism, Nondeterminism, Alternation, and Counting

By

Sanjay Gupta, Ph.D.

The Ohio State University, 1994

Prof. T.J. Long, Adviser

The goal of complexity theory is to determine the amount of computational resources needed to solve various problems. Structural complexity theory attempts to determine the computational resources needed to solve problems in various complexity classes and to determine the relationships among complexity classes. This approach to complexity theory is less adhoc than consideration of individual problems in isolation.

In this dissertation we have considered three sets of results, all related to increasing our understanding of the structure of various complexity classes and relationships among them. The three sets of results that are explored are:

1. Separations of time complexity classes.

2. Closure properties of several function classes.

3. Tools for embedding one complexity class in another.

Determinism, Nondeterminism, Alternation, and Counting

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Sanjay Gupta, B.Tech., M.S.

* * * * *

The Ohio State University

1994

Dissertation Committee:

Prof. T.J. Long

Prof. K.J. Supowit

Prof. N. Soundrarajan

Approved by

Adviser
Department of Computer
and Information Science

This thesis is dedicated to the aristocracy.

I believe in aristocracy, though — if that is the right word, and if a democrat may use it. Not an aristocracy of power, based upon rank and influence, but an aristocracy of the sensitive, the considerate and the plucky. Its members are to be found in all nations and classes, and all through the ages, and there is a secret understanding between them when they meet. They represent the true human tradition, the one permanent victory of our queer race over cruelty and chaos. Thousands of them perish in obscurity, a few are great names. They are sensitive for others as well as for themselves, they are considerate without being fussy, their pluck is not swankiness but the power to endure, and they can take a joke.

...... "What I believe" in E.M. Forster's *Two Cheers for Democracy*

ACKNOWLEDGEMENTS

A twenty one year old "boy" came to the U.S.A. from India in Feb, 1987, with deep beliefs, and yet unsure of his values; bright, and yet did not know what he could do; with great strength, and yet was sickly and constantly dependent on doctors. There was constant conflict between what he wanted to believe and do, and what he saw around him. He chose solitude to find from within what is right. Today, he runs marathons, and has not seen a doctor for seven years, does not depend on others for his values, and has submitted this dissertation in partial fulfillment of the requirements for the degree Doctor of Philosophy.

I am extremely grateful for the nurturing environment, the academic and social freedom of the U.S.A. that enabled me to discover myself. While my struggles in India indeed shaped me considerably, I can honestly say that if I had not come to the U.S.A., I would not be where I am today. (In one sense this statement is strictly meaningless.)

Numerous persons helped me in my endeavour, not just by helping me, but by being themselves. The foremost person is my advisor Tim Long whose academic integrity is inspirational. It is from Tim I am learning to be my own self and at the same time to be "respectful, thoughtful, and considerate" for others. He has always been available for whatever advice I have needed. His teaching style, his ability to keep the right perspective on research, and his social skills, all have influenced me considerably. I could not have hoped for a better advisor.

I am grateful to Neelam Soundrarajan for his personal and academic advice on numerous occasions that made my life easier. Only by knowing Neelam, one can find out how to be honest and nice at the same time. I am also grateful to Ken Supowit for several useful discussions I had with him, and for making his "magnificient library" available to me. His straightforwardness and sense of humor are hard to emulate. I also had several useful discussions with my colleagues J. Ramachandran and Ming-jye

Sheu.

It is hard to believe that Amit is just my friend. Amit and Anita provided me with a true family, at times exceeding what I would expect from my family. I also enjoyed the hospitality of Srinivas and Subha on several occasions.

It would have been very hard to survive Columbus without the hospitality of Paolo Bucci. If anything can make me run better, it is Paolo's pasta.

I had the constant support of the office staff of the department of computer science. I must mention the two superstars, Elley Quinlan and Tom Fletcher, without whose help life would have been indeed very hard.

Finally, I would like to thank my parents, who have been most understanding and supportive especially in my long absence from home.

VITA

September 9, 1965	Born - India
Aug, 82 - Jun, 86	B.Tech., Department of Computer Science and Engineering, Indian Institute of Technology, Delhi
Aug, 86 - Jan, 87	Programmer Analyst, Tata Unisys Limited, New Delhi, India
Feb, 87 - Dec, 87	Consultant, Financial Systems Center, Unisys Corporation, Atlanta, Georgia
Jan, 88 - Dec, 88	M.S., Department of Computer and Information Science, The Ohio State University
Jan, 92 - May, 92	Graduate Research Associate, Department of Computer Science, New Mexico State University
Jan, 88 - present	Graduate Teaching Associate, Department of Computer and Information Science, The Ohio State University

Publications

Journals

1. S. Gupta, The power of witness reduction, Technical Research Report, OSU-CISRC-2/91-TR5. To appear in *Journal of Computer and System Sciences.*

2. S. Gupta, On the closure of certain function classes under integer division by polynomially-bounded functions, *Information Processing Letters* 44(1992), pages 205-210.

3. S. Gupta, On bounded-probability operators and $C_{=}P$, Technical Research Report, OSU-CISRC-11/92-TR32. To appear in *Information Processing Letters.*

4. S. Gupta, On isolating an odd number of elements and its applications in complexity theory, Technical Research Report, OSU-CISRC-6/93-TR24. Under revision for *Mathematical Systems Theory.*

Conferences

5. S. Gupta, The power of witness reduction, Proceedings of 6th Annual Structure in Complexity Theory Conference, pages 43-59, IEEE Computer Society Press, 1991.

6. S. Gupta, Alternating time versus deterministic time: a separation, Proceedings of the 8th Structure in Complexity Theory Conference, pages 266-277, IEEE Computer Society Press, 1993 (part of the Federated Computing Research Conference).

Thesis

7. S. Gupta, Fuzzy Memory, Undergraduate thesis, 1986.

Fields of Study

Major Field: Computer and Information Science

Studies in:

Theoretical Computer Science	Prof. T.J. Long
Design and Analysis of Algorithms	Prof. K.J. Supowit
Programming Languages	Prof. N. Soundrarajan

Table of Contents

List of Figures

CHAPTER I

Introduction

The study of the application of science is easy for anyone who is master of the theory of it.

— Louis Pasteur

Consider the following well-known problems, which are very similar in appearance.

EULER TOUR PROBLEM: Given a graph G, as in Figure 1, is there a tour in G, starting from any fixed vertex, that traverses all the *edges* in G exactly once and returns to the starting vertex?

HAMILTONIAN CIRCUIT PROBLEM: Given a graph G, as in Figure 1, is there a tour in G that traverses all *vertices* exactly once and returns to the point of origin?

While the Euler Tour problem can be solved "efficiently" on most models of computation, we do not know of any way of "efficiently" solving the Hamiltonian Circuit problem for arbitrary graphs. Here "efficiently" means "within reasonable time", but in general, "efficiently" can be used to refer to a variety of computational resources such as time, space, amount of randomness, etc.

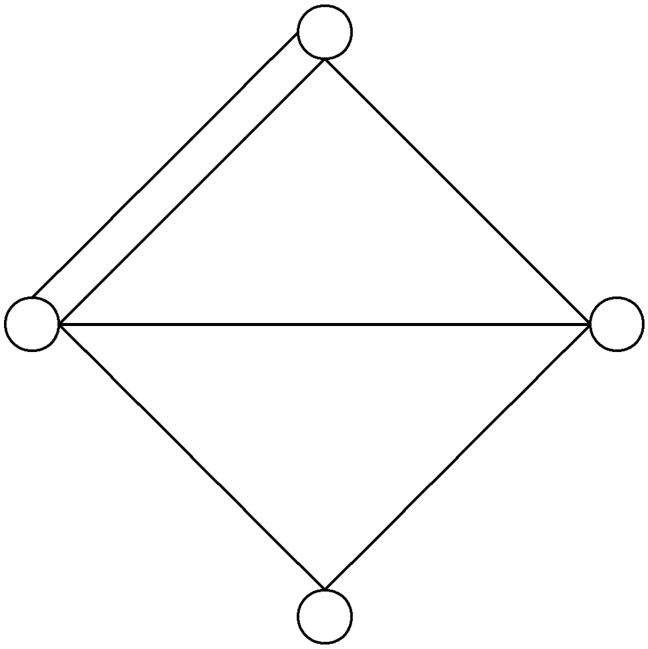

Figure 1: Graph G

The goal of complexity theory is to determine the amount of computational resources needed to solve various problems. Clearly, this depends on the computational model as well as the type of resource. In this thesis, the model of computation used is multi-tape Turing machines and the resource we are concerned with is time. Actually, we will use three types of Turing machines: deterministic, nondeterministic, and alternating. Nondeterministic machines are "guessing machines" — at each step they are allowed to guess the next move. Alternating Turing machines are a generalization of nondeterministic Turing machines and will be described later.

The Euler Tour problem and the Hamiltonian Circuit problem can be rephrased as language recognition problems. Define the sets ET and HC as follows:

$$\text{ET} = \{G \mid \text{Graph } G \text{ has an Euler tour }\}$$

$$\text{HC} = \{G \mid \text{Graph } G \text{ has a Hamiltonian circuit }\}$$

A graph G belongs to set ET if and only if G has an Euler tour. Similarly, a graph G belongs to set HC if and only if G has a Hamiltonian circuit. Let n denote the size of a reasonable encoding of graph G. The complexity of a set is usually measured as a function of the size of input. For example, it is easy to devise an algorithm for ET that in the worst case takes no more than $O(n^2)$ steps on a deterministic Turing machine. Thus, the language ET can be recognized within a polynomial number of steps in the size of the input. The class of such languages, that can be recognized in time polynomial in the length of the input, is denoted by P. On the other hand, we do not know of any algorithm to recognize, in the worst case, set HC that takes less than an exponential number of steps in the size of the input. Thus, we do not know whether HC belongs to class P or not.

Given a sequence of vertices, it is easy to *verify* whether it is a Hamiltonian circuit or not in a polynomial number of steps in the length of the input. Thus, a nondeterministic machine can *guess* a sequence of vertices that forms a Hamiltonian circuit and then verify in polynomial time the correctness of the guess. The class of problems for which a solution can be *verified* in polynomial time in the length of the input is denoted NP.

Clearly the class NP includes all the problems in P. Furthermore, there is a rich subclass of problems in NP, known as NP-complete problems, that contains seemingly different problems from very diverse areas — graph theory, network design, mathematical programming, algebra and number theory, games and puzzles, logic, automata and language theory, program optimization, and operations research — to name a few. These problems are beautifully interrelated — if one of them can be solved efficiently, then all of them can be.

Thus, NP-complete problems are very strongly related to each other. In general, a complexity class is a set of related problems, or equivalently a set of related languages. Because of the discovery of the class NP-complete, the notion of complexity classes has gained wide acceptance and a plethora of classes have been defined. Structural complexity theory attempts to determine the computational resources needed to solve problems in various complexity classes and to determine the relationships among complexity classes. This approach to complexity theory is less adhoc than consideration of individual problems in isolation. From a practical view point, but for our understanding of these concepts, researchers in different areas would still be wasting hours of their time, trying to obtain efficient algorithms for individual problems that have now been catalogued in "hard" complexity classes.

In this thesis we are concerned with increasing our understanding of various complexity classes and the relationships between them. The three sets of results that are explored are:

1. Separations of time complexity classes.
2. Closure properties of several function classes.
3. Isolating techniques used as a tool when embedding one complexity class in another.

In the remainder of this section, we discuss our results, organized by chapters. Appendix B summarizes the notation and definitions used in this thesis.

Chapter II — Separation results

As mentioned before, our model of computation is the multi-tape Turing machine. DTIME($t(n)$) denotes the class of languages that can be recognized by deterministic Turing machines in time $t(n)$. NTIME($t(n)$) denotes the class of languages that can be recognized by nondeterministic Turing machines in time $t(n)$. It is easy to see that DTIME($t(n)$) $\subseteq$ NTIME($t(n)$), since deterministic Turing machines are a special type of nondeterministic Turing machines.

Nondeterministic Turing machines are *existential* machines — at each time step they are allowed to choose among a finite number of possible moves. Thus, if the machine makes $t(n)$ steps, it can have an exponential number (in $t(n)$) of possibly different computations. If there *exists* any computation that accepts, so does the machine.

Similarly, we can define *universal* nondeterministic machines that accept if *all* the possible computations accept. We can go a step further and define a machine which alternates between universal and existential states. For example, a machine with two alternations, starting in an existential state, accepts if there exists a sequence of choices made in the existential state such that the machine accepts on all the following choices made in the universal state. These machines are called alternating machines [10], a natural generalization of nondeterministic machines.

$\Sigma_l(t(n))$ denotes the languages accepted in total time $t(n)$ by alternating machines that start in an existential state and make at most $l-1$ alternations from existential to universal phase, and vice versa. Clearly, NTIME$(t(n)) \subseteq \Sigma_l(t(n))$, for all $l \geq 1$, as nondeterministic machines are the same as Σ_1 machines.

The class P is the union of all languages that can be recognized in time n^k, for some $k \geq 1$, using deterministic Turing machines; that is, P$=\cup_{k\geq 1}$ DTIME(n^k). Similarly, NP$=\cup_{k\geq 1}$ NTIME(n^k). Clearly, P $\subseteq$ NP. It is widely believed that P $\neq$ NP.

It is easy to show that P $\neq$ NP if and only if NTIME$(n) \not\subseteq$ DTIME(n^k), for all $k \geq 1$; that is, P is different from NP if and only if languages recognized by nondeterministic Turing machines in linear time cannot be recognized by any polynomial time bounded deterministic Turing machine. Unfortunately, so far, we only know that NTIME$(n) \not\subseteq$ DTIME(n) [31]; that is, linear nondeterministic time is strictly more powerful than linear deterministic time.

Following the linear separation result[31], two very interesting problems, seemingly easier than the P $\stackrel{?}{=}$ NP question, have remained open. Is DTIME$(n^k) \neq$ NTIME(n^k), for each $k \geq 2$? Is NTIME$(n) \neq$ co-NTIME(n)? To understand the notation co-NTIME(n), given a class $\mathcal{C}$, co-$\mathcal{C}$ denotes the class of languages which are complements of languages in $\mathcal{C}$; that is $\overline{L} \in$ co-$\mathcal{C}$ if and only if $L \in \mathcal{C}$. Clearly, co-DTIME(n)=DTIME(n), but we do not know if the same is true for NTIME(n).

In Chapter II, we provide a partial answer to these fundamental questions. Our main theorem states: For each "reasonable" function $t(n)$ (which includes polynomials) DTIME$(t(n)) \neq \Sigma_2(t(n))$.

Thus, there are languages accepted by an immediate generalization of nondeterministic machines in time $t(n)$ that are not accepted by any deterministic machine in the same amount of time. An important corollary of our result is: At least one of the following two statements has to be true:

1. $\mathrm{NTIME}(n) \neq \text{co-}\mathrm{NTIME}(n)$.

2. $\mathrm{DTIME}(t(n)) \neq \mathrm{NTIME}(t(n))$, for each "reasonable" $t(n)$.

Chapters III and IV — Closure Properties and Witness Reduction

Another approach to help us solve the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question is to understand the structure of problems in NP. This relates to our second set of results. Consider any problem in NP, for example, the CLIQUE problem defined as follows:

$$\mathrm{CLIQUE} = \{(G,k) \mid \text{Graph } G \text{ has a clique of size } k\ \}.$$

Thus, the CLIQUE problem is: Given a graph G and an integer k, does G have a clique of size k? Let $\#\mathrm{Cliques}(G,k)$ denote the number of cliques of size k in graph G. Given two graphs G_1 and G_2, it is easy to produce a graph G such that $\#\mathrm{Cliques}(G,k) = \#\mathrm{Cliques}(G_1,k) + \#\mathrm{Cliques}(G_2,k)$. It is also easy to produce a graph G such that, $\#\mathrm{Cliques}(G,k) = \#\mathrm{Cliques}(G_1,k) * \#\mathrm{Cliques}(G_2,k)$.

In general, it is easy to add and multiply the number of solutions of instances of problems in NP. A natural question is: Is the same true for subtraction and division? This question was investigated by Ogiwara and Hemachandra[29].

To make these questions more precise, let $\#\mathrm{P}$ denote the class of functions f such that there exists a polynomial time bounded nondeterministic Turing machine M such that $f(x)$ = number of accepting paths of the machine M on input x. It is easy to prove that the class #P is closed under addition; that is, given f_1 and f_2 in #P, there exists a $g \in$ #P such that for all x, $g(x) = f_1(x) + f_2(x)$. Also, #P is closed under multiplication. Note that the range of all #P functions is the set of natural numbers. Hence, we cannot talk about closure of #P under general subtraction or division. What Ogiwara and Hemachandra considered was proper subtraction ($a \ominus b = \max\{0, a-b\}$) and integer division ($a \div b = \lfloor \frac{a}{b} \rfloor$).

Ogiwara and Hemachandra[29] proved that #P is closed under proper subtraction iff #P is closed under integer division iff #P is closed under *every* other function in #P. Thus, closure under proper subtraction or integer division implies closure under every other #P function. Such functions, closure under which implies closure under every other function, are called "hard functions" by Ogiwara and Hemachandra.

Ogiwara and Hemachandra also proved that #P is closed under proper subtraction implies that the polynomial hierarchy collapses to a very weak class lying between P and NP. The polynomial hierarchy (PH) [19, 26] denotes the class of languages accepted by alternating Turing machines, which make a finite number of alternations, in time n^k, for some $k \geq 1$; that is, PH $= \cup_{k,l\geq 1}\Sigma_l(n^k)$. Equivalently, PH can be viewed as a finite number of polynomially bounded universal and existential quantifiers over a polynomial time predicate. Thus, closure of #P under proper subtraction or integer division has very drastic consequences.

We address the following question: Why should closure under proper subtraction or integer division imply such drastic collapses? Is there something special about subtraction or division which is not true for addition and multiplication?

To explore these questions, in Chapter IV we define two natural extensions of #P: $\mathcal{Z}\#$P and $\mathcal{Q}\#$P. $\mathcal{Z}\#$P is the closure of #P under subtraction, and $\mathcal{Q}\#$P is the closure of $\mathcal{Z}\#$P under division. Thus, $f \in \mathcal{Z}\#$P iff there exists $g, h \in \#$P such that for all x, $f(x) = g(x) - h(x)$. Obviously, $\mathcal{Z}\#$P is closed under subtraction. However, consider the question of whether $\mathcal{Z}\#$P is also closed under proper subtraction. In Chapter IV, we prove that $\mathcal{Z}\#$P is closed under *proper* subtraction iff $\mathcal{Z}\#$P is closed under *integer* division iff $\mathcal{Z}\#$P is closed under *every* other function in $\mathcal{Z}\#$P. Thus, we show that proper subtraction and integer division remain hard for $\mathcal{Z}\#$P. We also prove that $\mathcal{Z}\#$P is closed under *proper* subtraction iff the counting hierarchy collapses to a weak counting class. The counting hierarchy(CH), first defined in [49], is used to classify problems that are seemingly harder than the problems in polynomial hierarchy. It is formally defined in Appendix B.

In the case of $\mathcal{Q}\#$P, we prove that $\mathcal{Q}\#$P is closed under *proper* subtraction iff $\mathcal{Q}\#$P is closed under *integer* division iff $\mathcal{Q}\#$P is closed under *every* other function in $\mathcal{Q}\#$P iff there are some "local" collapses in CH. Thus, we show that proper subtraction and integer division remain hard for $\mathcal{Q}\#$P as well.

We study several similar results for several different function classes. Based on these results, we have identified *witness reduction* as the cause for collapses due to closure under integer division and proper subtraction. Our thesis is that any function that performs some form of *witness reduction* also effects some form of collapse. Furthermore, we believe that for a function to be "hard" for some function class, it must effect some form of *witness reduction*.

In Chapter III, we extend the work of Ogiwara and Hemachandra in another direction. Since closure of #P under proper subtraction has such drastic consequences, #P therefore is unlikely to be closed under proper subtraction. Thus, it is natural to ask the question "Is #P closed under subtraction with a high probability?" In other words, given $f_1, f_2 \in \#\mathrm{P}$, does there exists a $h \in \#\mathrm{P}$ which "approximates" $f_1(x) \ominus f_2(x)$, for all x, with a high probability. We formalize this notion of "approximation" and obtain results very analogous to those of Ogiwara and Hemachandra.

Chapter V — Isolating an Odd Number of Elements

An important application of witness reduction is Toda's result[44], which states $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{CP}}$. PH is the polynomial time hierarchy mentioned earlier. CP is a counting class — a language is in CP iff it is accepted by a nondeterministic machine on a *majority* of computation paths. Thus, Toda's result states that any language in the polynomial time hierarchy can be accepted by a deterministic Turing machine that has access to a counting machine as a subroutine. Thus, access to one counting quantifier is more powerful than any finite number of existential and universal quantifiers over a deterministic polynomial time predicate.

Actually, Toda's result splits into two parts: $\mathrm{PH} \subseteq \mathrm{BP} \cdot \oplus\mathrm{P}$ and $\mathrm{BP} \cdot \oplus\mathrm{P} \subseteq \mathrm{P}^{\mathrm{CP}}$. Language L is in the class $\oplus\mathrm{P}$ if there is a nondeterministic machine M that accepts input x in L on an odd number of computations; that is, M has an odd number of accepting computations iff $x \in L$. Given a class $\mathcal{C}$, a language L is in the class $\mathrm{BP} \cdot \mathcal{C}$ iff there exists a set $A \in \mathcal{C}$, and a polynomial $p(n)$ such that $\mathrm{Prob}(x \in L \text{ iff } \langle x, y\rangle \in A) \geq \frac{3}{4}$, where y is chosen uniformly from $\{0,1\}^{p(|x|)}$. Thus, Toda showed that languages in polynomial time hierarchy can be accepted by nondeterministic Turing machines by an odd number of computations, with a high probability.

In proving PH $\subseteq$ BP$\cdot\oplus$P, any finite number of existential and universal quantifiers are collapsed into $\oplus$P using the witness reduction provided in a theorem due to Valiant and Vazirani [48]. They constructed a random reduction from a non-empty set to a set with cardinality exactly 1; that is, given a set S of n-bit vectors, the reduction constructs $n+1$ subsets such that if $S \neq \emptyset$, then one of the $n+1$ subsets will have exactly one element with probability greater than $\frac{1}{4}$. Of course, if $S = \emptyset$, then all the $n+1$ subsets remain empty. Hence, if we pick one of the $n+1$ subsets uniformly, we have a random reduction from a non-empty set to a set with cardinality 1 with probability at least $\frac{1}{4(n+1)}$.

This result has been extensively applied in complexity theory to show containment of the polynomial-time hierarchy in various counting classes[44, 46, 43], and to show that languages accepted by various constant depth circuits can also be accepted by depth-2 threshold circuits of superpolynomial size[1, 43, 51, 6]. In several of these complexity theory applications, it is sufficient to have a subset with an odd number of elements, not necessarily exactly one. In Chapter V we construct a random reduction from an arbitrary non-empty set to a set with an odd number of elements with a probability of success at least $\frac{3}{16}$. Thus, the probability of success, if we are only interested in isolating an odd number of elements, is independent of n. More precisely, given any set S of n-bit vectors, we can construct *one* subset such that if $S \neq \emptyset$, then the constructed subset will have an odd number of elements with probability greater than $\frac{3}{16}$.

Chapter VI — Complexity Theory Applications of Isolating an Odd Number of Elements

Finally, in Chapter VI, we improve bounds on several results in complexity theory by using our construction, given in Chapter V, in place of Valiant and Vazirani's construction[48]. In particular, we obtain better bounds on polynomials which approximate boolean functions, improve bounds on the running time of the $\oplus$P machines in Toda's result, and improve bounds on the size of threshold circuits accepting languages accepted by polynomial-sized constant-depth circuits with only AND/OR gates (AC^0 circuits).

CHAPTER II

Separations

For every complicated problem there is an answer that is short, simple and wrong.

— H.L. Mencken

2.1 Introduction

Separating complexity classes is a central goal of complexity theory. For example, is deterministic space strictly more powerful than deterministic time? Is nondeterminism strictly more powerful than determinism? These are some of the fundamental questions and, in spite of extensive work in the last few decades, only a few such separations have been obtained. For the Turing machine model of computation, aside from the standard time and space hierarchies[16, 41, 23], the important separations proven by Hopcroft, Paul and Valiant [17], Paul, Pippenger, Szemerédi, and Trotter [31], and Dymond and Tompa [11] are:

1. [17] For each $t(n) \geq n$, $\mathrm{DTIME}(t(n)) \subset \mathrm{DSPACE}(t(n))$.

2. [31] $\mathrm{DTIME}(n) \subset \mathrm{NTIME}(n)$.

3. [11] For each $t(n) \geq n$, DTIME$(t(n)) \subset$ ATIME$(t(n))$.

(The symbol $\subset$ denotes strict containment). Thus, the first result says that the set of languages accepted in space $t(n)$ strictly contains the set of languages accepted in deterministic time $t(n)$. The second result says that the set of languages accepted in nondeterministic linear time strictly contains the set of languages accepted in deterministic linear time. The third result says that the set of languages accepted by alternating machines in time $t(n)$ strictly contains the set of languages accepted in deterministic time $t(n)$. This result improves the separation given by the first result as ATIME$(t(n)) \subseteq$ DSPACE$(t(n))$, for all $t(n) \geq n$.

Although these results confirm our intuition that both space and nondeterminism are more powerful than deterministic time, they are still rather weak in their consequences. For example, it is not known if the result by Paul, Pippenger, Szemerédi, and Trotter [31] extends to arbitrary polynomials: Is DTIME$(n^k) \neq$ NTIME(n^k), for some $k > 1$? Another natural extension of the work of [31] also remains open: Is NTIME$(n) \neq$ co-NTIME(n)?

In this chapter our goal is to sharpen the separations between nondeterministic time and deterministic time, looking for consequences relating to the known separations. We do this by considering alternating time. Let $\Sigma_l(t(n))$ be the class of languages accepted by $t(n)$-time bounded alternating Turing machines [10] which make at most l alternations. It is easy to see the following containments: For each $l \geq 1$, DTIME$(t(n)) \subseteq \Sigma_l(t(n)) \subseteq$ ATIME$(t(n)) \subseteq$ DSPACE$(t(n))$. We show that the set of languages accepted by $t(n)$-time bounded alternating Turing machines which make at most two alternations strictly contains the set of languages accepted in deterministic time $t(n)$. This substantially refines the separation DTIME$(t(n)) \subset$ ATIME$(t(n))$. Our main theorem is:

Theorem A For each time-constructible function $t(n)$, DTIME$(t(n)) \subset \Sigma_2(t(n))$.

Thus, our result shows that not only is space more powerful than time, but merely two alternations are more powerful than deterministic time.

Our result has very interesting consequences regarding the two mentioned generalizations of the DTIME$(n) \subset$ NTIME(n) result. Specifically, we have the following corollary:

Corollary B If NTIME$(n) =$ co-NTIME(n), then for each time-constructible function $t(n)$, NTIME$(t(n)) \neq$ DTIME$(t(n))$.

Corollary B implies that NTIME$(n) \neq$ co-NTIME(n) or, for each $k \geq 1$, NTIME(n^k) $\neq$ DTIME(n^k). This shows that one of the two possible generalizations of DTIME(n) $\subset$ NTIME(n) has to be true, shedding some light on these persistent open problems.

Our proof techniques relate to the earlier ones as follows: The separation DTIME$(t(n))$ $\subset$ DSPACE$(t(n))$ was obtained by speeding-up of deterministic Turing machines by a factor of $\log t(n)$: DTIME$(t(n)) \subseteq$ DSPACE$(t(n)/\log t(n))$[17]. Similarly, the separation DTIME$(t(n)) \subset$ ATIME$(t(n))$ was obtained by speeding-up of deterministic Turing machines by a factor of $\log t(n)$: DTIME$(t(n)) \subseteq$ ATIME$(t(n)/\log t(n))$[11].

The separation DTIME$(n) \subset$ NTIME(n) was obtained by speeding-up of deterministic Turing machines by a factor of $\log^* n$ using only four alternations: DTIME$(t(n)$

$\log^* n) \subseteq \Sigma_4(t(n))$[31]. This small speed-up yields a contradiction if DTIME(n) = NTIME(n), thus establishing DTIME(n) $\subset$ NTIME(n).

Speed-up by a factor of $\log^* n$ seems to be inadequate to yield DTIME($t(n)$) $\subset$ $\Sigma_4(t(n))$ for $t(n)$ other than linear time. This is because the best known separation for deterministic time is DTIME($g(n)$) $\subset$ DTIME($f(n)$), if $f \in \omega(g \log g)$. The difficulty in strengthening this separation lies in simulating deterministic Turing machines with an arbitrary number of tapes on a deterministic Turing machine with a fixed number of tapes.

We can also attempt to use the $\log^* n$ speed-up to separate DTIME($t(n)$) from $\Sigma_4(t(n))$ using the hierarchies for Σ_k classes. For $\Sigma_1(t(n)) \equiv$ NTIME($t(n)$) the strongest known hierarchy theorem is due to [52]. It is shown there that if t and T are time-constructible functions such that $t(n+1) = o(T(n))$, then NTIME($t(n)$) $\neq$ NTIME($T(n)$). The NTIME hierarchy uses a key lemma due to Book, Greibach and Wegbreit[8] which states that a two-tape nondeterministic Turing machine can simulate an arbitrary tape nondeterministic Turing machine with only a constant factor time loss. In Appendix A, we show that this lemma generalizes to Σ_k machines. Using this lemma we can prove similar hierarchies for Σ_k classes. Thus, for all k, $\Sigma_k(t(n)) \neq \Sigma_k(T(n))$, if $t(n+1) = o(T(n))$. However, even with this efficient simulation result, $\log^* n$ speed-up is insufficient to give the separation DTIME($t(n)$) $\neq$ $\Sigma_4(t(n))$, for large $t(n)$ (for example, $t(n) = 2^{2^n}$).

Using the techniques of [31] we show that only two alternations are sufficient to achieve a $\log^* t(n)$ speed-up of deterministic Turing machines[1]; that is DTIME($t(n)$ $\log^* t(n)) \subseteq \Sigma_2(t(n))$. Under the hypothesis that $\Sigma_2(t(n)) =$ DTIME($t(n)$), we put the standard diagonalization language for DTIME($t(n)\sqrt{\log^* n}$) in DTIME($t(n)$) using only a *fixed number of tapes*. This yields a contradiction proving that DTIME($t(n)$) $\neq \Sigma_2(t(n))$; that is, DTIME($t(n)$) $\subset \Sigma_2(t(n))$.

In Appendix A we show that Book, Greibach and Wegbreit's[8] lemma generalizes to Σ_k machines. In Section 2.3 we show that DTIME($t(n)\log^* t(n)$) $\subseteq \Sigma_2(t(n))$; that is, deterministic Turing machines can be speeded-up by a factor of $\log^* t(n)$ using only two alternations. Using this result, in Section 2.4, we prove that DTIME($t(n)$) $\subset \Sigma_2(t(n))$, for each time constructible function $t(n)$. In Section 2.5, we prove Corollary B using our main theorem.

2.2 Definitions and Preliminaries

Our model of computation is the alternating Turing machine(ATM) [10]. All our ATMs make a fixed number of alternations. We do not know if our results generalize to other standard models of computation, like random access machines (RAMs), because the techniques of Paul, Pippenger, Szemerédi, and Trotter [31] are specific to Turing machine(TM) computations.

A $\Sigma_l(t(n))$-machine is an ATM that runs in time O($t(n)$), on inputs of length n, that starts in an existential state, and makes at most $l-1$ alternations. Note that

[1] Several researchers have observed this result though the proof has not been published anywhere to the best of our knowledge.

deterministic TMs are ATMs where each transition is deterministic, and nondeterministic TMs are same as Σ_1-machines. We use the notation $\Sigma_l(t(n))$ to denote the class of languages accepted by $\Sigma_l(t(n))$-machines.

All our languages are over a fixed alphabet $\Sigma = \{0,1\}$. $\langle \cdot, \cdot \rangle$ denotes any standard pairing function. The following simple pairing function is sufficient for our purpose: encode 0 as 00 and 1 as 11 and let 01 stand for ",". For example, $\langle 010, 11 \rangle$ is encoded as 001100011111. Similarly, we can encode any arbitrary alphabet using only Σ. Also, any ATM can be encoded using only the alphabet Σ, using standard techniques. Furthermore, we assume that it can be easily checked (in linear time) if a string encodes an ATM.

NTIME($t(n)$) denotes the set of languages accepted by $\Sigma_1(t(n))$ machines in time O($t(n)$). DTIME($t(n)$) denotes the set of languages accepted by deterministic TMs in time O($t(n)$). $L(M)$ denotes the language accepted by machine M. We use $\mathcal{N}$ to denote the set of natural numbers.

Definition 2.2.1 *A function $t : \mathcal{N} \mapsto \mathcal{N}$, $t(n) \geq n$, is time-constructible if there exists a deterministic Turing machine that on input 1^n halts in exactly $t(n)$ steps, for all n.*

The following theorem gives an equivalent characterization [20] of time-constructible functions. This characterization is sufficient for all time-constructible functions used in this chapter.

Theorem 2.2.2 *A function $t : \mathcal{N} \mapsto \mathcal{N}$, such that $t(n) \geq (1+\epsilon)n$ for some $\epsilon > 0$, is* *time-constructible if and only if it can be computed by a deterministic Turing machine in* $O(t(n))$ *steps.*

The following proposition will be useful in our proofs.

Proposition 2.2.3 *If* $t(n)$ *is time-constructible, then* $t(n)\lceil(\log^* n)^p\rceil$*, where* p *is a rational number, is also time-constructible.*

Proof. Given n, it is easy to compute a k such that, $2^{k-1} < n \leq 2^k$, in $O(n)$ steps; that is, it is easy to compute the most significant bit of $\lceil \log n \rceil$ in $O(n)$ steps. Let $k_0 = n$ on input 1^n. Starting from k_0 repeatedly find k_{i+1} such that, $2^{k_{i+1}-1} < k_i \leq 2^{k_{i+1}}$, until $k_i \leq 1$. Then $\log^* n = i$, by definition. The number of iterations is bounded by $1 + \log^* n$ and each iteration takes $O(\log n)$ steps (except the first one which takes $O(n)$ steps).

Taking the logarithm, multiplying and then exponentiating, we can compute $\lceil(\log^* n)^p\rceil$ in $O(\log^* n)$ steps. By definition, $t(n)$ can be computed in $O(t(n))$ steps, and finally, the results can be multiplied to give $t(n)\lceil(\log^* n)^p\rceil$ in $O(t(n)\lceil(\log^* n)^p\rceil)$ steps. $\Box$

We use $\sqrt{\log^* n}$ to denote $\lceil(\log^* n)^{1/2}\rceil$.

2.3 Speed-up using only two alternations

Balcázar, Diaz and Gabarró[3] contains an excellent presentation of the work of Paul, Pippenger, Szemerédi and Trotter[31][2]. An exercise at the end of Chapter 3 states

[2] The overall strategy and several techniques in this proof are due to [32] and [33].

that deterministic Turing machines can be speeded up by a factor of $\log^* n$ using only *two* alternations. In Section 4, we show that $\mathrm{DTIME}(t(n)) \neq \Sigma_2(t(n))$, for arbitrary time-constructible functions $t(n)$. The proof of this general result depends on the details of the Σ_2 speed-up algorithm. Thus, we present a complete proof of the Σ_2 speed-up. All the material in this section through Theorem 2.3.7 is from [3] with only minor changes.

Let M be a $t(n)$-time bounded deterministic TM, and let x be an input of length n. We partition the computation of M on x into "time segments" of $b(n)$ steps each, and we partition the tapes of M into blocks comprising of $b(n)$ contiguous cells, where $b(n)$ is a suitable function related to $t(n)$. Consider the following definition:

Definition 2.3.1 *[17] A deterministic machine M with m tapes is* block-respecting *with block size $b(n)$ if, during each time segment of its computation, each head of M visits cells in one and only one block.*

In other words, M is block-respecting if and only if each tape head crosses boundaries between blocks only at times which are multiples of $b(n)$. The following result from [17] will be used later on.

Proposition 2.3.2 *[17] Let $b(n)$, mapping $\mathcal{N}$ to $\mathcal{N}$, be computable in $\mathrm{O}(t(n))$ steps. If $t(n)$ is time-constructible, then every set L accepted in time $t(n)$ by a deterministic Turing machine with m tapes can be accepted in time $\mathrm{O}(t(n))$ by a block-respecting deterministic Turing machine, with block size $b(n)$, with $m+1$ tapes.*

Let M be an m-tape deterministic TM that is $t(n)$-time bounded and block-respecting with block-size $b(n)$, where n is the length of the input x. Then the computation of M on x can be partitioned into $a(n) = \lceil \frac{t(n)}{b(n)} \rceil$ time-segments, each one comprising of $b(n)$ consecutive steps. The following definition introduces the notion of a "dependency" graph on $a(n)$ vertices corresponding to these time-segments.

Definition 2.3.3 *[31] The* computation graph *associated with the computation of a block-respecting machine M on x is a labeled directed acyclic graph (dag) $G(V, E)$ where V is the ordered set $\{1, 2, \ldots, a(n)\}$, and nodes i and j, with $i < j$, are connected by an edge with label (x, y) if and only if*

1. *$j = i + 1$ with the label $(0, 0)$ or,*
2. *for some tape x of M a block y is scanned at time-segments i and j, but not at any time-segments in between i and j.*

Note that by definition, each node can have at most $m + 1$ predecessors and at most $m + 1$ successors; one due to the "time-segment line" (edges from i to $i + 1$), which gives the state dependencies and m due to one block on each of the m-tapes, which gives the tape dependencies between different time-segments.

Definition 2.3.4 *For any two nodes i and j in a dag G, we say that node i is a* predecessor *of node j if there is a path from i to j in G. We say i is an* immediate predecessor *of node j if there is an edge from i to j.*

Let $\{1, 2, \ldots, n\}$ be an ordered set of vertices, and denote by $H(n)$ the class of directed acyclic graphs on vertices $\{1, 2, \ldots, n\}$, in which the edges are defined in such a way that if (i, j) is an edge, then $i < j$. We say that edges (i, j) and (i', j') *cross* if $i < i' < j < j'$.

Let $H_1(n)$ denote the subclass of $H(n)$ comprising those graphs in which every vertex has at most one immediate predecessor and at most one immediate successor, and in which no two edges cross. Denote by $H_r(n)$ the subclass of $H(n)$ comprising those graphs that can be expressed as the union of r graphs in $H_1(n)$.

It is easy to prove the following proposition for a block-respecting deterministic TM with block size and length of each time-segment $b(n)$, and the number of time segments equal to $a(n)$.

Proposition 2.3.5 *[31] Let M be an m-tape block-respecting deterministic machine, with block size $b(n)$, that on input x of length n halts in time $t(n)$. Then the computation graph associated with the computation of M on x belongs to $H_{2m+1}(a(n))$, where $a(n) = \lceil \frac{t(n)}{b(n)} \rceil$ is the number of time-segments.*

The next theorem puts certain bounds on the number of predecessors of a set of nodes in a graph in H_r. Before that, we need a definition. Let $G - J$ denote the induced subgraph when nodes in J are removed from the graph G.

Definition 2.3.6 *[31] For any directed acyclic graph $G(V, E)$ and any integer m, a set of nodes $J \subseteq V$ is called an m-*segregator* for G if every node in $G - J$ has at most m predecessors in $G - J$.*

Thus, if the nodes of an m-segregator are removed from a graph G, then the number of predecessors of each node in the induced subgraph is bounded by m.

The following theorem states that there is a "small segregator" for $H_r(n)$, the graphs associated with the computations of block-respecting TMs.

Theorem 2.3.7 *[31] Let G be in $H_r(n)$ and let $n \geq 256$. Then G has a $6n/\log^* n$-segregator of size less than or equal to $15rn/\log^* n$.*

Recall that edges in the computation graph $G \in H_r(n)$, for some r, of a block-respecting deterministic TM denote dependencies (state or tape) between different time segments. Intuitively, Theorem 2.3.7 states that $G \in H_r(n)$ has a "small" set of nodes $J \subset G$, such that the number of dependencies in $G - J$ is "small".

Now we are ready to prove the result that for any time-constructible function $t(n)$, DTIME($t(n)\log^* n$) $\subseteq \Sigma_2(t(n))$. First, we discuss the proof conceptually to convince ourselves of its correctness. To this end we defer the details of the Σ_2 algorithm and the analysis of its time-complexity.

Consider a block-respecting deterministic $t(n)$-time bounded TM M. Let $b(n)$ be its block size and hence also the length of each time-segment. Then $a(n) = \lceil \frac{t(n)}{b(n)} \rceil$ is the number of time-segments. Let G be the computation graph of M on input x of length n. Then G is in $H_r(a(n))$ for some r. Recall that Theorem 2.3.7 states that there is a set J of nodes of size $O(a(n)/\log^* a(n))$ such the number of predecessors of each of the remaining nodes in $G - J$ is bounded by $O(a(n)/\log^* a(n))$. Now we make two observations about segregators and computation graphs which will be crucial to the proof. Both the observations are direct consequences of the definitions of segregator and computation graph.

Observation 1: Let J be a segregator of G in $H_r(a(n))$, as guaranteed by Theorem 2.3.7. Then the number of predecessors in $G - (J - \{j\})$ of each $j \in J$ is bounded by $6rn/\log^* n$. Hence, every node in G has only a "few" predecessors in $G - J$.

To see that Observation 1 is true, note that the number of edges incident on each $j \in J$ is bounded by r as $G \in H_r(n)$. If (i, j) is an edge and $i \in J$, then clearly i is not a predecessor of j in $G - (J - \{j\})$. If $i \notin J$, then by definition of the segregator, i can have at most $6n/\log^* n$ predecessors in $G - J$. Thus, the total number of predecessors of $j \in J$ in $G - (J - \{j\})$ is bounded by $6rn/\log^* n$.

Recall that each node of G corresponds to a time segment of $b(n)$ steps of computation such that each tape head remains within a block of size $b(n)$. If node i is an *immediate* predecessor of node j with an edge labeled (x, y), then the initial configuration for the block y on tape x, at time segment j, is the same as the final configuration of block y on tape x at the end of time segment i.

Observation 2: Let j be some node of the computation graph of the m-tape, block-respecting deterministic TM M on input x. Let P_j be the set of all immediate predecessors of j. The final block configurations for time-segments corresponding to each $i \in P_j$ completely determine the initial block configuration for time-segment j.

The observation follows directly from the definition of computation graph. Note that $i > 1$ always has at least one predecessor corresponding to the "time-segment line". This predecessor carries the state information. (For $i = 1$, the initial state is the same as the initial state of M on x). If i has m additional predecessors, then the final block configurations of time-segments corresponding to the predecessors gives the initial block configurations for blocks on each of the m-tapes, at time-segment i. On the other hand, if i has $r < m$ incident edges, besides the "time-segment edge", then $m-r$ blocks are visited for the first time at time-segment i. Thus, the configurations of these blocks is the same as in the initial configuration of M on input x.

Let M, $t(n)$, $b(n)$, $a(n)$ and G be as described before. Our Σ_2 Turing machine M' guesses the following existentially.

1. The block number of each tape head during each of the $a(n)$ time segments. The computation graph G is implicit in this information.

2. A "small segregator" J of G, of size $e(n) \leq 15(2m+1)a(n)/\log^* a(n) = \mathrm{O}(a(n)/\log^* a(n))$ whose existence is guaranteed by Theorem 2.3.7 if $G \in H_{2m+1}$.

3. The initial block configurations, $\overline{Init_J}$, for each of the $e(n)$ time segments corresponding to each node $j \in J$. Note that for each $j \in J$, this consists of $b(n)$ characters each (on the blocks on m tapes), tape head position for m tapes, and the state of M.

Our goal is to verify that the candidates guessed above are *consistent* with each other and with respect to the initial configuration of M on input x. In particular, we make sure that J is indeed a "small segregator" relative to G, that initial block configurations $\overline{Init_J}$ follow from each other and from the initial configuration of M on input x relative to G, and that G is correct relative to the final block configurations which follow deterministically from $\overline{Init_J}$. Later on, we'll show that this relative correctness implies absolute correctness, since M is a deterministic TM.

Let $\overline{Final_J}$ denote the final block configurations for each $j \in J$. The Σ_2 machine

M' computes $\overline{Final_J}$ deterministically from $\overline{Init_J}$ for each of the $e(n)$ nodes in J in time $\mathrm{O}(b(n))$. Thus $\overline{Final_J}$ is correct relative to $\overline{Init_J}$.

Note that if each node $j \in J$ has less than $e(n)$ predecessors in $G - (J - \{j\})$, then by Observation 1, J is a "small segregator". This is used to verify J relative to G.

Given a time-segment i, let $G_i = (V_i, E_i)$ be the subgraph of $G = (V, E)$ defined as follows: $V_i = \{k \in V \mid k \leq i\}$ and $E_i = \{(i_1, i_2) \in E \mid i_1, i_2 \in V_i, i_2 \notin J - \{i\}\}$. Thus, all the incoming edges for segregator nodes in V_i(other than node i, which may be a segregator node) are not present in G_i. Intuitively, we do not require the dependencies coming into segregator nodes since M' has already guessed candidates for the initial and final configurations at time-segments for each $j \in J$.

To verify the relative correctness of $\overline{Init_J}$ we make the following important observation.

Observation 3: Let $\overline{Final_J(j)}$ denote the final block configuration for time-segment j. Then, for each $i \in G$, the initial block configuration for time-segment i follows deterministically from $\overline{Final_J(j)}, j < i$, and from the initial configuration of M on x, relative to subgraph G_i.

Observation 3 follows inductively from Observation 2. The graph G_i may have several disjoint components. Consider the component which contains i. Clearly, G_i is a dag. Let L_1 be the set of nodes which have no incoming edges in G_i. Then, $j \in L_1$ only if $j = 1$ or $j \in J$. If $j = 1$ then deterministically compute the final block configuration of the time-segment from the initial configuration of M on input x. If

$j \in J$, then the final block configuration is $\overline{Final_J(j)}$. Thus, for each time-segment in L_1 we know the final block configurations. Let L_2 be the set of nodes in G_i which have immediate predecessors only in L_1. Using Observation 1, we know the initial block configurations of each time-segment in L_2. Thus, it is easy to compute deterministically the final block configurations of each time-segment in L_2. Proceeding in this fashion until $i \in L_k$, for some k, we obtain the initial block configuration for time-segment i. Furthermore, nodes in the component of G_i which contains node i, are either in J, or are predecessors of i in $G - J$. Thus, the number of nodes in this component of G_i is $O(e(n))$, ensuring that "not too much" deterministic simulation is required.

Observation 3 implies that we can verify the consistency of $\overline{Init_J(j)}$, for each $j \in J$, relative to $\overline{Init_J(i)}, i < j$, the subgraph G_j, and the initial configuration of M on x. In the final Σ_2 speed-up algorithm this verification is performed in parallel for different $j \in J$. [3]

Finally, we can verify that G is consistent with $\overline{Init_J}$ and $\overline{Final_J}$ as follows: It is easy to eliminate cases when, for example, a tape head is in block 1 during time-segment 1, and in block 5 during time-segment 2. Thus, what we have to verify is that the movement of a tape head at the end of a time-segment (tape heads move left, or right, or stay in the same block), is consistent with the guess in G of the block

[3] To perform this verification efficiently for each $j \in J$, let $P_j =$ predecessors of j in $G - (J - \{j\})$; that is, $P_j =$ nodes of G_j which lie in the same component as j, excluding the segregator nodes. M' guesses the initial block configurations of each node in P_j. Let this be a vector $\overline{Init_{P_j}}$. From this, M' deterministically computes the final block configurations $\overline{Final_{P_j}(i)}$ for each $i \in P_j$. Then it verifies $\overline{Init_{P_j}(i)}$ with the final block configurations of its immediate predecessors in parallel for different i.

position of tape heads during the next time segment. If this is done for each $j \in J$ and for each predecessor of j, this verifies the consistency of G with respect to final block configurations.

Thus, given G, J, and $\overline{Init_J}$, the checks described above verify that these are consistent with each other and with the initial configuration of M on x. Now we show that this relative correctness implies absolute correctness, using the fact that M is a deterministic TM. Let G, J, and $\overline{Init_J}$ be relatively correct with respect to each other and the initial configuration of M on x as described above, and let i be the leftmost time-segment for which either G or $\overline{Init_J(i)}$ is incorrect with respect to the actual computation of M on x; that is either, $i \in J$ and $\overline{Init_J(i)}$ is incorrect or, the tape head movement for some tape at the *end* of the computation of time-segment i is incorrect in G. By choice of i, G_i is consistent with the actual computation graph of M on x for all time segments prior to i, and also $\overline{Init_J(j)}, j < i$, is consistent with the actual computation of M on x. Since, $\overline{Init(i)}$ follows deterministically from $\overline{Init_J(j)}, j < i$, and G_i as in Observation 3, the relative consistency for time segment i actually implies that time segment i is consistent with the computation of M on x.

Now it is easy to see the speed-up process for a $t(n)\log^* t(n)$-time bounded deterministic TM to $\mathrm{O}(t(n))$ using only two alternations. Let $b(n)a(n) = \mathrm{O}(t(n)\log^* t(n))$. Guess the tape head positions (and hence G) and a segregator J of size $e(n) = \mathrm{O}(a(n)/\log^* a(n))$, whose existence is guaranteed by Theorem 2.3.7. Guess the initial configurations at the beginning of each of the $e(n)$ time-segments. This can be done in time $\mathrm{O}(b(n)e(n)) = \mathrm{O}(b(n)a(n)/\log^* a(n)) = \mathrm{O}(t(n))$ for an appropriate choice of $a(n)$. Computing the final configurations for each of the $e(n)$ segregator nodes takes $O(b(n)e(n))$ steps. Using these, we can universally verify the relative consistency of G and the initial configurations in $O(b(n)e(n))$ steps.

Theorem 2.3.8 *For each time-constructible function $t(n)$,* $\mathrm{DTIME}(t(n)\log^* t(n)) \subseteq \Sigma_2(t(n))$.

Proof. Let $L \in \mathrm{DTIME}(t(n)\log^* t(n))$ be accepted by a m-tape, block-respecting, with block size $b(n) = \lceil t(n)^{2/3} \rceil$, deterministic TM M which always halts within $ct(n)\log^* t(n)$ steps. This is possible due to Proposition 2.3.2. Let $a(n) = \lceil ct(n)^{1/3} \log^* t(n) \rceil$. Let us construct a Σ_2 machine M', which has the maximum of $m+4$ tapes or the number of tapes required to compute $t(n)$, as follows:

Phase 0

> Deterministically on input x of length n, M' computes $t(n)$, $b(n) = \lceil t(n)^{2/3} \rceil$, $a(n) = \lceil t(n)^{1/3}\log^* t(n) \rceil$, and $e(n) = 15(2m+1)\lceil a(n)/\log^* a(n) \rceil$. This can be done in time $O(t(n))$. Note that by Theorem 2.3.7 the computation graph G of M has a segregator of size less than or equal to $e(n)$, for sufficiently large n. Furthermore, $O(b(n)e(n)) = O(t(n))$.

Phase 1 (existential)

1. Existentially the machine guesses the blocks where the heads of each tape are located during each time-segment. The block numbers for the first time-segment are all ones. If the block number for time-segment $i+1$ is different from the

block number for the time-segment i by more than 1, then reject. This step has a time cost of $O(ma(n) \log a(n))$. Note that this information, which is essentially a list with $a(n)$ entries (each entry is an m-tuple), implicitly describes the computation graph G of M on x. By Theorem 2.3.7, G has a $6a(n)/\log^* a(n)$-segregator of size at most $e(n)$, as $G \in H_{2m+1}(a(n))$, even if it is an incorrect computation graph.

2. Existentially guess a subset J of vertices of G of size $e(n)$ on a separate tape. This will correspond to the "small segregator" of G. To do this, M guesses $a(n)$ ones or zeroes and rejects if the number of ones exceed $e(n)$. Therefore this step costs $O(a(n))$.

3. Let $J = J \cup \{f\}$, where f is the last node of G, corresponding to the final time segment. For each node in J, existentially guess the initial configurations of each of the m blocks. This can be done in $O(m|J|(b(n) + 1)) = O(me(n)b(n)) = O(t(n))$.

   Deterministically compute the final block configuration for time-segments corresponding to each node in J. Save the initial and final configurations, $\overline{Init_J}$ and $\overline{Final_J}$. This step costs $O(me(n)b(n)) = O(t(n))$. M' uses m additional tapes to perform this simulation.

4. If the final configuration of the last time segment includes an accepting state then proceed to the next phase, otherwise reject.

   The total cost of this phase is $O(t(n))$.

At the end of *Phase 1* M' has candidates for G, J, initial and final block configurations for time segments corresponding to each node in J. Furthermore, final configurations are correct if the initial configurations are correct. Note that $\overline{Init_J}$ and $\overline{Final_J}$ are $e(n)$ vectors of size $mb(n)$ each, stored sequentially on m tapes.

Phase 2 (universal)

1. Universally select $j \in J$ to verify (in parallel) the consistency of $\overline{Init_J(j)}$. This costs $\mathrm{O}(\log a(n))$. Deterministically compute P_j, the set of all predecessors of j in $G - (J - \{j\})$ using two tapes besides the two tapes which have G and J. This step costs $\mathrm{O}(me(n)a(n)\log a(n))$ and requires walking back on the list guessed in the first step of Phase 1. If at any stage the size of P_j exceeds $e(n)$ then reject. Thus, if G is correct then J is indeed a "small segregator", if none of the j computation paths reject with P_j being too big.

   Now verify that the movement of all tape heads in $\overline{Final_J(j)}$ is consistent with the guess of tape head positions in G. If incorrect then reject. Note that this is done independently for each j on different computation paths. This step takes $\mathrm{O}(b(n)e(n) + a(n)\log a(n)) = \mathrm{O}(t(n))$ steps.

2. Universally guess the initial configuration for all nodes in P_j. This is a non-deterministic guess to verify $\overline{Init_J(j)}$ efficiently, as noted in Observation 3. Deterministically compute the final block configurations for the time-segments corresponding to each node in P_j. This step costs $\mathrm{O}(b(n)e(n))$. Save the initial

and final block configurations $\overline{Init_{P_j}}$ and $\overline{Final_{P_j}}$. Each of the initial block configurations in $\overline{Init_{P_j}}$ follow deterministically from final block configurations of segregator J, $\overline{Final_J}$, and the initial configuration of M on x. In the next step we verify $\overline{Init_{P_j}}$ in parallel. However, the initial block configurations $\overline{Init_{P_j}}$ are guessed universally. Hence, we accept on incorrect guesses, which is fine, as we know there is a computation path which guesses $\overline{Init_{P_j}}$ consistent with $\overline{Final_J}$.

3. Universally select $i \in P_j$ to verify $\overline{Init_{P_j}(i)}$ for different i in parallel. This step costs $O(\log a(n))$. Compute immediate predecessors of i, all of which are in $P_j \cup J$. This can be done with a cost of $O(ma(n)\log a(n))$. Note that we know the final configurations of each of the nodes in $J \cup P_j$. Hence, by Observation 2, we know the initial configuration for node i.

4. If the initial configuration of i is inconsistent with the final configurations of the immediate predecessors of i, then accept. This step can be done in $O(me(n)b(n))$ as this requires searching through the vectors $\overline{Final_{P_j}}$, $\overline{Final_J}$ and $\overline{Init_{P_j}(i)}$. There is one path in Step 2 which guesses the correct initial configurations $\overline{Init_{P_j}}$, consistent with $\overline{Final_J}$.

   Check that the movement of all tape heads in $\overline{Final_{P_j}(i)}$ is consistent with the guess of tape head positions in G. If this is incorrect then G is inconsistent and hence reject. Otherwise proceed to the next step.

5. For $j \in J$ guessed in Step 1 verify the initial configuration of j with the final configurations of its immediate predecessors, all of which are in $P_j \cup J$. If the ini-

tial configuration of j is inconsistent then reject. This step costs $O(me(n)b(n))$. This step together with the Step 1 verifies the consistency of $\overline{Init_J(j)}$.

Thus, the running time of M is $O(t(n))$ and the correctness follows from the preceding discussion. $\square$

2.4 Separation result

In this section we prove $\mathrm{DTIME}(t(n)) \subset \Sigma_2(t(n)$ for arbitrary time-constructible functions. The idea of the proof is as follows: Consider the standard diagonalization language $L_m(M)$ which diagonalizes over all deterministic m-tape Turing machines, time bounded by $f(n) = t(n)\sqrt{\log^* n}$. Using the speed-up of deterministic Turing machines, this language is accepted by a universal Σ_2 machine in time which is faster by a factor of $\log^* n$. The Σ_2 machine computes $t(n)$ in deterministic time $t(n)$ and then runs for $c_m t(n)/\sqrt{\log^* n}$. We show that the same Σ_2 machine accepts *each* $L_m(M), m > 0$, in this fashion. Thus, when we clock the Σ_2 machine with $t(n)$, it accepts any $L_m(M), m \geq 1$, for sufficiently large inputs in $O(t(n))$ steps.

Assuming $\Sigma_2(t(n)) = \mathrm{DTIME}(t(n))$, we obtain a $t(n)$-time bounded deterministic Turing machine with a fixed number of tapes which accepts the diagonalization language $\cup_{m>0} L_m(M)$, for sufficiently large inputs. This gives a contradiction showing that $\Sigma_2(t(n)) \neq \mathrm{DTIME}(t(n))$[4].

First we prove the following proposition using standard diagonalization techniques [16]. Let g be a function from $\mathcal{N}$ to $\mathcal{N}$. Let $\mathrm{DTIME}_m(g)$ denote the set of languages

[4] For a short proof for a restricted class of time-constructible functions see [25].

that can be accepted by deterministic TMs with m tapes in time $\mathrm{O}(g)$. Note that $\mathrm{DTIME}(g) = \cup_{m \geq 1}\mathrm{DTIME}_m(g)$. We show that $\mathrm{DTIME}(f) \not\subseteq \mathrm{DTIME}_m(g)$ for any fixed m, if f is time-constructible and f is $\omega(g)$. The crucial point is that Turing machines for $\mathrm{DTIME}(f)$ have an arbitrary number of tapes which can simulate any Turing machine with a fixed number of tapes with a time-loss of only a constant factor.

Proposition 2.4.1 *Let f be a time-constructible function such that $f \in \omega(g)$, for some function g. Let m be a fixed constant. Then,* $\mathrm{DTIME}(f) \not\subseteq \mathrm{DTIME}_m(g)$, *where* $\mathrm{DTIME}_m(g)$ *is the set of languages accepted by deterministic TMs (on some fixed alphabet) with m tapes and with running time bounded by* $\mathrm{O}(g)$.

Proof. We construct by diagonalization a set $L_m(M)$ in $\mathrm{DTIME}(f)$ which is not in $\mathrm{DTIME}_m(g)$. $L_m(M)$ is the language accepted by the machine M which we describe below. Let m' be the maximum of $m+2$, and the number of tapes required to compute $f(n)$. Note that the machine M has m' tapes, which is at least two more than any m-tape TM. We denote these two distinguished tapes as "clock" tape and "code" tape. Now consider the following description of M:

<u>Machine M(with m' tapes)</u>

Input w;

Compute $f(|w|)$ and store the value on the "clock" tape;

Remove the unique prefix 1*0 of w. Let the remaining string be u;

If u does not encode an m-tape deterministic TM then reject;

Store u on the "code" tape. Using the remaining at least m tapes, simulate M_u on w. The number of steps the simulating machine M runs during this simulation is clocked by the contents of the "clock" tape;

If M_u accepts before the clock runs out then reject else accept.

Note that computing $f(|w|)$, removing the prefix 1^*0, and checking the syntax of u can be done in at most $\mathrm{O}(f(|w|))$ steps as f is time-constructible. Furthermore, as M has more tapes than the simulated machine M_u, therefore each step of M_u can be simulated in c_u steps by M, where c_u is a constant which depends only on the encoding u. Thus, $L_m(M) \in \mathrm{DTIME}(f)$.

By way of contradiction, suppose $L_m(M) \in \mathrm{DTIME}_m(g)$ is accepted by a m-tape deterministic TM M_u (which can be encoded as string u) whose running time is bounded by $c \cdot g(n)$. Let $l > |u|$ be such that $f(l+1+|u|) \geq c_u \cdot c \cdot g(l+1+|u|)$, where c_u is the simulation time loss for the machine encoding u. Then, on input $w = 1^l 0u$, the machine M completes the simulation of M_u. Now, $w \in L_m(M)$ if and only if M_u does not accept w in time $f(|w|)$ if and only if $w \notin L(M_u) = L_m(M)$ which is a contradiction. Thus, $L_m(M) \notin \mathrm{DTIME}_m(g)$. $\square$

Now we prove our main theorem.

Theorem 2.4.2 *For each time-constructible function* $t(n)$, $\mathrm{DTIME}(t(n)) \neq \Sigma_2(t(n))$.

Proof. The proof is by contradiction. Let $t(n)$ be a time-constructible function such that $\Sigma_2(t(n)) = \mathrm{DTIME}(t(n))$. Consider the diagonalization language, described in detail in Proposition 2.4.1. It is $L_m(M) = \{1^l0u \mid n = l + 1 + |u|, u$ encodes a m-tape deterministic TM M_u and M_u does not accept u in *total simulation time* $t(n)\sqrt{\log^* n}$ of machine $M\}$.

The new diagonalization language in this theorem is the language accepted by the Σ_2-machine M' described below. It is essentially same as $\cup_{m>0} L_m(M)$, where the input 1^l0u encodes a deterministic TM M_u, which is block-respecting on input 1^l0u. That M_u is block-respecting on input 1^l0u is checked during simulation. However, M' speeds up the simulation of M_u by a factor of $\log^* n$ and hence needs $c_u \cdot t(n)\sqrt{\log^* n}/\log^* n$ steps to complete the simulation. Thus, when the total simulation time of M' is clocked by $t(n)$, then, for sufficiently large inputs, it completes the simulation of M_u.

Now we construct the fixed-tape Σ_2 machine M', very similar to the one in the proof of Theorem 2.3.8, with one main difference. On input 1^l0u, M' simulates deterministic TM M_u irrespective of how many tapes M_u has. This simulation is very similar to the one given in Lemma A.0.1. Thus, M' guesses a sequence of "displays" and moves of M_u, and then verifies this serially reusing the same work space. In the existential phase, it rejects on incorrect guesses and in the universal phase it accepts on the wrong guesses. Also, M' accepts only if M_u rejects.

Let $b(n) = \lceil t(n)^{2/3} \rceil$ and $a(n) = \lceil t(n)^{1/3} \rceil \sqrt{\log^* n}$. Let us construct the Σ_2 machine M' which has a maximum of 9 tapes[5], or the number of tapes required to compute $t(n)$.

[5]We can do with fewer tapes, but the exact number is not important to our proof.

Phase 0

1. Input $w, |w| = n$;

2. Compute $t(n)$ and store the value on the "clock" tape;

3. Remove the unique prefix 1^*0 of w. Let the remaining string be u;

4. If u does not encode a deterministic TM then reject;

5. Store u on the "code" tape. Let u encode a m-tape deterministic machine.

6. Deterministically compute $b(n) = \lceil t(n)^{2/3} \rceil$, $a(n) = \lceil t(n)^{1/3} \rceil \sqrt{\log^* n}$, and $e(n)$ $= 15(2m+1) \lceil a(n)/\log^* a(n) \rceil$.

   All the steps in this phase can be completed in time $\mathrm{O}(t(n))$. The running time of M' is clocked by $t(n)$ from now on; that is, the total simulation time-bound for Phase 1 and Phase 2 is $t(n)$. If a computation does not terminate within this time bound, M' rejects. The analysis done in Phase 1 and Phase 2 is for the time required for the completion of each step, if the steps were not clocked. It will be clear that for sufficiently large inputs the clock allows completion of all steps.

Phase 1 (existential)

1. Existentially the machine guesses the blocks where the heads of each tape are located during each time-segment. The block numbers for the first time segment are all ones. If the block number for time-segment $i+1$ is different from the

block number for the time-segment i by more than 1, then reject. This step has a time cost of $O(ma(n)\log a(n))$. Note that this information, which is essentially a list with $a(n)$ entries (each entry is an m-tuple), implicitly describes the computation graph G of M on x. By Theorem 2.3.7, G has a $6a(n)/\log^* a(n)$-segregator of size at most $e(n)$, as $G \in H_{2m+1}(a(n))$, even if it is an incorrect computation graph.

2. Existentially guess a subset J of vertices of G of size $e(n)$ on a separate tape. This will correspond to the "small segregator" of G. To do this, M guesses $a(n)$ ones or zeroes and rejects if the number of ones exceed $e(n)$. Therefore this step costs $O(a(n))$.

3. Let $J = J \cup \{f\}$, where f is the last node of G, corresponding to the final time segment. For each node in J existentially guess the initial configurations of each of the m blocks. This can be done in $O(m|J|(b(n)+1)) = O(me(n)b(n))$.

   For each $j \in J$, guess a sequence of $b(n)$ legal displays and legal moves of M_u. If the guessed move does not follow from the guessed display then reject. Verify the sequence of displays, *serially reusing the same work space*, rejecting if incorrect. Also, if any tape-head attempts to cross the block boundary, reject. Save the initial and final configurations, $\overline{Init_J}$ and $\overline{Final_J}$. This step costs $O(me(n)b(n))$, where the constant depends on M_u. M' uses 2 additional tapes to perform this simulation.

4. If the final configuration of the last time segment includes a *rejecting* state then

proceed to the next phase, otherwise reject.

The total cost of this phase is $O(me(n)b(n) + ma(n)\log a(n)))$, where the constant depends on M_u.

At the end of *Phase 1* M' has candidates for G, J, initial and final block configurations for time segments corresponding to each node in J. Furthermore, final configurations are correct if the initial configurations are correct. Note that $\overline{Init_J}$ and $\overline{Final_J}$ are $e(n)$ vectors of size $mb(n)$ each, stored sequentially on *one* tape.

Phase 2 (universal)

1. Universally select $j \in J$ to verify (in parallel) the consistency of $\overline{Init_J(j)}$. This costs $O(\log a(n))$. Deterministically compute P_j, the set of all predecessors of j in $G - (J - \{j\})$ using two tapes besides the two tapes which have G and J. This step costs $O(me(n)a(n)\log a(n))$ and requires walking back on the list guessed in the first step of Phase 1. If at any stage the size of P_j exceeds $e(n)$ then reject. Thus, if G is correct then J is indeed a "small segregator", if none of the j computation paths reject with P_j being too big.

   Now verify that the movement of all tape heads in $\overline{Final_J(j)}$ is consistent with the guess of tape head positions in G. If incorrect then reject. This step takes $O(b(n)e(n) + a(n)\log a(n))$ steps.

2. Universally guess the initial configuration for all nodes in P_j. Now compute the final configuration of each node in P_j as follows: Guess a sequence of $b(n)$ legal

displays and legal moves of M_u. If any of the guessed move does not follow from the guessed display then accept. Verify the sequence of displays, serially reusing the same workspace, accepting if incorrect. Also, if any tape-head attempts to cross the block boundary on a correct sequence of legal displays and legal moves, reject. This step costs time $O(mb(n)e(n))$. Save the initial and final block configurations $\overline{Init_{P_j}}$ and $\overline{Final_{P_j}}$. Each of the initial block configurations in $\overline{Init_{P_j}}$ follow deterministically from final block configurations of segregator J, $\overline{Final_J}$, and the initial configuration of M on x. In the next step we verify $\overline{Init_{P_j}}$ in parallel. However, the initial block configurations $\overline{Init_{P_j}}$ are guessed universally. Hence, we accept on incorrect guesses, which is fine, as we know there is a computation path which guesses $\overline{Init_{P_j}}$ consistent with $\overline{Final_J}$.

3. Universally select $i \in P_j$ to verify $\overline{Init_J(i)}$ in parallel, for different i. This step costs $O(\log a(n))$. Compute immediate predecessors of i, all of which are in $P_j \cup J$. This can be done with a cost of $O(ma(n) \log a(n))$. Note that we know the final configurations of each of the nodes in $J \cup P_j$. Hence, by Observation 2 (Section 4), we know the initial configuration for node i.

4. If the initial configuration of i is inconsistent with the final configurations of the immediate predecessors of i, then accept. This step can be done in $O(me(n)b(n))$ as this requires searching through the vectors $\overline{Final_{P_j}}$, $\overline{Final_J}$ and $\overline{Init_{P_j}(i)}$. There is one path in Step 2 which guesses the correct initial configurations $\overline{Init_{P_j}}$, consistent with $\overline{Final_J}$. Check that the movement of all tape heads in $\overline{Final_{P_j}(i)}$ is consistent with the guess of tape head positions in G. If

this is incorrect then G is inconsistent and hence reject. Otherwise proceed to the next step.

5. For $j \in J$ guessed in Step 1 verify the initial configuration of j with the final configurations of its immediate predecessors, all of which are in $P_j \cup J$. If the initial configuration of j is inconsistent then reject. This step costs $\mathrm{O}(me(n)b(n))$. This step together with the Step 1 verifies the consistency of $\overline{Init_J(j)}$.

The running time of M' is $\mathrm{O}(t(n))$ since Phase 1 and Phase 2 are clocked by $t(n)$. Hence, $L(M') \in \Sigma_2(t(n))$. Assuming $\Sigma_2(t(n)) = \mathrm{DTIME}(t(n))$, $L(M')$ is accepted by an m-tape block respecting deterministic TM M_u which runs in time $c \cdot t(n)$. Let TM M_u be encoded as string u. On input $1^l 0u$ of length n, the total time required by unclocked Phase 1 and Phase 2 is $cc_u t(n)\sqrt{\log^* n}/\log^* n$ for completing the simulation of M_u on $1^l 0u$, where c_u is only dependent on the machine M_u. Let $l > |u|$ be such that $t(l+1+|u|) = t(n) > cc_u t(n)\sqrt{\log^* n}/\log^* n$. Then, M' completes the simulation of M_u on input $1^l 0u$. Thus, M' accepts if and only if M_u rejects; that is, $1^l 0u \in L(M')$ iff $1^l 0u \notin L(M_u) = L(M')$. This is a contradiction proving that our assumption that $\Sigma_2(t(n)) = \mathrm{DTIME}(t(n))$ is incorrect. $\square$

2.5 Nondeterministic time versus deterministic time

In this section we prove Corollary B mentioned in the introduction, showing that one of the two generalizations of the result $\mathrm{DTIME}(n) \subset \mathrm{NTIME}(n)$ must be true. We need the following theorem which has also been proved in [50]. The theorem says that if the bottom level of the "alternating hierarchy based on fixed numbers of alternation" collapses, then every level collapses. Thus, if NTIME(n) = co-NTIME(n), then for each fixed l, $\Sigma_l(n)$ = NTIME(n) = co-NTIME(n). This collapse for linear time can be translated to any time-constructible $t(n)$ using padding arguments.

Theorem 2.5.1 *If* NTIME(n) = co-NTIME(n) *then for each l and for each time-constructible $t(n)$,* $\Sigma_l(t(n))$ = NTIME$(t(n))$ = co-NTIME$(t(n))$.

Proof. First we show that if NTIME(n) = co-NTIME(n) then for each l, $\Sigma_l(n)$ = NTIME(n) = co-NTIME(n), using induction on l. The case for $l = 1$ follows from the hypothesis.

Assume $\Sigma_l(n)$ = NTIME(n) = co-NTIME(n) for some fixed $l \geq 1$. Let $L \in \Sigma_{l+1}(n)$ be accepted by a nondeterministic TM M which accepts in time $c_m \cdot n$. Now construct another nondeterministic TM M' as follows:

<u>TM M'</u>

Input $\langle x, y \rangle$.

Enter a universal state. Check $|y| = c_m|x|$.

Run M on x using y as the guess for the first existential phase. Note that this is a deterministic simulation corresponding to the existential state of M. After the deterministic simulation of the first existential phase of M, M' simulates M exactly following its existential and universal states.

Note that the number of alternations of M' is one less than the number of alternations of M. Therefore, $L(M') \in$ co-$\Sigma_k(n)$ = NTIME(n) = co-NTIME(n), using the induction hypothesis. Let $L(M')$ be accepted by the nondeterministic TM M''. Now construct M''' as follows:

TM M'''

Input x.

Guess $y, |y| = c_m|x|$.

Run M'' on $\langle x, y\rangle$.

As M'' is a nondeterministic machine with $L(M'')$ in NTIME(n), therefore, $L(M''') \in$ NTIME(n). Also, it is easy to see that $L = L(M''')$. This shows that NTIME$(n) =$ co-NTIME(n) implies that for each $l \geq 1, \Sigma_l(n) =$ NTIME(n).

Now we show that if for some l, $\Sigma_l(n) =$ NTIME(n), then for each time-constructible function $t(n)$, $\Sigma_l(t(n)) =$ NTIME$(t(n))$. Let $L \in \Sigma_l(t(n))$. Define the following padded version of L which can be accepted in linear time, as follows:

$p(L) = \{x01^m \mid m = t(|x|) - 1 - |x| \text{ and } x \in L\}$. It is easy to see that $p(L) \in \Sigma_l(n) =$ NTIME(n). Let $p(L)$ be accepted by a nondeterministic TM $\hat{M}$. Using $\hat{M}$ we can accept L as follows:

TM $\widehat{M}$

Input x.

Compute $m = t(|x|) - 1 - |x|$.

Run $\hat{M}$ on $x01^m \in p(L)$.

The time-constructibility of $t(n)$ assures that m can be computed in $t(n)$ steps. It is easy to see that the above program recognizes L in time $O(t(n))$. $\square$

Now it is easy to prove the following interesting corollary stating that one of the two natural generalizations of the result $\mathrm{DTIME}(n) \subset \mathrm{NTIME}(n)$ must be true.

Corollary 2.5.2 *If* $\mathrm{NTIME}(n) = \text{co-NTIME}(n)$, *then for each time-constructible function* $t(n)$, $\mathrm{NTIME}(t(n)) \neq \mathrm{DTIME}(t(n))$.

Proof. Suppose $\mathrm{NTIME}(n) = \text{co-NTIME}(n)$. Then, by Theorem 2.5.1, for each time-constructible function $t(n)$, $\Sigma_2(t(n)) = \mathrm{NTIME}(t(n))$. This implies, using Theorem 2.4.2, that for each time-constructible function $t(n)$, $\mathrm{NTIME}(t(n)) \neq \mathrm{DTIME}(t(n))$. $\square$

Thus, one (or both) of the following statements must be true: $\mathrm{NTIME}(n) \neq \text{co-NTIME}(n)$ or, for each time-constructible function $t(n)$, $\mathrm{DTIME}(t(n)) \subset \mathrm{NTIME}(t(n))$.

2.6 Open Problems and Discussion

In Theorem 2.3.8, $\Sigma_2(t(n))$ serves two purposes. Firstly, the nondeterministic phase is used to guess the computation ahead of time, and then the universal phase is used to verify the guess in parallel. Secondly, $\Sigma_2(t(n))$ is used to "simulate" random access memory on fixed-tape Turing machine, by guessing the contents of cells in the existential phase and then verifying the guesses in parallel, in the universal phase.

Thus, it seems unlikely that the techniques in this paper can be used to speed-up deterministic time using only Σ_1, or nondeterministic time.

We do not know if there is any speed-up for $t(n)$-time bounded nondeterministic TMs, similar to the speed-up of deterministic TM as in Theorem 2.3.8. However, it is clear from the proof that if a nondeterministic TM makes only $O(t(n))$ nondeterministic moves followed by $O(t(n)\log^* n)$ deterministic moves, then it can be speeded up to $O(t(n))$ using only two alternations. In this case all the nondeterministic moves are made in Phase 1, and the resulting configuration is taken as the initial configuration. That this speed-up is possible using four alternations, is shown in [50]. We would like to know if NTIME($t(n)$) can be speeded up using multiple alternations or even deterministic space.

Note that our technique of speeding-up the diagonalization language has shown that any small speed-up is sufficient to give separations for deterministic time. For example, even a speed-up by a factor of $(\log^* n)^{1/1000}$ using only nondeterminism (Σ_1-machines) can separate deterministic time from nondeterministic time. Thus, we would like to know if deterministic time can be speeded-up (by any small amount whatsoever) using only nondeterminism. Also note that the separation given by Theorem 2.4.2 can be strengthened to DTIME($t(n)\log^{*p} n$) $\subset \Sigma_2(t(n))$, for any $p < 1$, using the same ideas.

Another approach for separating DTIME($t(n)$) and NTIME($t(n)$) is as follows: Show that DTIME($t(n)$) $=$ NTIME($t(n)$) implies that DTIME($t(n)$) $= \Sigma_2(t(n))$, for arbitrary time-constructible functions. However, we do not know how this can be achieved for functions other than $t(n) = cn$.

Another interesting open problem is whether NTIME$(t(n)) \neq$ co-NTIME$(t(n))$? Note that if we have a surprise similar to NSPACE$(t(n)) =$ co-NSPACE$(t(n))$ [18, 42], Corollary 2.5.2 would provide a separation result.

Finally, we would like to know if NTIME$(n) \not\subseteq$ DTIME(n^2) or, in general, NTIME(n) $\not\subseteq$ DTIME(n^k)? This question is unlikely to be answered by the techniques used in this chapter. Note that NTIME$(n) \not\subseteq$ DTIME$(n^k), k \geq 1$, if and only if P $\neq$ NP.

CHAPTER III

Closure Properties and Witness Reduction

Research is to see what everybody else sees, and to think what nobody else has thought.

— Albert Szent-Gyorgyi

3.1 Introduction

In this chapter we study some algebraic properties of several complexity classes. Consider any problem in NP, for example, the CLIQUE problem defined as follows:

$$\text{CLIQUE} = \{(G, k) \mid \text{Graph } G \text{ has a clique of size } k\ \}.$$

Thus, the CLIQUE problem is: Given a graph G and an integer k, does G have a clique of size k? Let $\#\text{Cliques}(G, k)$ denote the number of cliques of size k in graph G. Given two graphs G_1 and G_2, it is easy to produce a graph G such that $\#\text{Cliques}(G, k) = \#\text{Cliques}(G_1, k) + \#\text{Cliques}(G_2, k)$. It is also easy to produce a graph G such that, $\#\text{Cliques}(G, k) = \#\text{Cliques}(G_1, k) * \#\text{Cliques}(G_2, k)$.

In general, it is easy to add and multiply the number of solutions of instances of problems in NP. A natural question is: Is the same true for subtraction and division? This question was investigated by Ogiwara and Hemachandra[29].

To make these questions more precise, let #P denote the class of functions f such that there exists a polynomial time bounded nondeterministic Turing machine M such that $f(x)$ = number of accepting paths of the machine M on input x. It is easy to prove that the class #P is closed under addition; that is, given f_1 and f_2 in #P, there exists a $g \in$ #P such that for all x, $g(x) = f_1(x) + f_2(x)$. Also, #P is closed under multiplication. Note that the range of all #P functions is the set of natural numbers. Hence, we cannot talk about closure of #P under general subtraction or division. What Ogiwara and Hemachandra considered was proper subtraction ($a \ominus b = \max\{0, a-b\}$) and integer division ($a \div b = \lfloor \frac{a}{b} \rfloor$).

Ogiwara and Hemachandra[29] proved that #P is closed under proper subtraction iff #P is closed under integer division iff #P is closed under *every* other function in #P. Thus, closure of #P under proper subtraction or integer division implies closure of #P under every other #P function. Such functions, closure under which implies closure under every other function, are called "hard functions" by Ogiwara and Hemachandra. They also proved that if#P is closed under proper subtraction then the polynomial hierarchy(PH) collapses to a very weak class lying between P and NP. Thus, closure of #P under proper subtraction or integer division has very drastic consequences.

We address the following question: Why should closure under proper subtraction or integer division imply such drastic collapses? Is there something special about subtraction or division which is not true for addition and multiplication?

To explore these questions, in Chapter IV we define two natural extensions of #P: $\mathcal{Z}\#$P and $\mathcal{Q}\#$P. $\mathcal{Z}\#$P is the closure of #P under subtraction, and $\mathcal{Q}\#$P is the closure of $\mathcal{Z}\#$P under division. Thus, $f \in \mathcal{Z}\#$P iff there exists $g, h \in \#$P such that for all x, $f(x) = g(x) - h(x)$. Obviously, $\mathcal{Z}\#$P is closed under subtraction. However, consider the question of whether $\mathcal{Z}\#$P is also closed under proper subtraction. In Chapter IV, we prove that $\mathcal{Z}\#$P is closed under *proper* subtraction iff $\mathcal{Z}\#$P is closed under *integer* division iff $\mathcal{Z}\#$P is closed under *every* other function in $\mathcal{Z}\#$P. Thus, we show that proper subtraction and integer division remain hard for $\mathcal{Z}\#$P. We also prove that $\mathcal{Z}\#$P is closed under *proper* subtraction iff the counting hierarchy(CH) collapses to a weak counting class.

In the case of $\mathcal{Q}\#$P, we prove that $\mathcal{Q}\#$P is closed under *proper* subtraction iff $\mathcal{Q}\#$P is closed under *integer* division iff $\mathcal{Q}\#$P is closed under *every* other function in $\mathcal{Q}\#$P iff there are some "local" collapses in CH. Thus, we show that proper subtraction and integer division remain hard for $\mathcal{Q}\#$P as well.

In this chapter, we review and then extend the work of Ogiwara and Hemachandra in another direction. Since closure of #P under proper subtraction has such drastic consequences, #P therefore is unlikely to be closed under proper subtraction. Thus, it is natural to ask the question "Is #P closed under subtraction with a high probability?" In other words, given $f_1, f_2 \in \#$P, does there exists a $h \in \#$P which "approximates" $f_1(x) \ominus f_2(x)$, for all x, with a high probability. We formalize this notion of "approximation" and obtain results very analogous to those of Ogiwara and Hemachandra.

We study several similar results for several different function classes. Based on these results, we have identified *witness reduction* as the cause for collapses due to closure under integer division and proper subtraction. Our thesis is that any function that performs some form of *witness reduction* also effects some form of collapse. Furthermore, we believe that for a function to be "hard" for some function class, it must effect some form of *witness reduction*.

In the remainder of this introduction we define *witness reduction* and identify several results in complexity theory that have implicitly used *witness reduction*. Readers who are only interested in our theorems can skip the remainder of this introduction.

Witness reduction

Acceptance and rejection criteria for languages are defined by a predicate on the range of a *witness function*, where a *witness function* is a function that depends on the computation of some machine. A common *witness function* is the number of accepting paths of a nondeterministic machine(NTM). By varying the predicate associated with this function, various complexity classes can be characterized. For example, for the class NP, the *witness function* is the number of accepting paths and the predicate is 'greater than or equal to 1'; that is, the acceptance criterion is 'at least one accepting computation'.

Intuitively, *witness reduction* means that a witness function is somehow 'simplified'. For example, again consider the class NP. A very strong form of witness reduction would, given an arbitrary language L in NP accepted by an NTM M, argue that there is another NTM M' such that M and M' accept L, and whenever

M' accepts, it accepts on only one computation. Of course, if this type of *witness reduction* were possible, then NP=UP, an unlikely event.

More formally, a *witness function* maps a machine and input to a domain $\mathcal{D}$. For example, $\mathcal{D}$ is the set of natural numbers when the *witness function* is 'number of accepting paths'. The predicates for acceptance and rejection criteria are then defined over $\mathcal{D}$.

Let a language L be defined using predicates τ and $\overline{\tau}$ over domain $\mathcal{D}$ and a *witness function* f with range $\mathcal{D}$ as follows:

$$\begin{aligned} x \in L &\Leftrightarrow \tau(f(M, x)) \\ x \notin L &\Leftrightarrow \overline{\tau}(f(M, x)), \end{aligned}$$

where M is machine witnessing membership in L. Note that the input x is not available to the predicates τ and $\overline{\tau}$; the acceptance and rejection criteria are solely defined in terms of the output of the *witness function* f. We say L admits *witness reduction* if there is a machine M', *witness function* f', and predicates τ' and $\overline{\tau}'$ such that

$$\begin{aligned} x \in L &\Leftrightarrow \tau'(f'(M', x)) \\ x \notin L &\Leftrightarrow \overline{\tau}'(f'(M', x)), \end{aligned}$$

and $f'(M', x) \prec f(M, x)$, for all x. $\prec$ denotes some appropriate 'simplification' over the domain $\mathcal{D}$. For example, if $\mathcal{D}$ is the set of natural numbers, $\prec$ may be the usual 'less than or equal to' operator; over the set of integers, it could be the 'less than or equal to' operator on absolute values.

Thus, languages admitting *witness reduction* have multiple views, some of which are 'simpler' than others. The 'simplification' becomes important when the language is embedded in a larger computation, and can lead to interesting collapses.

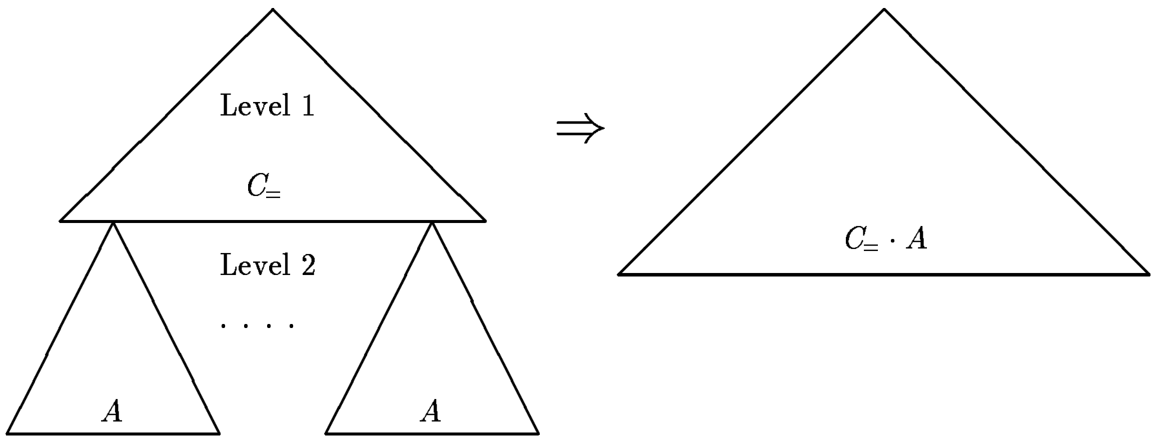

Figure 2: Two level guess tree

Witness reduction and collapsing hierarchies

One of the most important uses of *witness reduction* is to collapse hierarchies. As a very simple example, consider Figure 1. Suppose that the top tree corresponds to a $\mathrm{C}_=A$ computation; that is, the top tree accepts if and only if exactly $f(x)$ of its computations query a string $y \in A$, where $f \in \mathrm{PF}$. Now, suppose that A can be accepted by an NTM meeting the severe *witness reduction* constraints of UP; that is, $A \in \mathrm{UP}$. Then, the number of strings $y \in A$ queried by the top machine is exactly the same as the total number of accepting computations at the bottom of all of the level 2 trees, so that Figure 1 can be viewed as one large $\mathrm{C}_=\mathrm{P}$ tree, collapsing $\mathrm{C}_= \cdot \mathrm{UP}$ into $\mathrm{C}_=\mathrm{P}$.

As another example, suppose that the top tree of Figure 2 again corresponds to a $\mathrm{C}_=A$ computation. Let $A \in \exists\mathrm{P}$ and on input x, let the length of queries made to the set A at level 1 be bounded by some polynomial $p(|x|)$. Now, suppose it is possible to make a guess W with a high probability, where $|W|$ is bounded by some polynomial in $|x|$, so that there is an NTM which, given W and a string $y \in A$ of length at most $p(|x|)$, meets the *witness reduction* constraints of UP. We call this *random witness reduction.* Then, the number of strings $y \in A$ queried by the top machine is exactly the same as the total number of accepting computations at the bottom of all the level 2 trees, for each of the right guesses W. Hence, the existential quantifier of $\mathrm{C}_= \cdot \exists\mathrm{P}$ can be collapsed into $\mathrm{C}_=\mathrm{P}$ with a high probability, giving a $\mathrm{BP} \cdot \mathrm{C}_=\mathrm{P}$ computation tree. This is illustrated in Figure 3.

Note that in the above example, A need not be in UP. All that is required for the collapse to occur is that A satisfy the severe UP criteria on most of the guesses W. In fact, we will capture this concept by defining the class Pseudo(UP). Thus, in the above example, it is sufficient to have $A \in$ Pseudo(UP) for the collapse to occur.

A Complex form of witness reduction

Please be warned that *witness reduction* does not literally mean reducing the number of accepting computations, as in the previous two examples. It can be much more complicated than this. For example, let $L \in$ NP with the *witness function* 'number of accepting paths' and the predicate 'greater than or equal to 1'. Now, change the *witness function* to 'number of accepting paths (mod 2)', so that the range of the *witness function* is $\{0, 1\}$ and change the predicate to 'is equal to 1'; that is, the new acceptance criteria is odd number of accepting computations. This corresponds to the class parity P denoted by $\oplus$P. Given a string $x \in L$, using

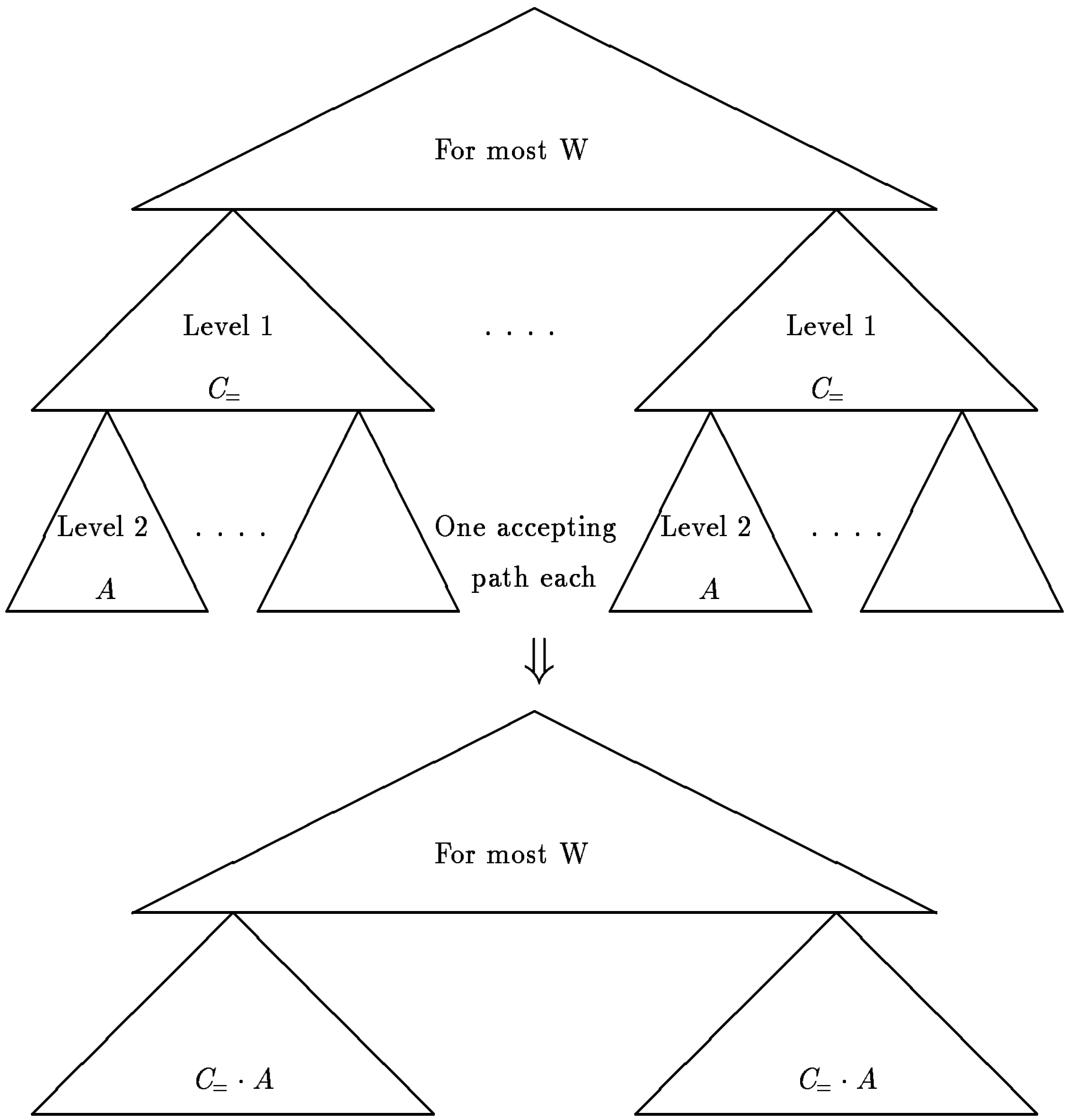

Figure 3: Random Witness Reduction

Valiant and Vazirani's [48] reduction, or the construction given in [15], it is possible to construct a polynomial number of machines so that one of them satisfies the new predicate under the new *witness function*, with a high probability. Using the fact that $\oplus$P is closed under Turing reductions[30], all of these machines can be merged into one machine with the acceptance criteria 'odd number of accepting computations', giving NP $\subseteq$ BP $\cdot \oplus$P or $\exists$P $\subseteq$ BP $\cdot \oplus$P. Thus, NP admits $\oplus$P *witness reduction* with a high probability.

PH $\subseteq$ BP $\cdot \oplus$P as an application of witness reduction

The complicated form of *witness reduction* in NP $\subseteq$ BP $\cdot \oplus$P was subsequently used by Toda [44] to collapse PH in BP$\cdot \oplus$P. To see how this works, using the *witness function* 'number of rejecting paths', the number of witnesses for non-membership in $\forall P$ can be reduced from many to one, with a high probability. This yields $\forall P \subseteq$ BP $\cdot \oplus$P. Note that This can be done with $\forall \cdot K$, for various complexity classes K, not necessarily only $\forall \cdot$ P.

Now consider a two level tree corresponding to $\exists \cdot \forall$P. Each of the level 2 trees corresponding to $\forall$P acceptance criteria can be replaced by trees corresponding to BP $\cdot \oplus$P acceptance criteria. Also, due to the second amplification lemma [38], one random guess is sufficient for each of the level 2 trees. Thus, with high probability, $\exists \cdot \forall$P is simplified to $\exists \oplus$P.

Again applying Valiant and Vazirani's result to level 1 trees, and using the fact that $\oplus$P is closed under Turing reductions, the acceptance criteria for level 1 can be changed from $\exists \oplus$P to $\oplus \oplus$P, with a high probability. This allows collapsing of the two levels into one with a high probability, because the *witness function* 'number of accepting paths mod 2' is closed under composition; that is, $((a_1 \bmod 2) + \ldots + (a_k \bmod 2)) \bmod 2 = (a_1 + \ldots + a_k) \bmod 2$, where $a_1, \ldots, a_k$ correspond to number of accepting paths for level 2 trees.

This process can be iterated finite number of times yielding Toda's result, so that any finite sequence of alternating $\exists$ and $\forall$ quantifiers can be collapsed into $\oplus$P with a high probability. The above process also yields that PH $\subseteq$ BP $\cdot$ Mod$_k$P for any prime number k, using the fact that Mod$_k$P is closed under Turing reductions and composition, for any prime number k[4].

In the above example, note that $\oplus$P served two purposes. Firstly, it allowed amplification of probability due to its closure under Turing reductions and secondly, it allowed the final composition of *witness functions* to collapse the two levels into one. Thus, even though BP $\cdot \oplus$P may be a more 'complex' view for a language in NP, it still provides interesting results in the context of PH.

Containment of PH in various counting classes, with a high probability

Using the *witness reduction* provided by Valiant and Vazirani's theorem, Toda and Ogiwara [46] have shown that any finite number of existential and universal quantifiers can be collapsed into various counting classes, with a high probability. This yields, $\mathcal{C} \cdot \mathrm{PH} \subseteq \widehat{\mathrm{BP}} \cdot \mathcal{C}$, for various counting classes $\mathcal{C}$. The $\widehat{\mathrm{BP}}$ operator is similar to the BP operator with "free amplification" built into the definition; that is, the error probability can be made arbitrarily small by definition.

Collapsing two levels into one

Now we formally define the notion of collapsing two levels of a guess tree into one. Let a language L be accepted as follows, using predicates τ_1, $\overline{\tau_1}$, and a *witness function* f such that

$$x \in L \Leftrightarrow \tau_1(f(M_A, x))$$
$$x \notin L \Leftrightarrow \overline{\tau_1}(f(M_A, x)),$$

where M_A is an NTM which guesses a string y of length at most $p_1(|x|)$ and accepts iff $\langle x, y\rangle \in A$, for some fixed set A. The guess tree for M_A would correspond to the first level of Figure 2.

Let set A be accepted as follows, using predicates τ_2, $\overline{\tau_2}$, and a *witness function* g such that

$$\langle x, y\rangle \in A \Leftrightarrow \tau_2(g(M_B, \langle x, y\rangle))$$
$$\langle x, y\rangle \notin A \Leftrightarrow \overline{\tau_2}(g(M_B, \langle x, y\rangle)),$$

where M_B is an NTM which guesses a string z of length at most $p_2(|\langle x, y\rangle|)$ and accepts iff $\langle x, y, z\rangle \in B$, for some set B in a complexity class $\mathcal{C}$. The guess tree for M_B would correspond to the second level of Figure 2. The two levels can be collapsed into one with respect to a *witness function* h if there exists a set $C \in \mathcal{C}$, a polynomial p, predicates τ and $\overline{\tau}$, and an NTM M_C such that

$$x \in L \Leftrightarrow \tau(h(M_C, x))$$
$$x \notin L \Leftrightarrow \overline{\tau}(h(M_C, x)).$$

Note that the machine M_C queries set C, which lies in the same complexity class as set B queried at the bottom of the two level guess tree.

We say that the two levels can be collapsed into one with a high probability, with respect to *witness function* h, if for all polynomials e there exist polynomials b and

p, a set $C \in \mathcal{C}$, predicates τ and $\overline{\tau}$, and an NTM M_C such that

$$\text{Prob}\begin{pmatrix} x \in L \Leftrightarrow \tau(h(M_C, \langle x, w\rangle)) \\ x \notin L \Leftrightarrow \overline{\tau}(h(M_C, \langle x, w\rangle)) \end{pmatrix} \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Oracle characterization of the counting hierarchy

The collapsing of two levels into one by changing *witness functions* was extensively used in the proofs of [47], to obtain the oracle characterization of the counting hierarchy, such as, $PP^{PP} = \text{C} \cdot \text{CP}$.

Hard functions for various function classes

The theme of witness reduction has also appeared in the recent work of Ogiwara and Hemachandra[29]. They introduced the idea of a function being hard for a class of functions. For example, Ogiwara and Hemachandra showed that #P is closed under composition with division, subtraction, and certain other functions if and only if #P is closed under composition with all #P functions. In addition, they show the interesting result that #P is closed under composition with these functions if and only if PH = CH = UP. More explicitly, Ogiwara and Hemachandra show that a function is hard for #P if and only if the interaction between the function and the class #P can reduce the number of accepting computations of NTMs on input x from an exact number, such as $f(x)$, to 1, and from any other number to 0. Thus, the hard functions for #P are those that can effect this very strong *witness reduction* in #P. Also, it is interesting to note that the *witness reduction* achieved by Ogiwara and Hemachandra is stronger than that of Toda, and hence produces stronger collapses.

In fact, we believe that this is exactly what makes a function hard for a class of functions: the ability of the function to interact with the class to effect appropriate types of *witness reduction*.

As mentioned earlier, in this chapter we build on the work of Ogiwara and Hemachandra in two ways. First, we introduce the idea of a function being randomly hard for a class of functions. We then show that each of the functions proved by Ogiwara and Hemachandra to be hard for #P are also randomly hard for #P. Carrying along the central theme that hardness is due to *witness reduction*, we show that a function is randomly hard for #P if and only if the function can be used to achieve the corresponding random versions of the *witness reduction* that Ogiwara and Hemachandra achieved for #P, and the corresponding random collapses: CH = PH = Pseudo(UP).

Our second extension to the work of Ogiwara and Hemachandra is also aimed at supporting the theme of *witness reduction* being the reason why certain functions are hard for a class of functions. We do this by considering two different *witness functions* other than 'number of accepting paths'. In particular, we consider not only number of accepting paths as *witness functions*, but also the AR function (equivalently, $\mathcal{Z}$#P) and the ratio of two AR functions (equivalently, $\mathcal{Q}$#P). AR functions are closed under subtraction while the ratio of two AR functions is closed under subtraction and division. However, proper subtraction and integer division, which reduce witnesses even in these new domains, do turn out to be hard functions for these new classes of functions. Thus, in the new domains, hardness is once again associated with some form of *witness reduction*.

Though the results of Ogiwara and Hemachandra for #P explicitly show the relationship between closure and *witness reduction*, it is not so obvious in other cases like SpanP, OptP, and RMidP. However, it is possible to prove the following results, which are stated in a slightly different way in [29].

- #P is closed under #P iff $\mathrm{C_=P} \subseteq \mathrm{UP}$.
- SpanP is closed under SpanP iff $\mathrm{C_=P} \subseteq \mathrm{NP}$.
- OptP is closed under OptP iff $\forall\mathrm{P} \subseteq \mathrm{NP}$.
- RMidP is closed under RMidP iff $\mathrm{C_=P} \subseteq \mathrm{NP}$.

The theme running through the proofs of all the above results is, indeed, *witness reduction*. These results should be compared with our results in Section 3.4, where the theme *random witness reduction* is used. Here, Almost(K), given any class K, is the class first defined by Nisan and Wigderson[28] using random oracles.

- #P is randomly closed under #P iff $\mathrm{C_=P} \subseteq \mathrm{Pseudo(UP)}$, where $\widehat{\mathrm{BP}} \cdot \mathrm{UP} \subseteq \mathrm{Pseudo(UP)} \subseteq \mathrm{Almost(UP)}$.
- SpanP is randomly closed under SpanP iff $\mathrm{C_=P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP} = \mathrm{Almost(NP)}$.
- OptP is randomly closed under OptP iff $\forall\mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP}$.
- RMidP is randomly closed under RMidP iff $\mathrm{C_=P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP}$.

We have not stated the proofs for RMidP; however, after reading the other proofs, it will be clear to the reader how to apply *random witness reduction* to the proofs of Ogiwara and Hemachandra to obtain random closure, based on the template provided in Section 3.3.

Finally, our main results for the new function classes are as follows:

- $\mathcal{Z}\#\mathrm{P}$ is closed under $\mathcal{Z}\#\mathrm{P}$ iff $\mathrm{C_{=}P} \subseteq \mathcal{Z}\mathrm{UP}$.
- $\mathcal{Q}\#\mathrm{P}$ is closed under $\mathcal{Q}\#\mathrm{P}$ iff $\mathrm{CP} \subseteq \mathrm{C_{=}P}$.
- $\mathcal{Z}\#\mathrm{P}$ is randomly closed under $\mathcal{Z}\#\mathrm{P}$ iff $\mathrm{C_{=}P} \subseteq \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP})$.
- $\mathcal{Q}\#\mathrm{P}$ is randomly closed under $\mathcal{Q}\#\mathrm{P}$ iff $\mathrm{CP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P}$.

This chapter is organized as follows: In Section 3.3 we describe the underlying structure of the proofs in [29] and in this thesis. In Section 3.4 we introduce and formalize the notion of randomly hard functions. We study what sort of collapses are produced by *random witness reduction* and show that division, subtraction, and span are randomly hard functions for #P. Then we obtain results for OptP and SpanP along the lines of Ogiwara and Hemachandra.

3.2 Definitions and Preliminaries

Our model of computation is polynomial time bounded Turing machines. The polynomials used in this chapter are bounds on resources used by Turing machines. Hence, we assume that the polynomials are nonnegative. All our languages are over a fixed alphabet Σ, containing two symbols $\{0, 1\}$. The set of natural numbers, the set of integers, and the set of rational numbers, are denoted by $\mathcal{N}$, $\mathcal{Z}$ and $\mathcal{Q}$, respectively. We denote by PF the class of polynomial-time computable functions. For a string, $x \in \Sigma^*$, $|x|$ denotes its length and for a set $A \subseteq \Sigma^*$, $\|A\|$ denotes its cardinality. We also use some standard pairing function $\langle \cdot, \cdot \rangle$. We assume that $|\langle x, y\rangle| \leq |\langle u, v\rangle|$, if $|x| \leq |u|$ and $|y| \leq |v|$. Furthermore, we use $\langle \cdot, \cdot, \cdot \rangle$ to denote $\langle \langle \cdot, \cdot \rangle, \cdot \rangle$.

We define the following functions on nondeterministic Turing machines (NTMs) and transducers.

Definition 3.2.1 *Given a nondeterministic machine N, let* $\mathrm{Acc}(N, x) =$ *number of accepting paths of N on input x, and* $\mathrm{Rej}(N, x) =$ *number of rejecting paths of N on input x. Also, let* $\mathrm{AR}(N, x) = \mathrm{Acc}(N, x) - \mathrm{Rej}(N, x)$.

Given a nondeterministic Turing transducer, we say y is an output string of M if, on some accepting computation path, M halts with y on its output tape. $\mathrm{Out}(M, x)$ *denotes the set of all output strings of M on x.*

Note that AR functions were used extensively to show closure properties for PP[5, 13, 14]. We review the definitions of complexity classes that we will discuss.

Definition 3.2.2

1. *A function $f : \Sigma^* \longmapsto \mathcal{N}$ is in* #P *iff there exists a polynomial time bounded nondeterministic machine M such that for all x,* $f(x) = \mathrm{Acc}(M, x)$.
2. *A language $L \in$* NP *iff there exists $f \in$* #P *such that for all x, $x \in L \Leftrightarrow f(x) \geq 1$.*

3. *A language* $L \in \mathrm{UP}$ *iff there exists* $f \in \#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow f(x) = 1$ *and* $x \notin L \Leftrightarrow f(x) = 0$.

4. *A language* $L \in \mathrm{C_{=}P}$ *iff there exists* $f \in \#\mathrm{P}$, *and* $g \in \mathrm{PF}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = f(x)$.

5. *A language* $L \in \mathrm{PP}$ *iff there exists polynomial time bounded NTM* M *such that for all* x, $x \in L \Leftrightarrow$ *more than half the computation paths of* M *on* x *accept.*

Next, we review the definitions of some standard operators.

Definition 3.2.3

1. *For a class* K, $\exists \cdot K$ *is the class of languages defined as follows: a set* L *is in* $\exists \cdot K$ *if there exists a polynomial* p *and a set* $A \in K$ *such that for every* $x \in \Sigma^*$,

$$x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y\rangle \in A\}\| \geq 1.$$

2. *For a class* K, $\forall \cdot K$ *is the class of languages defined as follows: a set* L *is in* $\forall \cdot K$ *if there exists a polynomial* p *and a set* $A \in K$ *such that for every* $x \in \Sigma^*$,

$$x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y\rangle \in A\}\| = 2^{p(|x|)}.$$

3. *For a class* K, $\mathrm{C} \cdot K$ *is the class of languages defined as follows: a set* L *is in* $\mathrm{C} \cdot K$ *if there exists a polynomial* p, *a function* $f \in \mathrm{PF}$, *and a set* $A \in K$, *such that for every* $x \in \Sigma^*$,

$$x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y\rangle \in A\}\| \geq f(x)$$

4. *For a class K, $C_= \cdot K$ is the class of languages defined as follows: a set L is in $C_= \cdot K$ if there exists a polynomial p, $f \in \mathrm{PF}$, and a set $A \in K$ such that for every $x \in \Sigma^*$,*

$$x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y\rangle \in A\}\| = f(x).$$

5. *For a class K, $\mathrm{BP} \cdot K$ is the class of languages defined as follows: a set L is in $\mathrm{BP} \cdot K$ if there exists a polynomial b and a set $A \in K$ such that for every $x \in \Sigma^*$,*

$$Prob(x \in L \Leftrightarrow \langle x, w\rangle \in A) \geq \frac{3}{4},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

The following proposition states the well-known quantifier characterization of complexity classes.

Proposition 3.2.4

1. $\mathrm{NP} = \exists\mathrm{P}$.

2. $\text{co-NP} = \forall\mathrm{P}$.

3. $\mathrm{PP} = \mathrm{CP}$.

Below are the definitions of two standard hierarchies, the polynomial time hierarchy(PH) and the polynomial-time counting hierarchy (CH).

Definition 3.2.5

1. $\mathrm{P} \in \mathrm{PH}$.

2. *If* $K \in \mathrm{PH}$, *then* $\exists \cdot K \in \mathrm{PH}$ *and* $\forall \cdot K \in \mathrm{PH}$.

3. PH *consists of only the classes described by 1 and 2.*

Definition 3.2.6

1. $\mathrm{P} \in \mathrm{CH}$.

2. *If* $K \in \mathrm{CH}$, *then* $\exists \cdot K \in \mathrm{CH}$, $\forall \cdot K \in CH$, *and* $\mathrm{C} \cdot K \in \mathrm{CH}$.

3. CH *consists of only the classes described by 1 and 2.*

To avoid introducing new notation, we will use CH and PH to denote both the family of classes as well as languages in the class. In the latter case $L \in \mathrm{CH}$ (or, $L \in \mathrm{PH}$) iff $L \in K$ for some $K \in \mathrm{CH}$ (or, $L \in \mathrm{PH}$). It should be clear from the context which interpretation we are referring to.

The following results are well-known.

Proposition 3.2.7

1. $\exists \cdot K \cup \forall \cdot K \subseteq \mathrm{C} \cdot K$, *for any class* K.

2. *[47] For any class* $K \in \mathrm{CH}$,

   (a) $\mathrm{C}_= \cdot K \subseteq \mathrm{C} \cdot K$.

   (b) $\exists \cdot \mathrm{C}_= \cdot K = \exists \cdot \mathrm{C} \cdot K$.

   (c) $\mathrm{C}_= \cdot \mathrm{C} \cdot K = \mathrm{C}_= \cdot \mathrm{C}_= \cdot K$.

3. $\mathrm{CH} = \cup_{k>0} \underbrace{\mathrm{C} \cdots \mathrm{C}}_{k} \mathrm{P} = \cup_{k>0} \underbrace{\mathrm{C}_= \cdots \mathrm{C}_=}_{k} \mathrm{P}$. *Here,* CH *is interpreted as a class of languages.*

4. *[39] L is in* $C_=P$ *iff there exists a nondeterministic polynomial-time Turing machine M and a polynomial p such that for every* $x \in \Sigma^*$,

   (a) $\mathrm{Acc}(M,x) + \mathrm{Rej}(M,x) = 2^{p(|x|)+1}$.

   (b) $x \in L$ *iff* $\mathrm{Acc}(M,x) = 2^{p(|x|)}$.

   (c) $x \notin L$ *iff* $0 < \mathrm{Acc}(M,x) < 2^{p(|x|)}$.

Next, we define some mathematical functions used in this thesis.

Definition 3.2.8

1. *For each finite set* $S \subset \mathcal{N}$, $\max(S)$ *denotes the largest member of* S.

2. *For any two* $a, b \in \mathcal{Q}$, *let* $a \ominus b$ *(proper subtraction) denote* $\max(\{0, a-b\})$.

3. *For any two* $a, b \in \mathcal{Q}, b \neq 0$, *let* $a \div b$ *(integer division) denote* $\lfloor \frac{a}{b} \rfloor$.

4. *For each number* $k \geq 2$ *and* $a_1, a_2, \ldots, a_k \in \mathcal{Q}$, *let* $span_k(a_1, a_2, \ldots, a_k)$ *denote the number of distinct values in* $\{a_1, a_2, \ldots, a_k\}$.

In preparation for the results of the results of Ogiwara and Hemachandra, consider the well-known result "#P is closed under addition"[35]; that is, given two functions $f_1, f_2 \in \#P$, there exists another function $h \in \#P$ such that, for all x, $h(x) = f_1(x) + f_2(x)$. Equivalently, for all x, $h(x) = \mathrm{Sum}(f_1(x), f_2(x))$, where $\mathrm{Sum}(y_1, y_2) = y_1 + y_2$. Similarly, one can talk about closure of #P under various other functions, like subtraction, division, etc.

In general, we can talk about closure of #P under a function f of arity i; that is, #P is closed under f(of arity i) iff for all $g_1, g_2, \ldots, g_i \in \#P$, there exists an $h \in \#P$

such that for all x, $h(x) = f(g_1(x), g_2(x), \ldots, g_i(x))$. In fact, we can go a step further and talk about closure of #P under a class of functions $\mathcal{CF}$. This would, of course, imply that for each $f \in \mathcal{CF}$, #P is closed under f.

As shown in [29] #P is closed under subtraction if and only if it is closed under PF. Thus, closure of #P under one function, namely subtraction, implies closure under every other function in PF. Ogiwara and Hemachandra call such functions PF-hard functions for #P, where PF refers to the fact that the functions are in PF. Finally, instead of just #P, it is possible to apply the same ideas to any function class $\mathcal{DF}$. In fact, Ogiwara and Hemachandra have obtained similar results for other function classes like OptP, SpanP, and MidP[22, 21, 45]. The following definition [29] formalizes the above mentioned concepts.

Definition 3.2.9

1. *Let $\mathcal{DF}$ be a class of functions. Let $i \geq 1$ and let $f : \mathcal{N}^i \longmapsto \mathcal{N}$. $\mathcal{DF}$* is closed under *f iff for all $g_1, g_2, \ldots, g_i \in \mathcal{DF}$, there exists an $h \in \mathcal{DF}$ such that, for all x, $h(x) = f(g_1(x), g_2(x), \ldots, g_i(x))$*

2. *Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. We say that $\mathcal{DF}$* is closed under *$\mathcal{CF}$ iff $\forall f \in \mathcal{CF}$ [$\mathcal{DF}$ is closed under f].*

3. *Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. Let $f \in \mathcal{CF}$. Then, f* is $\mathcal{CF}$-hard for *$\mathcal{DF}$ iff [$\mathcal{DF}$ is closed under $f \Rightarrow \mathcal{DF}$ is closed under $\mathcal{CF}$].*

The following are parts of the main theorem proved in Ogiwara and Hemachandra [29].

Theorem 3.2.10 (Witness Reduction Implies Collapsing) $\mathrm{C_{=}P} \subseteq \mathrm{UP} \Rightarrow \mathrm{CH} = \mathrm{PH} = \mathrm{UP}$.

Theorem 3.2.11 (Closure iff Witness Reduction) $\mathrm{C_{=}P} \subseteq \mathrm{UP} \Leftrightarrow \#\mathrm{P}$ *is closed under* $\#\mathrm{P}$.

Theorem 3.2.12 (Hard Functions for $\#\mathrm{P}$**)** *Proper subtraction*($\ominus$)*, integer division*($\div$) *and* $span_k$*, for all* $k \geq 2$*, are* $\#\mathrm{P}$*-hard for* $\#\mathrm{P}$.

Corollary 3.2.13 *(To Theorem 3.2.12)*$\#\mathrm{P}$ *is closed under* $\#\mathrm{P} \Leftrightarrow \#\mathrm{P}$ *is closed under* PF.

We have reproduced only parts of the original theorem from Ogiwara and Hemachandra to avoid cluttering the theme of this chapter. For example, Ogiwara and Hemachandra [29] have also shown that weak plurality and strong plurality are $\#\mathrm{P}$-hard. All our main results look similar to the above theorems, the theme running through all the results being *witness reduction.*

3.3 Structure of the Proofs

In this section we describe the underlying structure shared by the proofs for various function classes studied in [29] and in this thesis. These include the function classes $\#\mathrm{P}$, SpanP, OptP, $\mathcal{Z}\#\mathrm{P}$, and $\mathcal{Q}\#\mathrm{P}$. Let $\mathcal{DF}$ be one of these function classes. $\mathcal{DF}$ will have two special language classes, say $\mathcal{A}$ and $\mathcal{B}$, associated with it. In terms of $\mathcal{DF}$, $\mathcal{A}$, and $\mathcal{B}$, the theorems in this chapter and Chapter IV have the following format.

Theorem Template 1 (Witness Reduction Implies Collapsing) $\mathcal{A} \subseteq \mathcal{B} \Rightarrow$ Some form of collapse.

Theorem Template 2 (Witness Reduction iff Closure) $\mathcal{A} \subseteq \mathcal{B} \Leftrightarrow \mathcal{DF}$ is closed under $\mathcal{DF}$.

Theorem Template 3 (Hard Functions for $\mathcal{DF}$) Proper subtraction($\ominus$), integer division($\div$) and span_k, for all $k \geq 2$, are $\mathcal{DF}$-hard for $\mathcal{DF}$.

Corollary Template 4 $\mathcal{DF}$ is closed under $\mathcal{DF} \Leftrightarrow \mathcal{DF}$ is closed under PF.

$\mathcal{A} \subseteq \mathcal{B}$ denotes some form of *witness reduction*; the class $\mathcal{B}$ intuitively has a more restrictive acceptance and rejection criteria than class $\mathcal{A}$.

Proof Template for 1 We use $\mathcal{A} \subseteq \mathcal{B}$ to simplify computation trees, as explained in the introduction, so that computation trees can be merged. This allows us to collapse various hierarchies, depending on the classes $\mathcal{A}$ and $\mathcal{B}$. $\square$

Proof Template for 2[$\Longrightarrow$] Let $f \in \mathcal{DF}$ (witnessed by NTM M_f) be a k-ary function. Let $g_1, g_2, \ldots, g_k \in \mathcal{DF}$. Recall that we need to show, by definition, that $f(g_1, g_2, \ldots, g_k) \in \mathcal{DF}$, under the hypothesis $\mathcal{A} \subseteq \mathcal{B}$.

Define the following languages as graphs of the functions $g_1, g_2, \ldots, g_k$:

$$L_i = \{\langle x, y\rangle | y = g_i(x)\}, i = 1, 2, \ldots, k.$$

The selection of class $\mathcal{A}$, given $\mathcal{DF}$, allows us to prove $L_i \in \mathcal{A}$, which is contained in $\mathcal{B}$, under the hypothesis. Let $L_i \in \mathcal{B}$ be accepted by a machine M_i.

Now we construct an NTM M to witness $f(g_1(x), \ldots, g_k(x)) \in \mathcal{DF}$.

NTM M

1. Input x.

2. Guess y_i, $i = 1, 2, \ldots, k$. Each y_i is a candidate for $g_i(x)$.

3. Simulate NTM M_i on $\langle x, y_i \rangle$ sequentially, for each $i = 1, 2, \ldots, k$.

4. If any M_i rejects then M rejects and halts.

5. If for every $i = 1, 2, \ldots, k$, M_i accepts, then M simulates M_f on $\langle y_1, y_2, \ldots, y_k \rangle$ and accepts iff M_f accepts.

The restricted acceptance and rejection criteria for M_i will allow us to collapse the computation tree of Steps 2, 3, and 5 into one tree which computes f on the correct values of $g_1(x), g_2(x), \ldots, g_k(x)$. This shows that $f(g_1, \ldots, g_k) \in \mathcal{DF}$, when $\mathcal{A} \subseteq \mathcal{B}$.
$\square$

[$\Longleftarrow$] If $\mathcal{DF}$ is closed under $\mathcal{DF}$, then it is also closed under proper subtraction, as proper subtraction can be done in polynomial time, and all the function classes we study contain PF.

Starting with an arbitrary language $L \in \mathcal{A}$, we use the *witness function* for accepting L; the class $\mathcal{A}$ is chosen so that the *witness function* for accepting languages in $\mathcal{A}$ is in $\mathcal{DF}$. Then, using the hypothesis that $\mathcal{DF}$ is closed under proper subtraction, we obtain a simplified *witness function* in $\mathcal{DF}$, using which we show that $L \in \mathcal{B}$. $\square$

Proof Template for 3 We show that if $\mathcal{DF}$ is closed under proper subtraction, integer division, or span_k, for some $k \geq 2$, then $\mathcal{A} \subseteq \mathcal{B}$. This, together with Theorem Template 2, proves the theorem. $\square$

The corollary proof template 4 is immediate as proper subtraction is in PF, and is a hard function for $\mathcal{DF}$.

The above structure underlies all our results as well as those of Ogiwara and Hemachandra. In particular, our results in Section 4.3 have this exact structure with newly defined *witness functions.* The results in Section 3.4 are slightly different; there we study *random witness reduction* and randomly hard functions. In *random witness reduction*, machines only accept with arbitrarily high probability. We provide full details for the proofs for randomly hard functions for #P in Section 3.4. In subsequent sections, we provide the details only if the proofs are substantially different.

3.4 Random Witness Reduction

Ogiwara and Hemachandra [29] have shown that #P is closed under #P iff #P is closed under subtraction iff PH = CH = UP. Hence, #P is not very likely to be closed under subtraction. Thus, it is natural to ask the question, "Is #P closed under subtraction with a high probability?". In other words, given $f_1, f_2 \in$ #P, does there exists a function $h \in$ #P which "approximates" $f_1(x) \ominus f_2(x)$, for all x, with a high probability. In this section we formalize this notion of "approximation" and obtain results very analogous to those of Ogiwara and Hemachandra.

Definition 3.4.1

1. *Let $\mathcal{DF}$ be a class of functions. Let $i \geq 1$ and let $f : \mathcal{N}^i \longmapsto \mathcal{N}$.* $\mathcal{DF}$ is randomly closed under f *iff for all polynomials e, for all $g_1, g_2, \ldots, g_i \in \mathcal{DF}$, there exists an $h \in \mathcal{DF}$ and a polynomial b such that for all x,*

$$Prob(h(\langle x, w\rangle) = f(g_1(x), g_2(x), \ldots, g_i(x))) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly. Equivalently, we say that $\mathcal{DF}$ is closed under f with an arbitrarily high probability.

2. *Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. We say that $\mathcal{DF}$* is randomly closed under $\mathcal{CF}$ *iff $\forall f \in \mathcal{CF}$ [$\mathcal{DF}$ is randomly closed under f].*

3. *Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. Let $f \in \mathcal{CF}$. Then, f* is randomly $\mathcal{CF}$-hard for $\mathcal{DF}$ *iff [$\mathcal{DF}$ is randomly closed under $f \Rightarrow \mathcal{DF}$ is randomly closed under $\mathcal{CF}$].*

Note that all the definitions are very similar to those of Ogiwara and Hemachandra; the only difference being that we require functions to "behave properly" with arbitrarily high probability with respect to polynomial number of coin flips.

Before we state our results, we need several concepts. The following definition was first introduced by Toda and Ogiwara in [46].

Definition 3.4.2 *$L \in \widehat{\mathrm{BP}} \cdot \mathcal{C}$ iff for all polynomials e there exists $A \in \mathcal{C}$ and a polynomial b such that for all strings x,*

$$Prob(x \in L \Leftrightarrow \langle x, w\rangle \in A) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ *is chosen uniformly.*

The operator $\widehat{\mathrm{BP}}$ is essentially same as the standard BP operator (for example, see [38]), except that it has "free amplification" built into it. The following proposition summarizes the basic properties of $\widehat{\mathrm{BP}}$.

Proposition 3.4.3 *For all classes* $\mathcal{C}$ *and* $\mathcal{D}$*,*

1. $\widehat{\mathrm{BP}} \cdot \text{co-}\mathcal{C} = \text{co-}\widehat{\mathrm{BP}} \cdot \mathcal{C}$.
2. $\mathcal{C} \subseteq \mathcal{D}$ *implies* $\widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \mathcal{D}$.
3. *If* $\mathcal{C}$ *is closed under* $\leq_m^P$ *then* $\mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \mathcal{C}$.
4. *If* $\mathcal{C}$ *is closed under* $\leq_m^P$ *then* $\widehat{\mathrm{BP}} \cdot \widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \mathcal{C}$.

It is useful to keep in mind that for "standard" complexity classes, two $\widehat{\mathrm{BP}}$ operators can be merged into one. The following proposition provides the relationship between $\widehat{\mathrm{BP}}$ and BP operator.

Proposition 3.4.4 *For any class* $\mathcal{C}$*,*

1. $\widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \mathrm{BP} \cdot \mathcal{C}$.
2. *If* $\mathcal{C}$ *is closed under majority reduction then,* $\widehat{\mathrm{BP}} \cdot \mathcal{C} = \mathrm{BP} \cdot \mathcal{C}$.
3. *Let* M *be an oracle Turing machine which queries a set* $A \in \widehat{\mathrm{BP}} \cdot \mathcal{C}$ *and let all the queries be bounded in length by some polynomial in the length of the input. Then, for each polynomial* e *there exists an oracle Turing machine* M^**, a set* $B \in \mathcal{C}$*, and a polynomial* b *such that for all* x*,*

$$Prob(x \in L(M,A) \Leftrightarrow \langle x, w \rangle \in L(M^*, B)) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ *is chosen uniformly.*

Proof.

1. Obvious from the definitions.

2. The proof follows directly from the second amplification lemma [38].

3. The proof is essentially same as that of the operator interchange lemma[37]. Let M and A be as in the statement of the theorem. Let the length of all the queries made by M be bounded by a polynomial $p(n) \geq 0$, for all n. Let $e(n)$ be any fixed polynomial.

   As $A \in \widehat{\mathrm{BP}} \cdot \mathcal{C}$, there exists a $B \in \mathcal{C}$ and a polynomial $b(n)$ such that

   $$Prob(\neg(x \in A \Leftrightarrow \langle x, w\rangle \in B)) < \frac{1}{2^{p(|x|)+e(|x|)+1}},$$

   where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly. Or,

   $$Prob(\exists y, |y| \leq p(|x|), \neg(y \in A \Leftrightarrow \langle y, w\rangle \in B)) < \frac{2^{p(|x|)+1}}{2^{p(|x|)+e(|x|)+1}} = \frac{1}{2^{e(|x|)}},$$

   where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

   Now construct M^* as follows: On input $\langle x, w\rangle$, M^* simulates M. Whenever M queries q (to oracle set A), M^* queries $\langle q, w\rangle$ to B. The probability that M^* accepts iff M accepts is at least the probability of correct answers on all the queries, which exceeds $1 - \frac{1}{2^{e(|x|)}}$.

□

As polynomially bounded operators are special cases of oracle Turing machines whose query lengths are bounded by some polynomial in the length of the input, we have the following corollary to part 3 of Proposition 3.4.4.

Corollary 3.4.5 *For any class* $\mathcal{C}$

1. $\mathrm{C} \cdot \widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C} \cdot \mathcal{C}$.
2. $\mathrm{C}_{=} \cdot \widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C}_{=} \cdot \mathcal{C}$.
3. $\exists \cdot \widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \exists \cdot \mathcal{C}$.
4. $\forall \cdot \widehat{\mathrm{BP}} \cdot \mathcal{C} \subseteq \widehat{\mathrm{BP}} \cdot \forall \cdot \mathcal{C}$.

Thus, it is useful to keep in mind that $\widehat{\mathrm{BP}}$ can be "taken out" of polynomially bounded quantifiers. In the above, if the class $\mathcal{C}$ happens to be UP, then we can collapse it into the preceeding operator (*random witness reduction*).

Corollary 3.4.6

1. $\mathrm{C} \cdot \widehat{\mathrm{BP}} \cdot \mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C} \cdot \mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{CP}$.
2. $\mathrm{C}_{=} \cdot \widehat{\mathrm{BP}} \cdot \mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C}_{=} \cdot \mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C}_{=}\mathrm{P}$.
3. $\exists \cdot \widehat{\mathrm{BP}} \cdot \mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \exists \cdot \mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \exists \cdot \mathrm{P}$.
4. $\forall \cdot \widehat{\mathrm{BP}} \cdot \text{co-UP} \subseteq \widehat{\mathrm{BP}} \cdot \forall \cdot \text{co-UP} \subseteq \widehat{\mathrm{BP}} \cdot \forall \cdot \mathrm{P}$.

Next, we define the class Almost(K), given any class K. This was first defined by Nisan and Wigderson[28].

Definition 3.4.7 *$L \in$ Almost(K) iff $\mu(\{A \mid L \in K^A\}) = 1$, where K^A denotes class K relative to oracle set A.*

Let us understand what this definition means. Consider the standard lexicographic order of strings, $\epsilon, 0, 1, 00, 01, \ldots$. An oracle A corresponds to a real number $x_A \in [0,1]$ as follows: the ith decimal place of x_A has a 1 iff the ith string in this enumeration belongs to A. We view a finite string w as a prefix of the oracle set A, if w is a prefix of the corresponding real number x_A. Thus a set of oracles can be viewed as a subset of the closed interval $[0,1]$. Hence, the definition says that the measure of oracles A such that L lies in the class K relative to A is 1. Note that this does not necessarily mean that there is a fixed machine M such that $L \in M^A$ for almost all A. (Here, almost all is used to refer to the fact that the measure of such sets is 1.)

We need one more concept before presenting the main results. Consider the class UP. The *rejection criterion* for UP is not the logical negation of the *acceptance criterion.* If $L \in$ UP, then there is a two-sided agreement, one for acceptance and the other for rejection: there exists a polynomial time bounded machine M such that $x \in L \Leftrightarrow Acc(M,x) = 1$ (acceptance criterion) and $x \notin L \Leftrightarrow Acc(M,x) = 0$ (rejection criterion). Consider a language L such that for each polynomial e there exist a polynomial b and a polynomial time bounded machine M such that

$$\text{Prob}\begin{pmatrix} x \in L \Rightarrow \text{Acc}(M, \langle x, w\rangle) = 1 \\ x \notin L \Rightarrow \text{Acc}(M, \langle x, w\rangle) = 0 \end{pmatrix} \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly. Clearly, on 'bad' guesses (of w) the machine may not satisfy the severe UP criteria. For example, if $x \in L$ then for most guesses of w, M would have only one accepting path, but it may have more than one accepting path on 'bad' guesses of w. To capture this concept, we introduce the following definition.

Definition 3.4.8 $L \in \mathrm{Pseudo(UP)}$ *iff for each polynomial e there exists a polynomial b and a polynomial time bounded machine M such that*

$$Prob\left(\begin{array}{c} x \in L \Rightarrow Acc(M, \langle x, w\rangle) = 1 \\ x \notin L \Rightarrow Acc(M, \langle x, w\rangle) = 0 \end{array}\right) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

The following proposition shows that Pseudo(UP) is indeed very close to $\widehat{\mathrm{BP}} \cdot \mathrm{UP}$.

Proposition 3.4.9 $\widehat{\mathrm{BP}} \cdot \mathrm{UP} \subseteq \mathrm{Pseudo(UP)} \subseteq \mathrm{Almost(UP)}$.

Proof. $\widehat{\mathrm{BP}} \cdot \mathrm{UP} \subseteq \mathrm{Pseudo(UP)}$ follows directly from the definition. We show that $\mathrm{Pseudo(UP)} \subseteq \mathrm{Almost(UP)}$. The idea of the proof is essentially that of Bennet and Gill [7], where they show that $\mathrm{BPP}^A = \mathrm{P}^A$, for a random oracle A, with probability one. The probabilistic computation of Pseudo(UP) is simulated by queries to the random oracle. It is important to keep in mind that the random oracle is fixed before the computation begins.

Consider $L \in \mathrm{Pseudo(UP)}$, a fixed integer $k > 0$ and let $e(n) = 2n + k + 1$. Hence, there is a polynomial time bounded Turing Machine M and a polynomial $b(n)$ such that

$$\mathrm{Prob}\left(\begin{array}{c} x \in L \Rightarrow \mathrm{Acc}(M, \langle x, w\rangle) = 1 \\ x \notin L \Rightarrow \mathrm{Acc}(M, \langle x, w\rangle) = 0 \end{array}\right) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Construct M^* as follows: On input x, M^* queries the first $b(|x|)$ strings (according to the standard lexicographical order, $\epsilon, 0, 1, 00, \ldots$) to the random oracle A and stores the results (query answers) in a string w. It then simulates M on $\langle x, w\rangle$.

Let E_x be the set of oracles for which M^* makes an error on input x; that is, either $x \in L$ and M^* does not have exactly one accepting computation path, or $x \notin L$ and M^* has some accepting computation path. Let $w_1, w_2, \ldots, w_r$ denote the strings of length $b(|x|)$ that would cause M^* to make an error on x; in other words, M^* would error on x if w_i were a prefix of its oracle. Then, $r \leq \frac{2^{b(|x|)}}{2^{e(|x|)}}$. Furthermore, we can partition E_x into $E_x^1, E_x^2, \ldots, E_x^r$, where $E_x^j = \{A \in E_x \mid w_j \text{ is a prefix of } A\}$. Note that E_x^j is a closed interval of length $\frac{1}{2^{b(|x|)}}$. Thus,

$$\mu(E_x) \leq \left(\frac{2^{b(|x|)}}{2^{e(|x|)}}\right)\frac{1}{2^{b(|x|)}} = \frac{1}{2^{k+2|x|+1}}$$

Let $F_n = \cup_{|y|=n} E_y$ and $F = \cup_{n=0}^{\infty} F_n$. Thus, F is the set of oracles on which the machine makes an error on some input. Then,

$$\mu(F) \leq \mu(\cup_n F_n) \leq \sum_n \mu(F_n) \leq \sum_n \sum_{|y|=n} \mu(E_y) \leq \sum_n \sum_{|y|=n} \frac{1}{2^{2n+k+1}} \leq \sum_n \frac{2^n}{2^{2n+k+1}} \leq \frac{1}{2^k}$$

Since k was chosen arbitrarily, $\mu(F) = 0$. Thus, for almost all oracles, M^* accepts L.

□

The following proposition describes some easy properties of Pseudo(UP), which follow directly from the definitions.

Proposition 3.4.10

1. $\widehat{\mathrm{BP}} \cdot \mathrm{Pseudo(UP)} = \mathrm{Pseudo(UP)}$.

2. $\mathrm{Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP}$.

We also have the following interesting analogue of Corollary 3.4.6, which shows that Pseudo(UP) behaves like $\widehat{\mathrm{BP}} \cdot \mathrm{UP}$ with another operator in front. The proofs are essentially the same as in Corollary 3.4.6.

Proposition 3.4.11

1. $\mathrm{C} \cdot \mathrm{Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{CP}$.

2. $\mathrm{C}_= \cdot \mathrm{Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C}_=\mathrm{P}$.

3. $\exists \cdot \mathrm{Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \exists \cdot \mathrm{P}$.

4. $\forall \cdot \text{co-Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \forall \cdot \mathrm{P}$.

Finally, our proofs will use the following well-known results [46].

Proposition 3.4.12

1. $\mathrm{C} \cdot \mathrm{PH} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{CP}$.

2. $\mathrm{C}_= \cdot \mathrm{PH} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C}_=\mathrm{P}$.

PH in the above proposition refers to any finite number of $\exists$ and $\forall$ quantifiers over P.

3.4.1 Random Witness Reduction in #P

We now state our results for *random witness reduction*, which are analogous to *witness reduction* results of Ogiwara and Hemachandra [29].

Theorem 3.4.13 (Random Witness Reduction Implies Collapsing) $\mathrm{C_{=}P} \subseteq \mathrm{Pseudo(UP)} \Rightarrow \mathrm{CH} = \mathrm{PH} = \mathrm{Pseudo(UP)}$.

Theorem 3.4.14 (Random Witness Reduction iff Random Closure) $\mathrm{C_{=}P} \subseteq \mathrm{Pseudo(UP)} \Leftrightarrow \#\mathrm{P}$ *is randomly closed under* $\#\mathrm{P}$.

Theorem 3.4.15 (Randomly Hard Functions for #P) *Proper subtraction($\ominus$), integer division($\div$) and $span_k$, for all $k \geq 2$, are randomly $\#\mathrm{P}$-hard for $\#\mathrm{P}$.*

Corollary 3.4.16 $\#\mathrm{P}$ *is randomly closed under* $\#\mathrm{P} \Leftrightarrow \#\mathrm{P}$ *is randomly closed under* PF.

Proof of Theorem 3.4.13 $\mathrm{C_{=}P} \subseteq \mathrm{Pseudo(UP)}$ implies that $\mathrm{C_{=}} \cdot \mathrm{C_{=}} \cdot \mathrm{P} \subseteq \mathrm{C_{=}} \cdot \mathrm{Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{Pseudo(UP)} \subseteq \mathrm{Pseudo(UP)}$, using Propositions 3.4.10 and 3.4.11. Iterating this process gives $\forall k[\underbrace{\mathrm{C_{=}} \cdots \mathrm{C_{=}}}_{k} \mathrm{P} \subseteq \mathrm{Pseudo(UP)}]$. Thus, $\mathrm{PH} \subseteq \mathrm{CH} \subseteq \mathrm{Pseudo(UP)} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP} \subseteq \mathrm{PH}$. $\square$

Proof of Theorem 3.4.14[$\Longrightarrow$] Let e be a polynomial and $f \in \#\mathrm{P}$ (witnessed by NTM M_f) be a k-ary function. Let $g_1, g_2, \ldots, g_k \in \#\mathrm{P}$, where the running times of the corresponding NTMs are bounded by polynomials $p_1, p_2, \ldots, p_k$, respectively.

Define the following languages as graphs of the functions $g_1, g_2, \ldots, g_k$:

$$L_i = \{\langle x, y\rangle | y = g_i(x)\}, i = 1, 2, \ldots, k.$$

Claim 1: $L_i \in \mathrm{C_{=}P}$, for $i = 1, 2, \ldots, k$.

Proof. The following NTM M_i^* accepts L_i.

1. Input $\langle x, y\rangle$. Assume that $0 \leq y \leq 2^{p_i(|x|)}$.
2. Guess $b \in \{0, 1\}$.
3. if $b = 0$ then simulate the machine for g_i on x. M_i^* accepts iff this machine accepts.
4. if $b = 1$ then guess $l \in \{0, 1, 2, \ldots, 2^{p_i(|x|)}\}$. If $y \leq l$ then reject else accept.

Note that $Acc(M_i^*, \langle x, y\rangle) = 2^{p_i(|x|)} \Leftrightarrow y = g_i(x) \Leftrightarrow x \in L_i$. $\square$

Under our hypothesis, each $L_i \in$ Pseudo(UP). Choose a polynomial e^* such that $(1 - \frac{\sum_{1 \leq i \leq k} 2^{p_i(n)+1}}{2^{e^*(n)}}) \geq 1 - \frac{1}{2^{e(n)}}$, for all n. Then, there exists a polynomial time bounded NTM M_i and a polynomial b_i such that for all $\langle x, y\rangle$,

$$\mathrm{Prob}\begin{pmatrix} \langle x, y\rangle \in L_i \Rightarrow \mathrm{Acc}(M_i, \langle x, y, w\rangle) = 1 \\ \langle x, y\rangle \notin L_i \Rightarrow \mathrm{Acc}(M_i, \langle x, y, w\rangle) = 0 \end{pmatrix} \geq 1 - \frac{1}{2^{e^*(|x|)}},$$

where $w \in \{0, 1\}^{b_i(|\langle x, y\rangle|)}$ is chosen uniformly.

Now we construct an NTM M such that $\mathrm{Acc}(M, \langle x, w\rangle) = f(g_1(x), \ldots, g_k(x))$, with error probability less than $\frac{1}{2^{e(|x|)}}$. Choose a polynomial $b(n)$ such that $b(|x|) = \sum_{1 \leq i \leq k}(b_i(|\langle x, 1^{p_i(|x|)}\rangle|))$, and let $w \in \{0, 1\}^{b(|x|)}$ be chosen uniformly. Recall that by our choice of pairing function $b_i(|\langle x, y\rangle|) \leq b_i(|\langle x, 1^{p_i(|x|)}\rangle|)$, if $|y| \leq p_i(|x|)$.

NTM M

1. Input $\langle x, w\rangle$, $|w| = b(|x|) = \sum_{1\le i\le k}(b_i(|\langle x, 1^{p_i(|x|)}\rangle|))$.

2. Guess y_i, $|y_i| \le p_i(|x|)\}$, for each $i = 1, 2, \ldots, k$.

3. Let w_i be the ith string in w such that $|w_i| = b_i(|\langle x, 1^{p_i(|x|)}\rangle|)$.

4. Simulate NTM M_i on $\langle x, y_i, w_i\rangle$ sequentially, for each $i = 1, 2, \ldots, k$.

5. If any M_i rejects then M rejects and halts.

6. If for every $i = 1, 2, \ldots, k$, M_i accepts, then M simulates M_f on $\langle y_1, y_2, \ldots, y_k\rangle$ and accepts iff M_f accepts.

Claim 2:

$\text{Prob}(\text{Acc}(M, \langle x, w\rangle) = f(g_1(x), \ldots, g_k(x))) \ge 1 - \frac{1}{2^{e(|x|)}}$, where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Proof. Due to our choice of b, for each $i, 1 \le i \le k$, and for each $y_i, |y_i| \le p_i(|x|)$ we have,

$$\text{Prob}\begin{pmatrix} \langle x, y_i\rangle \in L_i \Rightarrow Acc(M_i, \langle x, y_i, w\rangle) = 1 \\ \langle x, y_i\rangle \notin L_i \Rightarrow Acc(M_i, \langle x, y_i, w\rangle) = 0 \end{pmatrix} \ge 1 - \frac{1}{2^{e^*(|x|)}},$$

or, for all y_i we have,

$$\text{Prob}\left(\bigwedge_{i=1}^{k}\begin{pmatrix} \langle x, y_i\rangle \in L_i \Rightarrow Acc(M_i, \langle x, y_i, w\rangle) = 1 \\ \langle x, y_i\rangle \notin L_i \Rightarrow Acc(M_i, \langle x, y_i, w\rangle) = 0 \end{pmatrix}\right) \ge (1 - \frac{\sum_{1\le i\le k} 2^{p_i(|x|)+1}}{2^{e^*(|x|)}}) \ge 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Let $w \in \{0,1\}^{b(|x|)}$ be such that the conjunctive condition holds. The probability of choosing such a w is at least $1 - \frac{1}{2^{e(|x|)}}$. For such a w, the conjunctive condition implies that on an incorrect guess $y_i \ne g_i(x)$, for some i, each computation of

M_i rejects, and hence M rejects and halts (at step 5 of NTM M). On the other hand, on the unique path corresoponding to the correct guesses, $y_i = g_i(x)$, for all i, $\mathrm{Acc}(M_i, \langle x, y_i, w\rangle) = 1$. Hence, for a correct guess w, there is exactly one path of NTM M (at Step 6) that accepts and simulates M_f with all the correct guesses $y_i = g_i(x), 1 \leq i \leq k$. This proves our claim. $\Box$

[$\Longleftarrow$] If #P is randomly closed under #P, then it is randomly closed under proper subtraction. We will show that this implies $\mathrm{C_=P} \subseteq \mathrm{Pseudo(UP)}$.

Let $L \in \mathrm{C_=P}$. By Proposition 3.2.7 there exists an NTM M and a polynomial p such that

$$\begin{array}{l} x \in L \Leftrightarrow \mathrm{Acc}(M, x) = 2^{p(|x|)} \\ x \notin L \Leftrightarrow \mathrm{Acc}(M, x) < 2^{p(|x|)}. \end{array}$$

Let $g_1(x) = \mathrm{Acc}(M, x) \in \#\mathrm{P}$, and $g_2(x) = 2^{p(|x|)} - 1$. Note that $x \in L \Leftrightarrow g_1(x) \ominus g_2(x) = 1$, and $x \notin L \Leftrightarrow g_1(x) \ominus g_2(x) = 0$. Under our hypothesis, for each polynomial e, there exists a function $h \in \#\mathrm{P}$ and a polynomial b such that

$$\mathrm{Prob}(h(\langle x, w\rangle) = g_1(x) \ominus g_2(x)) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Then, for a correct choice of w, we have $h(\langle x, w\rangle) = 1 \Leftrightarrow x \in L$, and $h(\langle x, w\rangle) = 0 \Leftrightarrow x \notin L$. Thus, for all e, there exists a TM M and a polynomial b such that

$$\mathrm{Prob}\left(\begin{array}{l} x \in L \Leftrightarrow \mathrm{Acc}(M, \langle x, w\rangle) = 1 \\ x \notin L \Leftrightarrow \mathrm{Acc}(M, \langle x, w\rangle) = 0 \end{array}\right) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly. This completes the proof of the theorem.

$\Box$

Proof of Theorem 3.4.15 We show that if #P is randomly closed under proper subtraction, integer division, or span_k, for some $k \geq 2$, then $\text{C}_=\text{P} \subseteq \text{Pseudo(UP)}$. This together with Theorem 3.4.14 proves the theorem.

As part of the proof of Theorem 3.4.14 we have already shown that if #P is randomly closed under proper subtraction, then $\text{C}_=\text{P} \subseteq \text{Pseudo(UP)}$. The proof that integer division is randomly #P-hard for #P is quite similar to that of proper subtraction; just take $g_2(x) = 2^{p(|x|)}$.

Now we show that span_k, for all $k \geq 2$, is randomly hard for #P. Let $L \in \text{C}_=\text{P}$. Then by Proposition 3.2.7, there exists an NTM M and a polynomial p such that

$$\begin{array}{l} x \in L \Leftrightarrow \text{Acc}(M,x) = 2^{p(|x|)} \\ x \notin L \Leftrightarrow \text{Acc}(M,x) < 2^{p(|x|)}. \end{array}$$

Let $g_1(x) = 2 \times \text{Acc}(M,x)$, $g_2(x) = 2^{p(|x|)} + \text{Acc}(M,x)$, and $g_i(x) = g_2(x)$, for $i = 3, \ldots, k$. Then, $g_1(x) = g_2(x) = 2 \times 2^{p(|x|)} \Leftrightarrow \text{Acc}(M,x) = 2^{p(|x|)} \Leftrightarrow x \in L$. Thus,

$$\begin{array}{l} x \in L \Leftrightarrow span_k(g_1(x), \ldots, g_k(x)) = 1 \\ x \notin L \Leftrightarrow span_k(g_1(x), \ldots, g_k(x)) = 2. \end{array}$$

Under our hypothesis, for all polynomials e, there exists a polynomial b_1 and an $h_1 \in \#\text{P}$ such that

$$\text{Prob}(h_1(\langle x, w_1 \rangle) = span_k(g_1(x), \ldots, g_k(x))) \geq 1 - \frac{1}{2^{e(|x|)+1}},$$

or,

$$\text{Prob}\left(\begin{array}{l} x \in L \Rightarrow h_1(\langle x, w_1 \rangle) = 1 \\ x \notin L \Rightarrow h_1(\langle x, w_1 \rangle) = 2 \end{array} \right) \geq 1 - \frac{1}{2^{e(|x|)+1}},$$

where $w_1 \in \{0,1\}^{b_1(|x|)}$ is chosen uniformly.

Again using the hypothesis, there exists a polynomial b_2 and an $h \in \#\mathrm{P}$ such that

$$\mathrm{Prob}(h(\langle x, w_1, w_2\rangle) = span_2(h_1(\langle x, w_1\rangle), 2)) \geq 1 - \frac{1}{2^{e(|x|)+1}},$$

where $w_2 \in \{0,1\}^{b_2(|\langle x, w_1\rangle|)}$ is chosen uniformly.

Note that, $h(x) = 1 \Leftrightarrow h_1(x) = 2$, and $h(x) = 2 \Leftrightarrow h_1(x) = 1$. Let $b(n) = b_1(n) + b_2(|\langle 1^n, 1^{b_1(n)}\rangle|)$. Thus,

$$\mathrm{Prob}\left(\begin{array}{c} x \in L \Leftrightarrow h(\langle x, w_1, w_2\rangle) = 2 \\ x \notin L \Leftrightarrow h(\langle x, w_1, w_2\rangle) = 1 \end{array}\right) \geq 1 - \frac{2}{2^{e(|x|)+1}},$$

where $w_1 w_2 \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Let $h \in \#\mathrm{P}$ be witnessed by NTM M^*. Now construct NTM M^{**} that on input $\langle x, w_1 w_2\rangle$ guesses two computations of M^* on $\langle x, w_1, w_2\rangle$ and accepts iff both the computations accept. Then,

$$\mathrm{Prob}\left(\begin{array}{c} x \in L \Leftrightarrow \mathrm{Acc}(M^{**}, \langle x, w_1 w_2\rangle) = 1 \\ x \notin L \Leftrightarrow \mathrm{Acc}(M^{**}, \langle x, w_1 w_2\rangle) = 0 \end{array}\right) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w_1 w_2 \in \{0,1\}^{b(|x|)}$ is chosen uniformly. Thus, $L \in \mathrm{Pseudo(UP)}$. $\square$

3.4.2 Random Witness Reduction in SpanP

In this section we apply *random witness reduction* to the class of functions measuring the number of distinct outputs of a nondeterministic machine. Our results should be contrasted with the results of Ogiwara and Hemachandra [29] for SpanP. Recall that $\mathrm{Out}(M, x)$ denotes the set of output strings of a Turing transducer M on input x.

Definition 3.4.17 *[21] A function $f : \Sigma^* \longmapsto \mathcal{N}$ is in* SpanP *iff there exists a polynomial time bounded nondeterministic Turing transducer M such that, for all x,*

$f(x) = \|\mathrm{Out}(M, x)\|$.

Note that $f(x) = 0$ iff all the computation paths of M reject. The following proposition summarizes the well-known properties of SpanP.

Proposition 3.4.18

1. $\mathrm{SpanP} \supseteq \#\mathrm{P}$.

2. *[9, 4] SpanP is closed under addition and multiplication.*

3. *[29] $f \in \mathrm{SpanP}$ iff there exists a polynomial p and a set $A \in \mathrm{NP}$ such that, for all x, $f(x) = \|\{y \mid |y| = p(|x|)$ and $\langle x, y\rangle \in A\}\|$.*

Proof.

1. Given a polynomial time bounded NTM M, construct a transducer M^* as follows: On input x, M^* simulates M keeping track of its computation path (guess bits) in a string y. If M terminates and accepts, then M^* outputs y. Clearly, for all x, $\|\mathrm{Out}(M^*, x)\| = \mathrm{Acc}(M, x)$.

2. Given two polynomial time bounded Turing transducers M_1 and M_2, construct M as follows: On input x, M guesses a bit b and then simulates M_1 if $b = 0$ and simulates M_2 if $b = 1$. Whenever, M_1 or M_2 outputs y, M outputs the string yb. Clearly, $\|\mathrm{Out}(M, x)\| = \|\mathrm{Out}(M_1, x)\| + \|\mathrm{Out}(M_2, x)\|$.

   To obtain $\|\mathrm{Out}(M_1, x)\| * \|\mathrm{Out}(M_2, x)\|$, construct M as follows: On input x, M simulates M_1 on x and if M_1 accepts and outputs y then M simulates M_2. If M outputs z, M outputs $\langle y, z\rangle$. It is easy to see that $\|\mathrm{Out}(M, x)\| = \|\mathrm{Out}(M_1, x)\| * \|\mathrm{Out}(M_2, x)\|$.

3. Let $f(x) = \|\mathrm{Out}(M, x)\|$, where M is a polynomial time bounded Turing transducer. Without loss of generality we may assume that each output of M, on input x, is of length $p(|x|)$. Let $A = \{\langle x, y\rangle \mid M \text{ on input } x \text{ outputs } y\}$.

   Conversely, given a set $A \in \mathrm{NP}$ and a polynomial p, consider a polynomial time bounded nondeterministic machine M that on input x, (a) guesses y of length $p(|x|)$ (b) nondeterministically tests if $\langle x, y\rangle \in A$ (c) outputs y along computation paths on which the check succeeded. Then, for every x, $\|\mathrm{Out}(M, x)\| = \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y\rangle \in A\}\|$.

$\square$

Now, we state our results for SpanP, which are presented in the format described in Section 3.3; the structure of proofs is similar to that of #P, and hence we do not provide all the details.

Theorem 3.4.19 (Random Witness Reduction Implies Collapsing) $\mathrm{C_{=}P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP} \Rightarrow \mathrm{CH} = \mathrm{PH} = \widehat{\mathrm{BP}} \cdot \mathrm{NP}$.

Theorem 3.4.20 (Random Witness Reduction iff Random Closure) $\mathrm{C_{=}P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{NP} \Leftrightarrow \mathrm{SpanP}$ *is randomly closed under* SpanP.

Theorem 3.4.21 (Randomly Hard Functions for SpanP) *Proper subtraction($\ominus$), integer division($\div$) and span$_k$, for all $k \geq 2$, are randomly* SpanP*-hard for* SpanP.

Corollary 3.4.22 SpanP *is randomly closed under* SpanP $\Leftrightarrow$ SpanP *is randomly closed under* PF.

To compare Theorems 3.4.20 and 3.4.14, note that $\widehat{\mathrm{BP}}\cdot\mathrm{NP} = \mathrm{Almost(NP)}$[28]. Thus, SpanP is closed under SpanP iff $\mathrm{C_=P} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{NP} = \mathrm{Almost(NP)}$, while #P is closed under #P iff $\mathrm{C_=P} \subseteq \mathrm{Pseudo(UP)}$, where $\widehat{\mathrm{BP}}\cdot\mathrm{UP} \subseteq \mathrm{Pseudo(UP)} \subseteq \mathrm{Almost(UP)}$.

Proof of Theorem 3.4.19 $\mathrm{C_=P} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{NP}$ implies that $\mathrm{C_=}\cdot\mathrm{C_=}\cdot\mathrm{P} \subseteq \mathrm{C_=}\cdot\widehat{\mathrm{BP}}\cdot\mathrm{NP} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{C_=P} \subseteq \widehat{\mathrm{BP}}\cdot\widehat{\mathrm{BP}}\cdot\mathrm{NP} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{NP}$, using Propositions 3.4.3 and 3.4.12. Iterating this process gives $\forall k[\underbrace{\mathrm{C_=}\cdots\mathrm{C_=}}_{k}\mathrm{P} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{NP}]$. Thus, $\mathrm{PH} \subseteq \mathrm{CH} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{NP} \subseteq \mathrm{PH}$. □

Proof of Theorem 3.4.20[$\Longrightarrow$] Let e be a polynomial and let $f \in \mathrm{SpanP}$, witnessed by Turing transducer M_f, be a k-ary function. Let $g_1, g_2, \ldots, g_k \in \mathrm{SpanP}$, where the running times of the corresponding Turing transducers are bounded by polynomials $p_1, p_2, \ldots, p_k$, respectively.

Now define the following languages as graphs of the functions $g_1, g_2, \ldots, g_k$:

$$L_i = \{\langle x, y\rangle | y = g_i(x)\}, i = 1, 2, \ldots, k$$

Claim 1: $L_i \in \mathrm{C_=}\cdot\mathrm{NP}$, for $i = 1, 2, \ldots, k$.

Proof. By proposition 3.4.18, there exists a set $A_i \in \mathrm{NP}$ and a polynomial t_i such that, for all x, $g_i(x) = \|\{u \mid \langle x, u\rangle \in A_i \text{ and } |u| = t_i(|x|)\}\|$. Construct NTM M_i^*, which accepts L_i, as follows:

1. Input $\langle x, y\rangle$.

2. Guess $u, |u| = t_i(|x|)$.

3. Accept iff $\langle x, u\rangle \in A_i$.

Step 3 runs the NP machine for A_i. Note that $\langle x, y\rangle \in L_i$ iff the number of u's such that $\langle x, y\rangle \in A_i$ is equal to y. On input $\langle x, y\rangle$, M_i^* can be modified by padding its computation tree with $2^{p_i(|x|)} - y$ accepting paths. Thus, $\langle x, y\rangle \in A_i$ iff the modified M_i^* has $2^{p_i(|x|)}$ accepting paths at the top level. Thus, $L_i \in \mathrm{C}_{=} \cdot \mathrm{NP}$. $\square$

Under our hypothesis, each $L_i \in \mathrm{C}_{=}\cdot\mathrm{NP} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{C}_{=}\mathrm{P} \subseteq \widehat{\mathrm{BP}}\cdot\widehat{\mathrm{BP}}\cdot\mathrm{NP} \subseteq \widehat{\mathrm{BP}}\cdot\mathrm{NP}$, using Propositions 3.4.12 and 3.4.3. Choose a polynomial e^* such that $(1 - \frac{\sum_{1 \le i \le k} 2^{p_i(n)+1}}{2^{e^*(n)}}) \geq 1 - \frac{1}{2^{e(n)}}$, for all n. Then, there exists a polynomial time bounded TM M_i and a polynomial b_i such that for all $\langle x, y\rangle$,

$$\mathrm{Prob}\Big(\ \langle x, y\rangle \in L_i \Leftrightarrow \mathrm{Acc}(M_i, \langle x, y, w\rangle) \geq 1\ \Big) \geq 1 - \frac{1}{2^{e^*(|x|)}},$$

where $w \in \{0,1\}^{b_i(|\langle x,y\rangle|)}$ is chosen uniformly.

We can construct a Turing transducer M such that $\|\mathrm{Out}(M, \langle x, w\rangle)\| = f(g_1(x), \ldots, g_k(x))$, with error probability less than $\frac{1}{2^{e(|x|)}}$. The machine M is exactly same as the corresponding machine for $\#$P(in the proof of Theorem 3.4.14), except in step 6, where M simulates M_f. In the case of SpanP, M_f is a transducer, and M outputs the same value as output by M_f.

Claim 2:

$\mathrm{Prob}(\|\mathrm{Out}(M, \langle x, w\rangle)\| = f(g_1(x), \ldots, g_k(x))) \geq 1 - \frac{1}{2^{e(|x|)}}$, where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Proof. The proof of the claim is similar to the one given for $\#$P. The only difference is that on the unique path corresponding to the correct guesses, $y_i = g_i(x)$, for all i, there may be several accepting paths of each M_i at step 5. However, each path, on which each M_i accepts, for $i = 1, 2, \ldots, k$, simulates M_f and outputs the same set $\mathrm{Out}(M, \langle y_1, \ldots, y_k \rangle)$.

This completes the proof. $\Box$

[$\Longleftarrow$] The proof is again similar to that for #P, except that h is computed by a Turing transducer. Modifying the transducer, to obtain a machine M that accepts iff the transducer accepts, provides the machine required for the proof. $\Box$

Proof of Theorem 3.4.21 The proof is essentially the same as that for #P. $\Box$

3.4.3 Random Witness Reduction in OptP

In this section we apply *random witness reduction* to the class of functions computing the maximum output of polynomial time bounded nondeterministic transducers. Recall that $\mathrm{Out}(M, x)$ denotes the set of outputs of the transducer M on input x.

Definition 3.4.23 *[22] A function $f : \Sigma^* \longmapsto \mathcal{N}$ is in* OptP *iff there exists a polynomial time bounded nondeterministic transducer M such that for all x, $f(x) = \max(\mathrm{Out}(M, x))$.*

Clearly, PF $\subseteq$ OptP. Krentel [22] gave the following characterization for the graph of an OptP function.

Proposition 3.4.24 *For any function $f \in$ OptP, the set $\{\langle x, y \rangle \mid y = f(x)\}$ is in* P^{NP}.

We now state our results for OptP. A function is randomly hard for OptP iff it can reduce the number of accepting paths of an NTM from 'not all' to 'none', with an arbitrarily high probability. In other words, if the machine rejects on some path, it rejects on all computation paths; that is, languages in co-NP can be accepted by a nondeterministic machine with an arbitrarily high probability.

Theorem 3.4.25 (Random Witness Reduction Implies Collapsing) $\forall \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} \Rightarrow \mathrm{PH} = \widehat{\mathrm{BP}} \cdot \exists \mathrm{P}$.

Theorem 3.4.26 (Random Witness Reduction iff Random Closure) $\forall \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} \Leftrightarrow$ OptP *is randomly closed under* OptP.

Theorem 3.4.27 (Randomly Hard Functions for OptP**)** *Proper subtraction($\ominus$), integer division($\div$) and span$_k$, for all $k \geq 2$, are randomly* OptP*-hard for* OptP.

Corollary 3.4.28 OptP *is randomly closed under* OptP $\Leftrightarrow$ OptP *is randomly closed under* PF.

Proof of Theorem 3.4.25 $\forall \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P}$ implies that $\widehat{\mathrm{BP}} \cdot \forall \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P}$. Taking complement of both sides we have, $\widehat{\mathrm{BP}} \cdot \exists \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \forall \mathrm{P}$. Hence, $\widehat{\mathrm{BP}} \cdot \exists \mathrm{P} = \widehat{\mathrm{BP}} \cdot \forall \mathrm{P}$. Now, using induction on the number of quantifiers, we show that $\mathrm{PH} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} = \widehat{\mathrm{BP}} \cdot \forall \mathrm{P}$. The base case is trivial as $\forall \mathrm{P} \cup \exists \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \forall \mathrm{P} = \widehat{\mathrm{BP}} \cdot \exists \mathrm{P}$. To see the induction step, note that $\exists \cdot \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists \cdot \exists \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P}$. Also, by Corollary 3.4.5, $\forall \cdot \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} = \forall \cdot \widehat{\mathrm{BP}} \cdot \forall \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \forall \cdot \forall \mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \forall \mathrm{P}$. Iterating this process gives $\mathrm{PH} \subseteq \widehat{\mathrm{BP}} \cdot \exists \mathrm{P} \subseteq \mathrm{PH}$. $\square$

Proof of Theorem 3.4.26[$\Longrightarrow$] This proof is similar to the corresponding proof for SpanP. (The graph language for functions in OptP is in $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{PH} \subseteq \widehat{\mathrm{BP}} \cdot \exists\mathrm{P}$, using Proposition 3.4.24, Theorem 3.4.25, and the hypothesis.) $\square$

[$\Longleftarrow$] If OptP is closed under OptP, then it is surely closed under proper subtraction. We show that this implies $\forall\mathrm{P} \subseteq \widehat{\mathrm{BP}} \cdot \exists\mathrm{P}$.

Let $L \in \forall\mathrm{P}$ be accepted by a nondeterministic machine M. Modify this machine to construct a nondeterministic transducer M' which simulates M, and outputs 0 if M accepts, and 1 if M rejects. Let $g(x) = \max(\mathrm{Out}(M', x)$. Note that $x \in L \Leftrightarrow 1 \ominus g(x) = 1$ and $x \notin L \Leftrightarrow 1 \ominus g(x) = 0$. Under our hypothesis, for each polynomial e, there exists a function $h \in \mathrm{OptP}$ and a polynomial b such that

$$\mathrm{Prob}(h(\langle x, w\rangle) = 1 \ominus g(x)) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

Then, for a correct choice of w, we have $h(\langle x, w\rangle) = 1 \Leftrightarrow x \in L$ and $h(\langle x, w\rangle) = 0 \Leftrightarrow x \notin L$; that is, there exists a Turing transducer which accepts and outputs 1 (on at least one path) iff $x \in L$. Modify the transducer to obtain a machine M^* which accepts iff the transducer accepts and outputs 1. Thus, for all e, there exists a machine M^* and a polynomial b such that

$$\mathrm{Prob}\left(\ x \in L \Leftrightarrow \mathrm{Acc}(M^*, \langle x, w\rangle) \geq 1\ \right) \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly. This completes the proof of the theorem.

$\square$

Proof of Theorem 3.4.27 We show that if OptP is randomly closed under proper subtraction, integer division, or span_k, for some $k \geq 2$, then $\forall\text{P} \subseteq \widehat{\text{BP}} \cdot \exists\text{P}$. This together with Theorem 3.4.26 proves the theorem.

As part of the proof of Theorem 3.4.26, we have already shown that if OptP is randomly closed under proper subtraction, then $\forall\text{P} \subseteq \widehat{\text{BP}} \cdot \exists\text{P}$. The proof that integer division is randomly OptP-hard for OptP is quite similar to that of proper subtraction; just use $1 \div (g(x) + 1)$.

The proof that span_k, for all $k \geq 2$, is randomly hard for OptP is similar to the one done for #P. The only difference is that we need the Turing transducer M' constructed in the proof of Theorem 3.4.26. $\square$

3.5 Concluding Remarks

In this chapter we studied *witness reduction*, which has played a crucial role in several recent results in complexity theory. These include Toda's result that $\text{PH} \subseteq \text{BP} \cdot \oplus\text{P}$, the "collapsing" of PH into $\oplus$P with a high probability, Toda and Ogiwara's results which "collapses" PH into various counting classes with a high probability, and hard functions for various function classes studied by Ogiwara and Hemachandra. Ogiwara and Hemachandra's results establish a connection between functions being hard for #P and functions interacting with the class to effect *witness reduction*. In fact, we believe that the ability to achieve some form of *witness reduction* is what makes a function hard for a class of functions. To support our thesis we introduced the notion of randomly hard functions and obtained similar results. In the next chapter, we will apply *witness reduction* to some natural extensions of $\#$P to support our thesis.

CHAPTER IV

Witness Reduction in $\mathcal{Z}\#$P and $\mathcal{Q}\#$P

Some claim I have many defects, others claim I am perfect,

Nobody truly sees, I simply reflect.

— Sanjay Gupta

4.1 Introduction

In this chapter we continue on the *witness reduction* theme introduced in Chapter III. We first consider the issue of why subtraction is a hard function for #P? To this end, we define a class $\mathcal{Z}\#$P, which is similar to #P, except that it is closed under subtraction. Does the closure of $\mathcal{Z}\#$P under subtraction imply that $\mathcal{Z}\#$P is closed under $\mathcal{Z}\#$P? Apparently not, because subtraction in $\mathcal{Z}\#$P does not effect *witness reduction.* However, proper subtraction, as well as integer division, effect *witness reduction* even in $\mathcal{Z}\#$P, and hence turn out to be $\mathcal{Z}\#$P-hard for $\mathcal{Z}\#$P.

Similarly, consider the issue of why division is a hard function for #P and $\mathcal{Z}\#$P? We define a class $\mathcal{Q}\#$P, which is similar to $\mathcal{Z}\#$P and #P, except that it is also closed under division. Again, closure under division apparently does not imply that $\mathcal{Q}\#$P is closed under $\mathcal{Q}\#$P. However, integer division, which effects *witness reduction* in

$\mathcal{Q}\#\mathrm{P}$, turns out to be $\mathcal{Q}\#\mathrm{P}$-hard for $\mathcal{Q}\#\mathrm{P}$. These examples provide more evidence for the thesis that it is the ability of a function to interact with a class of functions to effect *witness reduction* that makes functions hard for a class of functions.

4.2 Definitions and Preliminaries

Some of the following definitions given in Chapter III are repeated here to make comparisons with the new classes $\mathcal{Z}\#\mathrm{P}$ and $\mathcal{Q}\#\mathrm{P}$.

Definition 4.2.1

1. *A function* $f : \Sigma^* \longmapsto \mathcal{N}$ *is in* $\#\mathrm{P}$ *iff there exists a polynomial time bounded nondeterministic machine* M *such that for all* x, $f(x) = \mathrm{Acc}(M, x)$.
2. *A function* $f : \Sigma^* \longmapsto \mathcal{Z}$ *is in* $\mathcal{Z}\#\mathrm{P}$ *iff there exists a polynomial time bounded nondeterministic machine* M *such that for all* x, $f(x) = \mathrm{AR}(M, x)$[1].
3. *A function* $f : \Sigma^* \longmapsto \mathcal{Q}$ *is in* $\mathcal{Q}\#\mathrm{P}$ *iff there exists a pair of polynomial time bounded nondeterministic machines* (M^n, M^d) *such that for all* x, $\mathrm{AR}(M^d, x) \neq 0$ *and* $f(x) = \frac{AR(M^n,x)}{AR(M^d,x)}$.

Definition 4.2.2

1. *A language* $L \in \mathrm{C}_{=}\mathrm{P}$ *iff there exists a function* $f \in \mathrm{PF}$, $f : \Sigma^* \longmapsto \mathcal{N}$, *and* $g \in \#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = f(x)$.
2. *A language* $L \in \mathcal{Z}\mathrm{C}_{=}\mathrm{P}$ *iff there exists a function* $f \in \mathrm{PF}$, $f : \Sigma^* \longmapsto \mathcal{Z}$, *and* $g \in \mathcal{Z}\#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = f(x)$.

[1] This class is same as GapP[12].

3. *A language* $L \in \mathcal{Q}\mathrm{C_{=}P}$ *iff there exists a function* $f \in \mathrm{PF}$, $f : \Sigma^* \longmapsto \mathcal{Q}$, *and* $g \in \mathcal{Q}\#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = f(x)$.

Definition 4.2.3

1. *A language* $L \in \mathrm{UP}$ *iff there exists a function* $g \in \#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = 1$ *and* $x \notin L \Leftrightarrow g(x) = 0$.

2. *A language* $L \in \mathcal{Z}\mathrm{UP}$ *iff there exists a function* $g \in \mathcal{Z}\#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = 1$ *and* $x \notin L \Leftrightarrow g(x) = 0$ [2].

3. *A language* $L \in \mathcal{Q}\mathrm{UP}$ *iff there exists a function* $g \in \mathcal{Q}\#\mathrm{P}$ *such that for all* x, $x \in L \Leftrightarrow g(x) = 1$ *and* $x \notin L \Leftrightarrow g(x) = 0$.

Note that it is possible to redefine any class of languages, originally defined in terms of number of accepting paths of an NTM, under a new domain of interpretation, integers or rational numbers, as above. For example, $\mathcal{Z}\mathrm{NP}$ is the class of languages L such that there exists a function $g \in \mathcal{Z}\#\mathrm{P}$ such that $x \in L \Leftrightarrow g(x) \geq 1$, and $x \notin L \Leftrightarrow g(x) = 0$. However, this may not always yield a new class; $\mathcal{Z}\mathrm{NP}$ is actually same as co-$\mathrm{C_{=}P}$. In fact, the counting hierarchy CH can be viewed as $\mathcal{Z}\mathrm{NP}^{\mathcal{Z}NP^{\cdot^{\cdot^{\cdot}}}}$. Thus, CH is same as PH under the new domain of interpretation.

The following proposition compares the basic closure properties of $\#\mathrm{P}$, $\mathcal{Z}\#\mathrm{P}$ and $\mathcal{Q}\#\mathrm{P}$.

Proposition 4.2.4

1. *[35]* $\#\mathrm{P}$ *is closed under addition and multiplication (on N).*

---

[2] This class is same as SPP[12] and XP[29].

2. *$\mathcal{Z}\#$P is closed under addition, multiplication, and subtraction (on Z).*

3. *$\mathcal{Q}\#$P is closed under addition, multiplication, subtraction, and division (by a non-zero $\mathcal{Q}\#$P function)(on Q).*

Proof.

1. Given f_1 and f_2 in $\#$P, witnessed by NTMs M_1 and M_2, construct NTM M as follows: On input x, M guesses a bit b. If $b = 0$ then it simulates M_1 and if $b = 1$ then it simulates M_2. Clearly, $\mathrm{Acc}(M, x) = \mathrm{Acc}(M_1, x) + \mathrm{Acc}(M_2, x)$. To obtain $f_1 * f_2$, construct M as follows: On input x, M simulates M_1 followed by M_2. It accepts on a computation path iff both the machines accept.

2. The proof that $\mathcal{Z}\#$P is closed under addition is similar to that of $\#$P. To obtain $\mathrm{AR}(M_1, x) * \mathrm{AR}(M_2, x)$, construct M as follows: On input x, M simulates M_1 followed by M_2. M accepts on a computation path iff the computation paths of M_1 and M_2 either both accept or both reject. It is not difficult to see that $\mathrm{AR}(M, x) = \mathrm{AR}(M_1, x) * \mathrm{AR}(M_2, x)$.

   Given an NTM M, construct M^* which accepts iff M rejects. Then, $\mathrm{AR}(M^*, x) = -\mathrm{AR}(M, x)$. This together with the fact that $\mathcal{Z}\#$P is closed under addition shows that it is also closed under subtraction.

3. Let $f_1, f_2 \in \mathcal{Q}\#$P be witnessed by NTM pairs (M_1^n, M_1^d) and (M_2^n, M_2^d) where, for all x, $\mathrm{AR}(M_i^d, x) \neq 0, i = 1, 2$. Then,

   (a) $\frac{AR(M_1^n,x)}{AR(M_1^d,x)} \pm \frac{AR(M_2^n,x)}{AR(M_2^d,x)} = \frac{AR(M_1^n,x)*AR(M_2^d,x) \pm AR(M_2^n,x)*AR(M_1^d,x)}{AR(M_1^d,x)*AR(M_2^d,x)}$.

(b) $\frac{AR(M_1^n,x)}{AR(M_1^d,x)} * \frac{AR(M_2^n,x)}{AR(M_2^d,x)} = \frac{AR(M_1^n,x)*AR(M_2^n,x)}{AR(M_1^d,x)*AR(M_2^d,x)}$.

(c) $\frac{AR(M_1^n,x)}{AR(M_1^d,x)} / \frac{AR(M_2^n,x)}{AR(M_2^d,x)} = \frac{AR(M_1^n,x)*AR(M_2^d,x)}{AR(M_1^d,x)*AR(M_2^n,x)}, \frac{AR(M_2^n,x)}{AR(M_2^d,x)} \neq 0$.

As AR functions are closed under addition, subtraction, and multiplication, in each of the above cases we can construct a pair of NTMs to obtain the corresponding function. Also, note that in each case the denominator NTMs take on non-zero values on all inputs. This shows that $\mathcal{Q}\#\mathrm{P}$ is closed under addition, subtraction, multiplication and division (by a non-zero function).

□

In the next proposition, $\mathrm{P}^{\|\#\mathrm{P}}$ denotes parallel queries to a #P function; that is, new queries do not depend on previous query answers.

Proposition 4.2.5

1. $\#\mathrm{P} \subseteq \mathcal{Z}\#\mathrm{P} \subseteq \mathcal{Q}\#\mathrm{P}$.

2. $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\mathcal{Z}\#\mathrm{P}[1]} = \mathrm{P}^{\mathcal{Q}\#\mathrm{P}[1]} = \mathrm{P}^{\|\#\mathrm{P}}$.

Proof.

1. Let $f \in \#\mathrm{P}$ be witnessed by NTM M. Modify M to obtain M^* as follows: On input x, M^* simulates M on x and if M halts in a reject state then M^* guesses a bit and accepts on 1 and rejects on 0. Then, $\mathrm{AR}(M^*, x) = \mathrm{Acc}(M, x) = f(x)$. Thus, $f \in \#\mathrm{P}$.

Let $g \in \mathcal{Z}\#\mathrm{P}$ be witnessed by NTM M. Then, the pair of NTMs (M, M_1) witness that $g \in \mathcal{Q}\#\mathrm{P}$, where NTM M_1 has exactly one accepting path (and no rejecting paths) on all inputs.

2. $\mathrm{P}^{\#\mathrm{P}[1]} \subseteq \mathrm{P}^{\mathcal{Z}\#\mathrm{P}[1]} \subseteq \mathrm{P}^{\mathcal{Q}\#\mathrm{P}[1]}$ follows directly from part 1. We show that $\mathrm{P}^{\mathcal{Q}\#\mathrm{P}[1]} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$ by encoding 4 queries into one. Let $L \in \mathrm{P}^{\mathcal{Q}\#\mathrm{P}[1]}$ be accepted by TM M which queries a function $g \in \mathcal{Q}\#\mathrm{P}$ witnessed by NTMs (M^n, M^d), each time-bounded by a polynomial p. Let $f(x) = \mathrm{Acc}(M^n, x) + 2^{p(|x|)} * \mathrm{Rej}(M^n, x) + 2^{2p(|x|)} * \mathrm{Acc}(M^d, x) + 2^{3p(|x|)} * \mathrm{Rej}(M^d, x)$. As $\#\mathrm{P}$ is closed under addition and multiplication and $\mathrm{Rej}(M, x) \in \#\mathrm{P}$ (by interchanging the accepting and rejecting states of M), $f \in \#\mathrm{P}$. Construct a machine M^* such that whenever M queries z to the function g, M^* queries z to the function f. M^* then extracts the values of $\mathrm{Acc}(M^n, x), \mathrm{Acc}(M^d, x)$ etc. from the query answer $f(z)$ and computes $g(z)$. Clearly, M^* accepts L.

   Note that a polynomial number of parallel queries can be encoded into one query, as above, giving $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\|\#\mathrm{P}}$.

$\square$

Finally, our proofs will use the following proposition.

Proposition 4.2.6

1. $\mathrm{C}_=\mathrm{P} = \mathcal{Z}\mathrm{C}_=\mathrm{P} = \mathcal{Q}\mathrm{C}_=\mathrm{P}$.

2. $\mathrm{C}_=.\mathcal{Z}\mathrm{UP} \subseteq \mathcal{Z}\mathrm{C}_=\mathrm{P}$.

3. $\mathcal{Q}\mathrm{UP} = \mathrm{C}_{=}\mathrm{P} \cap \text{co-}\mathrm{C}_{=}\mathrm{P}$.

4. Let $L \in \mathcal{Z}\mathrm{C}_{=}\mathrm{P} = \mathcal{Q}\mathrm{C}_{=}\mathrm{P}$. *Then, there exists an NTM* M *and a polynomial* p *such that for all* x, $x \in L \Leftrightarrow \mathrm{AR}(M,x) = 2^{p(|x|)}$ *and* $x \notin L \Leftrightarrow 0 \leq \mathrm{AR}(M,x) < 2^{p(|x|)}$.

Proof.

1. By part 1 of Proposition 4.2.5, we have $\mathrm{C}_{=}\mathrm{P} \subseteq \mathcal{Z}\mathrm{C}_{=}\mathrm{P} \subseteq \mathcal{Q}\mathrm{C}_{=}\mathrm{P}$. We show that $\mathcal{Q}\mathrm{C}_{=}\mathrm{P} \subseteq \mathrm{C}_{=}\mathrm{P}$. As $\mathcal{Q}\#\mathrm{P}$ is closed under subtraction, $L \in \mathcal{Q}\mathrm{C}_{=}\mathrm{P}$ iff there exists a function $g \in \mathcal{Q}\#\mathrm{P}$ such that for all x, $x \in L \Leftrightarrow g(x) = 0$. Let $g \in \mathcal{Q}\#\mathrm{P}$ be witnessed by the pair of NTMs (M^n, M^d). Then, $g(x) = 0 \Leftrightarrow \mathrm{AR}(M^n, x) = 0$. Let $p(n)$ be the running time bound on M^n.

   Construct and NTM M that guesses exactly $p(n)+1$ bits as follows: On input x, M simulates M^n. When a computation of M^n, say y, terminates, then M extends y into a full subtree of computations, each computation of length $p(|x|)+1$. Further, if computation y accepts, then the subtree of computations extending y contributes a net of $+2$ to $\mathrm{AR}(M,x)$, and if y rejects, the subtree contributes a net of -2 to $\mathrm{AR}(M,x)$. This can be easily achieved as follows: Consider a computation path of M^n, say y, $|y| \leq p(|x|)$. Then, M guesses z, $|z| = p(|x|) - |y|$, and a bit b. It accepts on computation path $y0^{p(|x|)-|y|}0$ and $y0^{p(|x|)-|y|}1$, iff M^n accepts. On guesses, $z \neq 0^{p(|x|)-|y|}$, it accepts on $yz0$ and rejects on $yz1$. It follows that $\mathrm{AR}(M,x) = 2 * \mathrm{AR}(M^n, x)$. As M guesses exactly $2^{p(|x|)}+1$ bits, we have, $x \in L \Leftrightarrow 2 * \mathrm{AR}(M^n,x) = 0 \Leftrightarrow \mathrm{Acc}(M,x) = 2^{p(|x|)}$. Thus, $L \in \mathrm{C}_{=}\mathrm{P}$.

2. Let $L \in \mathrm{C}_{=}.\mathcal{Z}\mathrm{UP}$. Then there exists a set $A \in \mathcal{Z}\mathrm{UP}$, and $f \in \mathrm{PF}$, and a polynomial p such that $x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|), \langle x,y\rangle \in A\}\| = f(x)$. Let A be accepted by an NTM M_A. Then, $\langle x,y\rangle \in A \Leftrightarrow \mathrm{AR}(M_A, \langle x,y\rangle) = 1$ and $\langle x,y\rangle \notin A \Leftrightarrow AR(M_A, \langle x,y\rangle) = 0$. Construct NTM M which, on input x, guesses $y, |y| = p(|x|)$ and simulates M_A on $\langle x,y\rangle$. It is easy to see that, $\mathrm{AR}(M,x) = f(x) \Leftrightarrow x \in L$.

3. Clearly, $\mathcal{Q}\mathrm{UP} \subseteq \mathcal{Q}\mathrm{C}_{=}\mathrm{P} \cap \text{co-}\mathcal{Q}\mathrm{C}_{=}\mathrm{P} = \mathcal{Z}\mathrm{C}_{=}\mathrm{P} \cap \text{co-}\mathcal{Z}\mathrm{C}_{=}\mathrm{P}$. Consider $L \in \mathcal{Z}\mathrm{C}_{=}\mathrm{P} \cap \text{co-}\mathcal{Z}\mathrm{C}_{=}\mathrm{P}$. Then there exist M_1 and M_2, witnessing $L \in \mathcal{Z}\mathrm{C}_{=}\mathrm{P}$ and $\overline{L} \in \mathcal{Z}\mathrm{C}_{=}\mathrm{P}$ respectively, such that

$$\begin{array}{c} x \in L \Leftrightarrow \mathrm{AR}(M_1,x) = 0 \\ x \notin L \Leftrightarrow \mathrm{AR}(M_2,x) = 0 \end{array}$$

Let $f(x) = \frac{(AR(M_2,x))^2}{(AR(M_1,x))^2+(AR(M_2,x))^2}$. Clearly, for all x, $(\mathrm{AR}(M_1,x))^2+(\mathrm{AR}(M_2,x))^2 \neq 0$. Thus, $f \in \mathcal{Q}\#\mathrm{P}$ and

$$\begin{array}{c} f(x) = 0 \Leftrightarrow \mathrm{AR}(M_2,x) = 0 \Leftrightarrow x \notin L \\ f(x) = 1 \Leftrightarrow \mathrm{AR}(M_1,x) = 0 \Leftrightarrow x \in L \end{array}$$

Hence, f witnesses that $L \in \mathcal{Q}\mathrm{UP}$.

4. Let $L \in \mathcal{Z}\mathrm{C}_{=}\mathrm{P}$ be witnessed by an NTM M, time-bounded by a polynomial q, and $f \in PF$. Let $h(x) = AR(M,x) \in \mathcal{Z}\#\mathrm{P}$. As $\mathcal{Z}\#\mathrm{P}$ is closed under subtraction and multiplication, let $h^*(x) = -(AR(M,x) - f(x))^2 + 2^{2q(|x|)} \in \mathcal{Z}\#\mathrm{P}$ be witnessed by NTM M^*. The NTM M^* has the required properties.

□

4.3 Witness Reduction

Now we are ready to prove our results for the new function classes $\mathcal{Z}\#\mathrm{P}$ and $\mathcal{Q}\#\mathrm{P}$.

4.3.1 Witness reduction in $\mathcal{Z}\#\mathrm{P}$

Theorem 4.3.1 (Witness Reduction Implies Collapsing) $\mathrm{C}_=\mathrm{P} \subseteq \mathcal{Z}\mathrm{UP} \Rightarrow CH = \mathcal{Z}\mathrm{UP}$.

Theorem 4.3.2 (Witness Reduction iff Closure) $\mathrm{C}_=\mathrm{P} \subseteq \mathcal{Z}\mathrm{UP} \Leftrightarrow \mathcal{Z}\#\mathrm{P}$ *is closed under* $\mathcal{Z}\#\mathrm{P}$.

Theorem 4.3.3 (Hard Functions for $\mathcal{Z}\#\mathrm{P}$) *Proper subtraction($\ominus$), integer division($\div$) and span$_k$, for all $k \geq 2$, are $\mathcal{Z}\#\mathrm{P}$-hard for $\mathcal{Z}\#\mathrm{P}$.*

Corollary 4.3.4 *(to Theorem 4.3.3)$\mathcal{Z}\#\mathrm{P}$ is closed under $\mathcal{Z}\#\mathrm{P}$ $\Leftrightarrow$ $\mathcal{Z}\#\mathrm{P}$ is closed under* PF.

Proof of Theorem 4.3.1 $\mathrm{C}_=\mathrm{P} \subseteq \mathcal{Z}\mathrm{UP}$ implies that $\mathrm{C}_= \cdot \mathrm{C}_= \cdot \mathrm{P} \subseteq \mathrm{C}_= \cdot \mathcal{Z}\mathrm{UP} \subseteq \mathrm{C}_=\mathrm{P} \subseteq \mathcal{Z}\mathrm{UP}$, using Proposition 4.2.6. Iterating this process it is easy to prove that $\underbrace{\mathrm{C}_= \cdots \mathrm{C}_= \cdot \mathrm{C}_=}_{k}\mathrm{P} \subseteq \mathcal{Z}\mathrm{UP}$, for all $k \geq 1$. Thus, $\mathrm{CH} \subseteq \mathcal{Z}\mathrm{UP} \subseteq \mathcal{Z}\mathrm{C}_=\mathrm{P} = \mathrm{C}_=\mathrm{P} \subseteq \mathrm{CH}$. $\square$

Proof of Theorem 4.3.2 [$\Longleftarrow$] Let $L \in \mathcal{Z}\mathrm{C}_=\mathrm{P}$. Then by Proposition 4.2.6 there exists an NTM M and a polynomial p such that

$$\begin{aligned} x \in L &\Leftrightarrow AR(M,x) = 2^{p(|x|)} \\ x \notin L &\Leftrightarrow 0 \leq AR(M,x) < 2^{p(|x|)} \end{aligned}$$

Let $g_1(x) = AR(M,x)$ and $g_2(x) = 2^{p(|x|)}$. Under the hypothesis, since integer division is a $\mathcal{Z}\#$P-function, there exists $h \in \mathcal{Z}\#$P such that $h(x) = \lfloor g_1(x)/g_2(x) \rfloor$. Thus,

$$x \in L \Leftrightarrow h(x) = 1$$
$$x \notin L \Leftrightarrow h(x) = 0.$$

Hence, $h \in \mathcal{Z}\#$P witnesses that $L \in \mathcal{Z}$UP.

[$\Longrightarrow$] Let $f \in \mathcal{Z}\#$P be a k-ary function witnessed by NTM M_f. Let $g_1, g_2, \ldots, g_k \in \mathcal{Z}\#$P be witnessed by NTMs $M_1, \ldots, M_k$, each polynomial time bounded by $p_1, p_2, \ldots, p_k$, respectively. Define the following languages as graphs of the above functions.

$$L_i = \{\langle x, y\rangle | y = g_i(x) = AR(M_i, x)\}, 1 \leq i \leq k$$

Claim 1: $L_i \in \mathcal{Z}\mathrm{C}_=\mathrm{P} = \mathrm{C}_=\mathrm{P}$

Proof: The following NTM M_i^* accepts L_i.

1. Input $\langle x, y\rangle$. We assume that $-2^{p_i(|x|)} \leq y \leq 2^{p_i(|x|)}$.

2. Guess $b \in \{0, 1\}$.

3. If $b = 0$ then M_i^* simulates M_i on x and accepts iff M_i accepts.

4. If $b = 1$ then M_i^* nondeterministically branches into a tree with (number of accepts - number of rejects) equal to -y.

Note that $AR(M_i^*, \langle x, y\rangle) = 0 \Leftrightarrow y = AR(M_i, x) \Leftrightarrow \langle x, y\rangle \in L_i$. Hence, $L_i \in \mathcal{Z}\mathrm{C}_=\mathrm{P}$. $\square$

Under our hypothesis, there exists an NTM $M_i^{**}, 1 \le i \le k$ such that

$$\langle x, y\rangle \in L_i \Leftrightarrow AR(M_i^{**}, \langle x, y\rangle) = 1$$

$$\langle x, y\rangle \notin L_i \Leftrightarrow AR(M_i^{**}, \langle x, y\rangle) = 0$$

Now we construct NTM M which witnesses that $f(g_1, g_2, \ldots, g_k) \in \mathcal{Z}\#\mathrm{P}$.

NTM M

1. Input x.

2. Guess $y_i, -2^{p_i(|x|)} \le y_i \le 2^{p_i(|x|)}, 1 \le i \le k$.

3. Run M_i^{**} on $\langle x, y_i\rangle, 1 \le i \le k$, sequentially. This follows some computations $C_1, C_2, \ldots, C_k$, where C_i is a computation of M_i^{**} on $\langle x, y_i\rangle$.

4. Run M_f on $\langle y_1, y_2, \ldots, y_k\rangle$ following a final computation C_{k+1}.

5. The computation $C_1, C_2, \ldots, C_{k+1}$ of M accepts iff an even number of $C_1, C_2, \ldots, C_{k+1}$ reject.

It is not difficult to see that

$$AR(M, x) = \sum_{y_1, y_2, \cdots, y_k} [AR(M_1^{**}, \langle x, y_1\rangle) \ldots AR(M_k^{**}, \langle x, y_k\rangle) AR(M_f, \langle y_1, y_2, \ldots, y_k\rangle)]$$

where the sum is taken over all the guesses $y_1, y_2, \ldots, y_k$. This is because each C_i that accepts contributes a $+1$ in $AR(M_i^{**}, \langle x, y_i\rangle)$ and each C_i that rejects contributes a -1 in $AR(M_i^{**}, \langle x, y_i\rangle)$.

By the definition of M_i^{**}, the above product is 0 if any one guess $y_i \neq g_i(x)$. Also, $AR(M_i^{**}, \langle x, y\rangle) = 1$ if $y = g_i(x)$. Hence,

$$AR(M, x) = AR(M_f, \langle g_1(x), g_2(x), \ldots, g_k(x)\rangle)$$

Also, the running time of NTM M is bounded by some polynomial. $\square$

Proof of Theorem 4.3.3 We show that closure of $\mathcal{Z}\#\mathrm{P}$ under proper subtraction, integer division and span implies that $\mathrm{C}_=\mathrm{P} \subseteq \mathcal{Z}\mathrm{UP}$. Then, by Theorem 4.3.2, we are done.

Let $L \in \mathcal{Z}\mathrm{C}_=\mathrm{P} = \mathrm{C}_=\mathrm{P}$. Then, there exists a function $g \in \mathcal{Z}\#\mathrm{P}$ and a polynomial p such that $x \in L \Leftrightarrow g(x) = 2^{p(|x|)}$ and $x \notin L \Leftrightarrow 0 \leq g(x) < 2^{p(|x|)}$. Let $g_2(x) = 2^{p(|x|)} - 1$. Then, by the hypothesis, $h = g \ominus g_2 \in \mathcal{Z}\#\mathrm{P}$ and $x \in L \Leftrightarrow h(x) = 1$ and $x \notin L \Leftrightarrow h(x) = 0$. Thus, $L \in \mathcal{Z}\mathrm{UP}$. This shows that proper subtraction is hard for $\mathcal{Z}\#\mathrm{P}$.

Let $g_2(x) = 2^{p(|x|)}$. Then, under the hypothesis, $h = g \div g_2 \in \mathcal{Z}\#\mathrm{P}$ and $x \in L \Leftrightarrow h(x) = 1$ and $x \notin L \Leftrightarrow h(x) = 0$. Thus, $L \in \mathcal{Z}\mathrm{UP}$. This shows that integer division is hard for $\mathcal{Z}\#\mathrm{P}$.

Given $L \in \mathcal{Z}\mathrm{C}_=\mathrm{P}$, as $\mathcal{Z}\#\mathrm{P}$ is closed under subtraction, there exists a function $g \in \mathcal{Z}\#\mathrm{P}$ such that $x \in L \Leftrightarrow g(x) = 0$. Let $g_i(x) = 2 \cdot g(x)$, for $2 \leq i \leq k$. Then, under the hypothesis, there exists $h \in \mathcal{Z}\#\mathrm{P}$, such that $h(x) = span_k(g, g_2, \ldots, g_k)$. Hence, $x \in L \Leftrightarrow h(x) = 1$ and $x \notin L \Leftrightarrow h(x) = 2$. Again, as $\mathcal{Z}\#\mathrm{P}$ is closed under subtraction, $h^*(x) = 2 - h(x) \in \mathcal{Z}\#\mathrm{P}$. Now, $x \in L \Leftrightarrow h^*(x) = 1$ and $x \notin L \Leftrightarrow h^*(x) = 0$ or, $L \in \mathcal{Z}\mathrm{UP}$. $\square$

4.3.2 Witness Reduction in $\mathcal{Q}\#$P

In this section we need an additional result on quantifiers. Note that $\forall \cdot \mathrm{C}_=\mathrm{P} = \mathrm{C}_=\mathrm{P}$. This is because, by Proposition 3.2.7, L is in $\mathrm{C}_=\mathrm{P}$ iff there exists a nondeterministic polynomial time bounded Turing machine M and a polynomial p such that for every $x \in \Sigma^*$,

1. $\mathrm{Acc}(M, x) + \mathrm{Rej}(M, x) = 2^{p(|x|)+1}$.

2. $x \in L$ iff $\mathrm{Acc}(M, x) = 2^{p(|x|)}$.

3. $x \notin L$ iff $0 < \mathrm{Acc}(M, x) < 2^{p(|x|)}$.

Thus, the two levels of quantification in $\forall \cdot \mathrm{C}_=\mathrm{P}$ can be viewed as one, preserving the $\mathrm{C}_=\mathrm{P}$ acceptance criteria, but with a different exact threshold function. Similarly, we have $\exists \cdot \text{co-}\mathrm{C}_=\mathrm{P} = \text{co-}\mathrm{C}_=\mathrm{P}$.

This yields the following interesting lemma.

Lemma 4.3.5 $\mathrm{CP} = \mathrm{C}_=\mathrm{P}$ *iff* $\mathrm{C}_=\mathrm{P}$ *is closed under complement.*

Proof. The left to right implication is obvious as $\mathrm{CP} = \mathrm{PP}$ is closed under complement. To see the other implication, note that $\exists \cdot \mathrm{CP} = \exists \cdot \mathrm{C}_=\mathrm{P}$[47]. Now, under our hypothesis that $\mathrm{C}_=\mathrm{P}$ is closed under complement, we have $\mathrm{C}_=\mathrm{P} \subseteq \mathrm{CP} \subseteq \exists \cdot \mathrm{CP} = \exists \cdot \mathrm{C}_=\mathrm{P} = \exists \cdot \text{co-}\mathrm{C}_=\mathrm{P} = \text{co-}\mathrm{C}_=\mathrm{P} = \mathrm{C}_=\mathrm{P}$. $\square$

Combining this with Proposition 4.2.6 we have the following corollary.

Corollary 4.3.6 $\mathcal{Q}\mathrm{C_=P} \subseteq \mathcal{Q}\mathrm{UP} \Leftrightarrow \mathrm{C_=P} \subseteq \mathrm{C_=P} \cap \text{co-}\mathrm{C_=P} \Leftrightarrow \mathrm{C_=P}$ *is closed under complement* $\Leftrightarrow \mathrm{CP} = \mathrm{C_=P}$.

Thus, in the proofs of the following theorems it is sufficient to show $\mathcal{Q}\mathrm{C_=P} \subseteq \mathcal{Q}\mathrm{UP} \Leftrightarrow \mathcal{Q}\#\mathrm{P}$ is closed under $\mathcal{Q}\#\mathrm{P}$.

Theorem 4.3.7 (Witness Reduction Implies Collapsing) $\mathrm{CP} = \mathrm{C_=P} \Rightarrow \underbrace{\exists \cdot \forall \cdots}_{k>0} \mathrm{CP} \subseteq \mathrm{C_=P}$.

Theorem 4.3.8 (Witness Reduction iff Closure) $\mathrm{CP} = \mathrm{C_=P} \Leftrightarrow \mathrm{C_=P}$ *is closed under complement* $\Leftrightarrow \mathcal{Q}\#\mathrm{P}$ *is closed under* $\mathcal{Q}\#\mathrm{P}$.

Theorem 4.3.9 (Hard Functions for $\mathcal{Q}\#\mathrm{P}$) *Proper subtraction($\ominus$), integer division($\div$) and span$_k$, for all $k \geq 2$, are $\mathcal{Q}\#\mathrm{P}$-hard for $\mathcal{Q}\#\mathrm{P}$.*

Corollary 4.3.10 *(To Theorem 4.3.9)$\mathcal{Q}\#\mathrm{P}$ is closed under $\mathcal{Q}\#\mathrm{P} \Leftrightarrow \mathcal{Q}\#\mathrm{P}$ is closed under* PF.

Proof of Theorem 4.3.7 As $\mathrm{C_=P}$ is closed under complement $\Leftrightarrow \mathrm{CP} = \mathrm{C_=P}$, we have $\exists \cdot \mathrm{CP} \subseteq \exists \cdot \text{co-}\mathrm{C_=P} = \text{co-}\mathrm{C_=P} = \mathrm{C_=P}$ and $\forall \cdot \mathrm{CP} \subseteq \forall \cdot \mathrm{C_=P} = \mathrm{C_=P}$. Iterating this process proves the theorem. $\square$

Proof of Theorem 4.3.8 [$\Longleftarrow$] Let $L \in \mathcal{Q}\mathrm{C_=P} = \mathcal{Z}\mathrm{C_=P}$. Then, by part 4 of Proposition 4.2.6, there exists an NTM M and a polynomial p such that

$$\begin{aligned} x \in L &\Leftrightarrow AR(M,x) = 2^{p(|x|)} \\ x \notin L &\Leftrightarrow 0 \leq AR(M,x) < 2^{p(|x|)} \end{aligned}$$

Let $g_1(x) = AR(M,x)$ and $g_2(x) = 2^{p(|x|)}$. Under the hypothesis, since integer division is a $\mathcal{Q}\#\mathrm{P}$-function, there exists $h \in \mathcal{Q}\#\mathrm{P}$ such that $h(x) = \lfloor g_1(x)/g_2(x) \rfloor$. Thus,

$$\begin{aligned} x \in L &\Leftrightarrow h(x) = 1 \\ x \notin L &\Leftrightarrow h(x) = 0. \end{aligned}$$

Hence, h witnesses that $L \in \mathcal{Q}\mathrm{UP}$.

[$\Longrightarrow$] Let $f \in \mathcal{Q}\#\mathrm{P}$ be a k-ary function, witnessed by an NTM pair (M_f^n, M_f^d). Let $g_1, g_2, \ldots, g_k \in \mathcal{Q}\#\mathrm{P}$ be witnessed by NTM pairs $(M_1^n, M_1^d), (M_2^n, M_2^d), \ldots, (M_k^n, M_k^d)$, respectively, each polynomial time bounded. Define the following languages as graphs of the above functions:

$$L_i = \{\langle x, y\rangle | y = AR(M_i^n, x)\}, 1 \leq i \leq k$$

Then, as shown before in Theorem 4.3.2(Claim 1), each $L_i \in \mathcal{Z}\mathrm{C}_=\mathrm{P}$.

Under our hypothesis, $\mathcal{Z}\mathrm{C}_=\mathrm{P} = \mathcal{Q}\mathrm{C}_=\mathrm{P} \subseteq \mathcal{Q}\mathrm{UP}$. Thus, there exist NTM pairs $(\hat{M}_i^n, \hat{M}_i^d)$, $1 \leq i \leq k$, each polynomial time bounded by $(p_1^n, p_1^d), (p_2^n, p_2^d), \ldots, (p_k^n, p_k^d)$, such that,

$$\begin{aligned} \langle x, y\rangle \in L_i &\Leftrightarrow \frac{AR(\hat{M}_i^n, \langle x,y\rangle)}{AR(\hat{M}_i^d, \langle x,y\rangle)} = 1 \\ \langle x, y\rangle \notin L_i &\Leftrightarrow \frac{AR(\hat{M}_i^n, \langle x,y\rangle)}{AR(\hat{M}_i^d, \langle x,y\rangle)} = 0 \Leftrightarrow AR(\hat{M}_i^n, \langle x, y\rangle) = 0 \end{aligned}$$

Now we construct NTM (M^n, M^d) which witness that $f(g_1, g_2, \ldots, g_k) \in \mathcal{Q}\#\mathrm{P}$.

NTM M^n [and M^d]

1. Input x.

2. Guess $y_i = \frac{n_i}{d_i}, -2^{p_i^n(|x|)} \leq n_i \leq 2^{p_i^n(|x|)}, 1 \leq d_i \leq 2^{p_i^d(|x|)}, 1 \leq i \leq k$. (Note that it is possible to guess multiple y_i). If n_i and d_i are not relatively prime for some i, then guess $b \in \{0, 1\}$ and accept on $b = 1$ and reject on $b = 0$.

3. Run $\hat{M}_i^n$ on $\langle x, y_i \rangle, 1 \le i \le k$, sequentially. This follows some computations $C_1, C_2, \ldots, C_k$, where C_i is a computation of $\hat{M}_i^n$ on $\langle x, y_i \rangle$.

4. Run $M_f^n[M_f^d]$ on $\langle y_1, y_2, \ldots, y_k \rangle$ following a final computation C_{k+1}.

5. The computation $C_1, C_2, \ldots, C_k, C_{k+1}$ of $M^n[M^d]$ accepts iff an even number of $C_1, C_2, \ldots, C_k, C_{k+1}$ reject.

It is not difficult to see that

$$AR(M^n, x) = \sum_{y_1, y_2, \ldots, y_k} [AR(\hat{M}_1^n, \langle x, y_1 \rangle) \cdots AR(\hat{M}_k^n, \langle x, y_k \rangle) \cdot AR(M_f^n, \langle y_1, y_2, \ldots, y_k \rangle)],$$

and,

$$AR(M^d, x) = \sum_{y_1, y_2, \ldots, y_k} [AR(\hat{M}_1^n, \langle x, y_1 \rangle) \cdots AR(\hat{M}_k^n, \langle x, y_k \rangle) \cdot AR(M_f^d, \langle y_1, y_2, \ldots, y_k \rangle)],$$

where the sum is taken over all the guesses $y_1, y_2, \ldots, y_k$. This is because each C_i that accepts contributes a $+1$ in $AR(\hat{M}_i^n, \langle x, y \rangle)$ and each C_i that rejects contributes a -1 in $AR(\hat{M}_i^n, \langle x, y \rangle)$ and similarly for $\hat{M}_i^d$. Multiple guesses of y_i's do not contribute to the value of $AR(M^n, x)$ or $AR(M^d, x)$.

By the definition of $\hat{M}_i^n$, the above product is 0 if any one guess $y_i \ne g_i(x)$. Hence,

$$AR(M^n, x) = AR(\hat{M}_1^n, \langle x, y_1 \rangle) \cdot AR(\hat{M}_2^n, \langle x, y_2 \rangle) \cdots AR(\hat{M}_k^n, \langle x, y_k \rangle) \cdot AR(M_f^n, \langle y_1, y_2, \ldots, y_k \rangle),$$

and,

$$AR(M^d, x) = AR(\hat{M}_1^n, \langle x, y_1 \rangle) \cdot AR(\hat{M}_2^n, \langle x, y_2 \rangle) \cdots AR(\hat{M}_k^n, \langle x, y_k \rangle) \cdot AR(M_f^d, \langle y_1, y_2, \ldots, y_k \rangle),$$

with the correct guesses $y_i = g_i(x)$ and $AR(\hat{M}_i^n, \langle x, y_i \rangle) \ne 0$, for $1 \le i \le k$. Thus, $\frac{AR(M^n, x)}{AR(M^d, x)} = \frac{AR(M_f^n, \langle y_1, y_2, \ldots, y_k \rangle)}{AR(M_f^d, \langle y_1, y_2, \ldots, y_k \rangle)} = f(g_1(x), \ldots, g_k(x))$. Also, the running time of NTM

M is bounded by some polynomial. $\square$

Proof of Theorem 4.3.9 We show that closure of $\mathcal{Q}\#$P under proper subtraction, integer division and span$_k$, for $k \geq 2$ implies that $\mathcal{Q}$C$_=$P $\subseteq \mathcal{Q}$UP. Hence, by Theorem 4.3.8, we are done.

The proof for integer division was done as part of Theorem 4.3.8. The proof for proper subtraction is very similar, take $g_2(x) = 2^{p(|x|)} - 1$.

Given $L \in \mathcal{Q}$C$_=$P $= \mathcal{Z}$C$_=$P, as $\mathcal{Z}\#$P is closed under subtraction, there exists a function $g \in \mathcal{Z}\#$P such that $x \in L \Leftrightarrow g(x) = 0$. Let $g_i(x) = 2 \cdot g(x)$, for $2 \leq i \leq k$. Then, under the hypothesis, there exists $h \in \mathcal{Q}\#$P, such that $h(x) = span_k(g(x), g_2(x), \ldots, g_k(x))$. Hence, $x \in L \Leftrightarrow h(x) = 1$ and $x \notin L \Leftrightarrow h(x) = 2$. Again, as $\mathcal{Q}\#$P is closed under subtraction, $h^*(x) = 2 - h(x) \in \mathcal{Q}\#$P. Now, $x \in L \Leftrightarrow h^*(x) = 1$ and $x \notin L \Leftrightarrow h^*(x) = 0$ or, $L \in \mathcal{Q}$UP. $\square$

4.4 Random Witness Reduction in $\mathcal{Z}\#$P and $\mathcal{Q}\#$P

In the previous sections we studied *random witness reduction* in #P and then *witness reduction* in $\mathcal{Z}\#$P and $\mathcal{Q}\#$P. The concept of random witness reduction in $\mathcal{Z}\#$P and $\mathcal{Q}\#$P is analogous to that of random witness reduction in #P. For example, we can talk about random closure of these function classes or, randomly hard functions for $\mathcal{Z}\#$P and $\mathcal{Q}\#$P. Thus, the general definitions given in Section 3.2 can be applied to these new function classes.

Using the techniques of Chapter III and Section 4.3, it is easy to see the results in this section. Hence, we just state the results without giving proofs. We obtain corresponding collapses and closures with high probability. This is a weaker form of witness reduction — hence it produces weaker collapses.

4.4.1 Random Witness Reduction in $\mathcal{Z}\#\mathrm{P}$

To give proper characterization of the closure properties of $\mathcal{Z}\#\mathrm{P}$, we define the following class, which is similar to Pseudo(UP), under the witness function AR.

Definition 4.4.1 *$L \in \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP})$ iff for each polynomial e there exists a polynomial b and a polynomial time bounded machine M such that*

$$Prob\begin{pmatrix} x \in L \Rightarrow \mathrm{AR}(M, \langle x, w\rangle) = 1 \\ x \notin L \Rightarrow \mathrm{AR}(M, \langle x, w\rangle) = 0 \end{pmatrix} \geq 1 - \frac{1}{2^{e(|x|)}}$$

where $w \in \{0,1\}^{b(|x|)}$ is chosen uniformly.

The proof of the following proposition is similar to that of Proposition 3.4.9.

Proposition 4.4.2 $\mathcal{Z}\mathrm{UP} \subseteq \widehat{\mathrm{BP}} \cdot \mathcal{Z}\mathrm{UP} \subseteq \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \subseteq \mathrm{Almost}(\mathcal{Z}\mathrm{UP})$

It is easy to see that $\widehat{\mathrm{BP}} \cdot \mathcal{Z}\mathrm{UP} \subseteq \mathrm{PP}$; however, it is not clear if $\mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \subseteq \mathrm{PP}$. If this were the case then, $\mathrm{PH} \subseteq \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \subseteq \mathrm{PP}$.

The following Propositions are similar to Propositions 3.4.10 and 3.4.11.

Proposition 4.4.3

1. $\widehat{\mathrm{BP}} \cdot \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) = \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP})$.

2. $\mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \cap \widehat{\mathrm{BP}} \cdot \text{co-}\mathrm{C_{=}P}$.

Proposition 4.4.4

1. $\mathrm{C} \cdot \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{CP}$.

2. $\mathrm{C}_{=} \cdot \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C}_{=}\mathrm{P}$.

Using the above results and the techniques of the Sections 3.4 and 4.3, it is easy to prove the following results.

Theorem 4.4.5 (Random Witness Reduction Implies Collapsing) $\mathrm{C}_{=}\mathrm{P} \subseteq \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \Rightarrow CH = \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP})$.

Theorem 4.4.6 (Random Witness Reduction iff Random Closure) $\mathrm{C}_{=}\mathrm{P} \subseteq \mathrm{Pseudo}(\mathcal{Z}\mathrm{UP}) \Leftrightarrow \mathcal{Z}\#\mathrm{P}$ *is randomly closed under* $\mathcal{Z}\#\mathrm{P}$.

Theorem 4.4.7 (Randomly Hard Functions for $\mathcal{Z}\#\mathrm{P}$) *Proper subtraction($\ominus$), integer division($\div$) and span$_k$, for all $k \geq 2$, are randomly $\mathcal{Z}\#\mathrm{P}$-hard for $\mathcal{Z}\#\mathrm{P}$.*

Corollary 4.4.8 $\mathcal{Z}\#\mathrm{P}$ *is randomly closed under* $\mathcal{Z}\#\mathrm{P} \Leftrightarrow \mathcal{Z}\#\mathrm{P}$ *is randomly closed under* PF.

4.4.2 Random witness reduction in $\mathcal{Q}\#\mathrm{P}$

To give proper characterization of the closure properties of $\mathcal{Q}\#\mathrm{P}$, we need the following class, which is similar to $\mathrm{Pseudo}(\mathcal{Z}\mathrm{UP})$, under the witness function "ratio of two AR functions".

Definition 4.4.9 $L \in \mathrm{Pseudo}(\mathcal{Q}\mathrm{UP})$ *iff for each polynomial e there exists a polynomial b and a function $f \in \mathcal{Q}\#\mathrm{P}$ such that*

$$Prob\begin{pmatrix} x \in L \Rightarrow f(\langle x, w\rangle) = 1 \\ x \notin L \Rightarrow f(\langle x, w\rangle) = 0 \end{pmatrix} \geq 1 - \frac{1}{2^{e(|x|)}},$$

where $w \in \{0,1\}^{b(|x|)}$ *is chosen uniformly.*

Using Proposition 4.2.6 and the definition of Pseudo($\mathcal{Q}$UP), it is easy to prove the following results.

Proposition 4.4.10 Pseudo($\mathcal{Q}$UP) $= \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \cap \widehat{\mathrm{BP}} \cdot \text{co-}\mathrm{C_{=}P}$.

Hence, co-$\mathrm{C_{=}P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \Leftrightarrow \mathrm{C_{=}P} \subseteq$ Pseudo($\mathcal{Q}$UP). Using this, the proof technique of Lemma 4.3.5, and the properties of $\widehat{\mathrm{BP}}$ operator, it is easy to see the following proposition.

Proposition 4.4.11 $\mathrm{CP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \Leftrightarrow \text{co-}\mathrm{C_{=}P} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P}$.

Now, it is easy to prove the following theorems.

Theorem 4.4.12 (Random Witness Reduction Implies Collapsing) $\mathrm{CP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \Rightarrow \underbrace{\exists \cdot \forall \cdots \cdot \mathrm{CP}}_{k>0} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P}$.

Theorem 4.4.13 (Random Witness Reduction iff Random Closure) $\mathrm{CP} \subseteq \widehat{\mathrm{BP}} \cdot \mathrm{C_{=}P} \Leftrightarrow \mathcal{Q}\#\mathrm{P}$ *is randomly closed under* $\mathcal{Q}\#\mathrm{P}$.

Theorem 4.4.14 (Randomly Hard Functions for $\mathcal{Q}\#\mathrm{P}$) *Proper subtraction($\ominus$), integer division($\div$) and span$_k$, for all $k \geq 2$, are randomly $\mathcal{Q}\#\mathrm{P}$-hard for $\mathcal{Q}\#\mathrm{P}$.*

Corollary 4.4.15 $\mathcal{Q}\#\mathrm{P}$ *is randomly closed under* $\mathcal{Q}\#\mathrm{P} \Leftrightarrow \mathcal{Q}\#\mathrm{P}$ *is randomly closed under* PF.

4.5 Concluding Remarks

One of the motivations for defining new domains($\mathcal{Z}$, $\mathcal{Q}$) for interpretation of functions computed by NTMs was to study the reasons for existence of hard functions. However, as stated before, any class that is originally defined in terms of number of accepting paths of a NTM can be redefined, under a new interpretation. We believe that this is of independent interest. It remains to be seen if the enrichment of nondeterministic complexity classes, provided by these new domains, will bring us any closer to the goals of complexity theory.

CHAPTER V

Isolating an Odd Number of Elements

I am a great believer in luck and the harder I work the more I have it.

— Stephan Leacock

5.1 Introduction

In Chapter III, we showed how Valiant and Vazirani's theorem [48] can be used to achieve *random witness reduction.* They constructed a random reduction from a non-empty set to a set with cardinality exactly 1. Given a set S of n-bit vectors, the reduction constructs $n+1$ subsets such that if $S \neq \emptyset$, then one of the $n+1$ subsets will have exactly one element with probability greater than $\frac{1}{4}$. Of course, if $S = \emptyset$, then all the $n+1$ subsets remain empty. Hence, if we pick one of the $n+1$ subsets uniformly, we have a random reduction from a non-empty set to a set with cardinality 1, with probability at least $\frac{1}{4(n+1)}$.

This result has been extensively applied in complexity theory to show containment of the polynomial-time hierarchy in various counting classes[44, 46, 43], and to show that languages accepted by various constant depth circuits can also be accepted by depth-2 threshold circuits of superpolynomial size[1, 43, 51, 6]. In several of these complexity theory applications, it is sufficient to have a subset which has an odd number of elements, not necessarily exactly one element. In this chapter we construct a random reduction from an arbitrary non-empty set to a set with an odd number of elements with a probability of success at least $\frac{3}{16}$. Thus, the probability of success, if we are only interested in isolating an odd number of elements, is independent of n. More precisely, given any set S of n-bit vectors, we can construct *one* subset such that if $S \neq \emptyset$, then the constructed subset will have an odd number of elements with probability greater than $\frac{3}{16}$.

Isolating an odd number of elements has appeared implicitly in the literature in several different contexts. For example, the construction of linear codes can be modified to isolate an odd number of elements[24, 43, 36]. Naor and Naor[27] implicitly isolate an odd number of elements, with probability $\frac{1}{8}$, in their construction of "small-bias probability spaces". In comparing these constructions with ours, our construction, given in the proof of Theorem 5.2.11, is conceptually and computationally extremely simple; the only computational step being calculation of inner products of n-bit vectors. As observed in [48], this step can be easily transformed into a boolean formula. Furthermore, [27] requires that the set S of n-bit vectors be "extended" to a set of $n + \log n$ bit vectors. In addition to our construction, in Chapter VI, we demonstrate several of its applications, including better bounds on the size of threshold circuits accepting languages accepted by AC^0 circuits.

Our main results can be summarized as follows:

Theorem 5.2.10 *Let* $S \subseteq \{0,1\}^n - \{0^n\}$ *be a nonempty set. Let* $w_1, w_2, \ldots, w_{n-1}$

be chosen uniformly and independently from $\{0,1\}^n$. *Define* $H_i = \{v \mid v \cdot w_i = 0\}$, $S_i = S \cap H_1 \cap H_2 \cap \ldots H_{i-1} \cap \overline{H_i}$, *for* $1 \leq i \leq n-1$, *and* $S_n = S \cap H_1 \cap H_2 \cap \ldots \cap H_{n-1}$. *Then,* $P_n(S) \stackrel{\text{def}}{=} Prob(\exists i, 1 \leq i \leq n$ [S_i *is linearly restrictable*]$) > \frac{3}{8}$. Note that the subsets S_i form a disjoint partition of set S. The notion of *linearly restrictable* sets is a generalization of sets with odd cardinality. As is evident from the following lemma, *linearly restrictable* sets can easily be reduced to sets with odd cardinality with high probability.

Lemma 5.2.7(Part 1) *Let* $S_1, S_2, \ldots, S_m \subseteq \{0,1\}^n$ *be* m *disjoint sets such that at least one of them is linearly restrictable. Let* $w_1, \ldots, w_m$ *be chosen uniformly and independently from* $\{0,1\}^n$. *Define* $S' = \{x \mid x \in S_i$ *and* $x \cdot w_i = 0, 1 \leq i \leq m\}$. *Then,* $Prob(|S'|$ *is odd* $) = Prob(|S'|$ *is even* $) = \frac{1}{2}$.

Given any non-empty set, these two results can be used to construct a set with odd cardinality, with a constant probability. The full construction is given in the proof of Theorem 5.2.11.

5.2 Main Results

In this section we show how to isolate an odd number of elements of a non-empty set. We start by stating the main theorem in [48]. The proof of our main theorem parallels the proof of the following theorem.

Theorem 5.2.1 *[48] Let* $S \subseteq \{0,1\}^n$ *be a nonempty set. Let* $w_1, w_2, \ldots$ *be chosen uniformly and independently from* $\{0,1\}^n$. *Define* $H_i = \{v \mid v \cdot w_i = 0\}$, $S_0 = S$, *and* $S_i = S_0 \cap H_1 \cap H_2 \cap \ldots \cap H_i$, *for* $i \geq 1$. *Then,* $Prob(\exists i \geq 0$ $[|S_i| = 1]) > \frac{1}{2}$.

Note that all the algebra is done in GF$[2]^n$. Given $u, v \in \{0,1\}^n, u = (u_1, \ldots, u_n), v = (v_1, \ldots, v_n)$, $u \cdot v$ denotes the inner product of u and v; that is, $u \cdot v = \sum_{i=1}^{n} u_i v_i \pmod 2$. Thus, Theorem 5.2.1 states that a set can be "linearly hit" with random vectors, each "hit" possibly reducing its size. Ultimately, one of the "hits" produces a subset of cardinality 1 with high probability. The following theorem, a corollary to the proof of Theorem 5.2.1, states that the same results can be obtained by using n linearly independent random vectors.

Theorem 5.2.2 *[48] Let $S \subseteq \{0,1\}^n$ be a nonempty set. Let $w_1, w_2, \ldots, w_n \in \{0,1\}^n$ be chosen uniformly from the sample space $\Omega \subset \{0,1\}^{n^2}$, where $w_1 w_2 \ldots w_n \in \Omega$ implies that w_i are linearly independent. Define $H_i = \{v \mid v \cdot w_i = 0\}$, $S_0 = S$, and $S_i = S_0 \cap H_1 \cap H_2 \cap \ldots \cap H_i$, for $1 \leq i \leq n$. Then, Prob$(\exists i, 0 \leq i \leq n\ [|S_i| = 1]) > \frac{1}{2}$.*

If we choose $w_1, w_2, \ldots, w_n$ uniformly from $\{0,1\}^n$, then these vectors are linearly independent with high probability. Using this, Valiant and Vazirani proved the following result.

Theorem 5.2.3 *[48] Let $S \subseteq \{0,1\}^n$ be a nonempty set. Let $w_1, w_2, \ldots, w_n$ be chosen uniformly and independently from $\{0,1\}^n$. Define $H_i = \{v \mid v \cdot w_i = 0\}$, $S_0 = S$, and $S_i = S_0 \cap H_1 \cap H_2 \cap \ldots \cap H_i$, $1 \leq i \leq n$. Then, Prob$(\exists i, 0 \leq i \leq n\ [|S_i| = 1]) > \frac{1}{4}$.*

Thus, if we pick one of the $n+1$ sets$(S_0, \ldots, S_n)$ uniformly, then it has cardinality 1 with probability at least $\frac{1}{4(n+1)}$. We are interested in making this lower bound on probability a constant, by isolating a subset of S which has odd cardinality, not necessarily cardinality exactly one. To this end we define the notion of *linearly restrictable* sets and show that sets which are not *linearly restrictable* do not change cardinality (from even to odd or vice versa) when hit by any vector.

Definition 5.2.4 *$S \subseteq \{0,1\}^n$ is called linearly restrictable if and only if $\sum_{x \in S} x \neq 0^n$. In particular, the empty set is defined to be not linearly restrictable.*

Proposition 5.2.5 *Let $S \subseteq \{0,1\}^n$ not be linearly restrictable. Then, for all $w \in \{0,1\}^n$, $|S| \equiv |S \cap H_w| \pmod 2$, where $H_w = \{v \mid v \cdot w = 0\}$.*

Proof. For all $w \in \{0,1\}^n$, $\sum_{x \in S}(x \cdot w) = (\sum_{x \in S} x) \cdot w = 0^n \cdot w = 0$. Thus, the number of elements $x \in S$ such that $x \cdot w = 1$ must be even. Hence, an even number of elements of set S are not present in $S \cap H_w$, for any w. $\square$

In particular, if S is not *linearly restrictable* and $|S|$ is even, then $|S \cap H_w|$ and $|S \cap \overline{H}_w|$ remain even, for all w. As mentioned in the introduction, *linearly restrictable* sets are a generalization of sets with odd cardinality. The following deterministic reduction, given a set S with odd cardinality, produces a *linearly restrictable* set: f maps $\{0,1\}^n$ to $\{0,1\}^{n+1}$, such that $f(x) = 1x$; that is, the vector x is prefixed by 1. It is easy to see that $|S|$ is odd implies that $f(S)$ is *linearly restrictable.*

The next proposition shows that *linearly restrictable* sets can be randomly reduced to sets with odd cardinality, with probability exactly $\frac{1}{2}$.

Proposition 5.2.6 *Let $S \subseteq \{0,1\}^n$ be linearly restrictable. Let $w \in \{0,1\}^n$ be chosen uniformly and let $H_w = \{v \mid v \cdot w = 0\}$. Then, Prob$(|S| \not\equiv |S \cap H_w| \pmod 2)$ = Prob$(|S| \equiv |S \cap H_w| \pmod 2) = \frac{1}{2}$.*

Remark: *For every linearly restrictable set S, Prob($|S \cap H_w|$ is odd) $= \frac{1}{2}$, when w is chosen uniformly.*

Proof. Let $v = \sum_{x \in S} x \neq 0^n$. Then, $\text{Prob}(v.w = 1) = \frac{1}{2}$. For w's such that $v \cdot w = 1$, $(\sum_{x \in S} x) \cdot w = \sum_{x \in S}(x \cdot w) = 1$. Therefore, for an odd number of $x \in S$, $x \cdot w = 1$. Hence, for such w's, $|S \cap H_w| \not\equiv |S| (\bmod\ 2)$. Of course, $\text{Prob}(|S| \equiv |S \cap H_w| (\bmod\ 2))$ $= 1 - \frac{1}{2} = \frac{1}{2}$. $\square$

Eventually we will show how to construct disjoint subsets of a given set S such that one of them is *linearly restrictable* with high probability. Using these disjoint subsets, we can construct one subset of S which has odd cardinality with high probability. Our next lemma shows two ways of constructing this subset given the disjoint subsets, where at least one of them is *linearly restrictable.*

Lemma 5.2.7 *Let $S_1, S_2, \ldots, S_m \subseteq \{0,1\}^n$ be m disjoint sets such that at least one of them is linearly restrictable.*

1. *Let $w_1, \ldots, w_m$ be chosen uniformly and independently from $\{0,1\}^n$. Define $S' = \{x \mid x \in S_i$ and $x \cdot w_i = 0, 1 \leq i \leq m\}$. Then, Prob($|S'|$ is odd) $=$ Prob($|S'|$ is even) $= \frac{1}{2}$.*
2. *Let w be chosen uniformly from $\{0,1\}^n$ and $b_1, \ldots, b_m$ be independently chosen bits. Define $H_w = \{x \mid x \cdot w = 0\}$ and $S' = \{x \mid x \in S_i \cap H_w$ and $b_i = 1, 1 \leq i \leq m\}$. Then, Prob($|S'|$ is odd) $\geq \frac{1}{4}$ and Prob($|S'|$ is even) $\geq \frac{1}{4}$.*

Proof.

1. Define $S'' = \{0^{(i-1)n} x 0^{(m-i)n} \mid x \in S_i\}$. Note that $|S''| = |\cup_{i=1}^{m} S_i|$. It is easy to see that if one of the subsets S_i is *linearly restrictable*, then so is S''. If we use $w = w_1 \ldots w_m$ (which denotes the concatenation of w_i's) to "hit" set S'', Proposition 5.2.6 gives Prob($|S'' \cap H_w|$ is odd) = Prob($|S'' \cap H_w|$ is odd) $= \frac{1}{2}$, where $H_w = \{x \in \{0,1\}^{mn} \mid x \cdot (w_1 w_2 \ldots w_m) = 0\}$. Note that $|S'| = |\cup_{i=1}^{m} (S_i \cap H_{w_i})| = |S'' \cap H_w|$, where $H_{w_i} = \{x \in \{0,1\}^n \mid x \cdot w_i = 0\}$, which proves the lemma.

2. It is easy to see the lemma for $m = 1$. Assume that the statement is true for $1 \le k \le m-1$. If S_m is not *linearly restrictable*, then the lemma follows by the induction hypothesis. Hence, from now on we assume that S_m is *linearly restrictable.*

   Fix the values of $b_1, \ldots, b_{m-1}$. Define $\widehat{S} = \{x \mid x \in S_i,\ b_i = 1,\ 1 \le i \le m-1\}$.

   Case 1: ($\widehat{S}$ is *linearly restrictable*) Prob($|S'|$ is odd) $\ge$ Prob($|\widehat{S} \cap H_w|$ is odd and $b_m = 0$) $= \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$.

   Case 2: ($\widehat{S}$ is not *linearly restrictable*) If $\widehat{S}$ is even, then Prob($|S'|$ is odd) = Prob($|S_m \cap H_w|$ is odd and $b_m = 1$) $= \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. If $\widehat{S}$ is odd, then Prob($|S'|$ is odd) = Prob($|S_m \cap H_w|$ is even or $b_m = 0$) $= 1 - \frac{1}{2} \times \frac{1}{2} = \frac{3}{4}$.

   Similarly, we can show that Prob($|S'|$ is even) $\ge \frac{1}{4}$.

$\square$

Note that part 1 of Lemma 5.2.7 uses $m \cdot n$ bits, while part 2 uses $m + n$ bits.

Next, we show how to partition a set $S(0^n \notin S)$ into disjoint subsets such that one of the subsets is *linearly restrictable* with high probability. We start by proving the following theorem whose proof parallels that of Theorem 5.2.1. The intuition behind the proof is as follows: Consider the sequence of subsets $S = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_i \cdots$ defined in the statement of Theorem 5.2.1. Clearly, $S_{i-1} - S_i, i = 1, 2, \ldots$ gives a partition of set S into disjoint subsets. Note that one of these subsets will have exactly one element x (when $S_{i-1} = \{x\}$ and $S_i = \emptyset$), with high probability. As $x \neq 0^n$, $S_{i-1} - S_i$ is *linearly restrictable.* Of course, if we construct only n subsets by using only n vectors $w_1, \ldots, w_n$, then we have n subsets of S such that one of the subsets is *linearly restrictable* with high probability.

We obtain better bounds on probability than given by Theorem 5.2.1 by observing that sets with rank 2 are either already *linearly restrictable*, or can be partitioned into disjoint subsets such that one of the subsets is *linearly restrictable*, with probability 1. We assume $0^n \notin S$ to preclude the trivial case when $S = \{0^n\}$, and hence no subset of S is *linearly restrictable.* Of course, in this case $|S|$ is already odd, satisfying our eventual goal.

Theorem 5.2.8 *Let $S \subseteq \{0,1\}^n - \{0^n\}$ be a nonempty set. Let $w_1, w_2, \ldots$ be chosen independently and uniformly from $\{0,1\}^n$. Define $H_i = \{v \mid v \cdot w_i = 0\}$ and $S_i = S \cap H_1 \cap H_2 \cap \ldots \cap H_{i-1} \cap \overline{H_i}$; that is, $S_i = \{x \mid x \in S$ and $x \cdot w_1 = \cdots = x \cdot w_{i-1} = 0$ and $x \cdot w_i = 1\}$, for $i \geq 1$. Then, $P(S) \stackrel{\text{def}}{=} Prob(\exists i \geq 1\ [S_i$ is linearly restrictable$]) > \frac{3}{4}$.*

Proof. Let $T_k = \min\{P(S)\}$, where S ranges over all nonempty subsets of $\{0,1\}^n - \{0^n\}$ with $\mathrm{rank}(S) \leq k$. We show by induction on k that $T_1 = 1$ and $T_k \geq \frac{3 \cdot 2^{k-2}}{2^k - 1}$, for $k \geq 2$.

If $k = 1$, then $S = \{x\} \neq \{0^n\}$ and $T_1 = 1$. A nonsingular linear transformation on $\{0,1\}^n$ permutes the vectors as well as the hyperplanes of $\{0,1\}^n$. As we are choosing w_i's uniformly from $\{0,1\}^n$, we may assume without loss of generality that if $\mathrm{rank}(S) = k$, then $\{e_1, e_2, \ldots, e_k\} \subseteq S$, where e_i is the ith unit vector, and these vectors span S.

For $k = 2$, $S = \{e_1, e_2\}$ or $S = \{e_1, e_2, e_1 + e_2\}$. In either case, it is easy to verify that $T_2 = 1$.

When randomly choosing $w_1, w_2, \ldots$, we encounter, with probability 1, an index l such that $\mathrm{rank}(S \cap H_1 \cap H_2 \cap \ldots \cap H_l) < k$. Let i be the smallest such index. Thus $S \cap H_1 \cap H_2 \cap \ldots \cap H_{i-1} = S$, and H_i satisfies $\mathrm{rank}(S \cap H_i) < k$. Given such an H_i, we will apply induction on the rank of S to obtain the result.

Assume that the statement of the theorem is true for all $1 \leq m \leq k-1$, and let $\mathrm{rank}(S) = k$. Note that $\mathrm{rank}(S \cap H_i) < k$ iff $w_i \notin 0^k\{0,1\}^{n-k}$. There are $2^n - 2^{n-k}$ such vectors for w_i. Also, $e_j \notin S \cap H_i$ for all $1 \leq j \leq k$ iff $w \in 1^k\{0,1\}^{n-k}$. Therefore,

$$\mathrm{Prob}(\forall j\ [e_j \notin S \cap H_i] \mid \mathrm{rank}(S \cap H_i) < k) = \frac{2^{n-k}}{2^n - 2^{n-k}}$$

If $e_j \in S \cap H_i$ for some j, then $S \cap H_i \neq \emptyset$. Hence, the probability that $S \cap H_i \neq \emptyset$, relative to the event that $\mathrm{rank}(S \cap H_i) < k$, satisfies

$$\mathrm{Prob}(S \cap H_i \neq \emptyset \mid \mathrm{rank}(S \cap H_i) < k) \geq 1 - \frac{2^{n-k}}{2^n - 2^{n-k}} = \frac{2^k - 2}{2^k - 1}$$

In the worst case, $S \cap \overline{H_i}$ is not *linearly restrictable*. However, if $S \cap H_i \neq \emptyset$, then we can apply induction on $\mathrm{rank}(S \cap H_i) < k$. As w_j's are chosen independently,

$$T_k \geq T_{k-1} \cdot \mathrm{Prob}(S \cap H_i \neq \emptyset \mid \mathrm{rank}(S \cap H_i) < k) \geq \frac{3 \cdot 2^{k-3}}{2^{k-1}-1}\frac{2^k-2}{2^k-1} = \frac{3 \cdot 2^{k-2}}{2^k-1}$$

□

We can show that there are sets for which $\frac{3 \cdot 2^{k-2}}{2^k-1}$ is the best we can hope to achieve, if we follow the above procedure for isolating a *linearly restrictable* subset.

The following theorems state the analogues of Theorems 5.2.2 and 5.2.3 for Theorem 5.2.8.

Theorem 5.2.9 *Let $S \subseteq \{0,1\}^n - \{0^n\}$ be a nonempty set. Let $w_1, w_2, \ldots, w_{n-1} \in \{0,1\}^n$ be chosen uniformly from the sample space $\Omega \subset \{0,1\}^{n(n-1)}$, where $(w_1 w_2 \ldots w_n) \in \Omega$ implies that w_i are linearly independent. Define $H_i = \{v \mid v \cdot w_i = 0\}$, $S_i = S \cap H_1 \cap H_2 \cap \ldots \cap H_{i-1} \cap \overline{H_i}$, for $1 \leq i \leq n-1$, and $S_n = S \cap H_1 \cap H_2 \cap \ldots \cap H_{n-1}$. Then, Prob$(\exists i, 1 \leq i \leq n$ $[S_i$ is linearly restrictable$]) > \frac{3}{4}$.*

Proof. Satisfaction of the condition "$\exists i \geq n$ $[S_i$ is linearly restrictable$]$" in Theorem 5.2.8 depends on the indices $i_1 < i_2 < \cdots < i_{n-1}$ for which

$$H_{i_1} \supset (H_{i_1} \cap H_{i_2}) \supset \cdots \supset (H_{i_1} \cap H_{i_2} \cap \cdots \cap H_{i_{n-1}}),$$

where $\mathrm{rank}(H_{i_1} \cap H_{i_2} \cap \cdots \cap H_{i_{n-1}}) = 1$. Hence, when $w_1, w_2, \ldots, w_{n-1}$ are linearly independent we achieve the same bounds as that of Theorem 5.2.8. □

As shown in [48], if we choose n vectors uniformly, then with high probability these are linearly independent. This yields the following result.

Theorem 5.2.10 *Let $S \subseteq \{0,1\}^n - \{0^n\}$ be a nonempty set. Let $w_1, w_2, \ldots, w_{n-1}$ be chosen uniformly and independently from $\{0,1\}^n$. Define $H_i = \{v \mid v \cdot w_i = 0\}$, $S_i = S \cap H_1 \cap H_2 \cap \ldots H_{i-1} \cap \overline{H_i}$, for $1 \leq i \leq n-1$, and $S_n = S \cap H_1 \cap H_2 \cap \ldots \cap H_{n-1}$. Then, $P_n(S) \stackrel{\text{def}}{=} Prob(\exists i, 1 \leq i \leq n\ [S_i$ is linearly restrictable$]) > \frac{3}{8}$.*

Proof. As $S \subseteq \{0,1\}^n - \{0^n\}$, clearly $P_n(S) \geq P(S) \cdot \text{Prob}(\text{rank}(H_1 \cap \cdots \cap H_{n-1}) = 1)$, where $P(S)$ is as defined in the statement of Theorem 5.2.8. Note that,

$$\text{Prob}(\text{rank}(H_1 \cap \cdots \cap H_{n-1}) = 1) = (\frac{2^n - 1}{2^n})(\frac{2^n - 2}{2^n}) \cdots (\frac{2^n - 2^{n-2}}{2^n}) > \frac{1}{2}$$

Thus, $P_n(S) > \frac{3}{4} \times \frac{1}{2} = \frac{3}{8}$. $\square$

The probability of obtaining n linearly independent vectors can be made arbitrarily high by choosing $m \geq n$ vectors. Hence, $P_m(S)$ can be made arbitrarily close to $\frac{3}{4}$ by choosing $m \geq n$. Finally, we combine Theorem 5.2.10 and Lemma 5.2.7 to construct, given a non-empty set, a subset with odd cardinality.

Theorem 5.2.11 *Given $S \subseteq \{0,1\}^n$, we can construct a set $S' \subseteq S$ such that if $S = \emptyset$, then $S' = \emptyset$, and if $S \neq \emptyset$, then $Prob(|S'|$ is odd $) > \frac{3}{16}$.*

Proof. Given $S \subseteq \{0,1\}^n$, guess $w_1, w_2, \ldots, w_{n-1}$ and $z_1, z_2, \ldots, z_n$ uniformly and independently from $\{0,1\}^n$. The set S' is constructed as follows:

1. $0^n \in S' \Leftrightarrow 0^n \in S$.

2. $x \neq 0^n \in S' \Leftrightarrow x \in S$ and $z_i \cdot x = 0$, where i is such that $w_1 \cdot x = w_2 \cdot x = \cdots = w_{i-1} \cdot x = 0$ and, if $i \leq n-1$, then $w_i \cdot x = 1$.

It is easy to see that S' has the desired properties by Theorem 5.2.10 and Lemma 5.2.7(part 1). $\square$

CHAPTER VI

Complexity Theory Applications of Isolating an Odd Number of Elements

Wisdom consists in knowing what to do with what you know.

— Source Unknown

In this chapter we show how isolating an odd number of elements can be used to improve some of the recent results in complexity theory. We assume that the reader is familiar with the original results and hence avoid giving the full details. However, we provide enough details to demonstrate the specific steps of the original proofs which can be improved by using our construction.

6.1 Approximating Boolean Functions by Polynomials

Let $\mathrm{OR}(x_1, \ldots, x_n)$ and $\mathrm{NOR}(x_1, \ldots, x_n)$ denote the Boolean functions OR and NOR respectively; that is, $\mathrm{OR}(x_1, \ldots, x_n) = 1$ iff $x_i = 1$, for some i, and $\mathrm{NOR}(x_1, \ldots, x_n) = 1 - \mathrm{OR}(x_1, \ldots, x_n)$. Given a prime number q, Smolensky[40] and Razborov[34] constructed the following random polynomials of degree $\mathrm{O}(q)$, using $\mathrm{O}(qn)$ uniformly chosen random bits:

$$p(x_1, \ldots, x_n) = (\sum_{i=1}^{n}(r_i x_i))^{q-1} (\mathrm{mod}\ q),$$

where $r_1, r_2, \ldots, r_n$ are chosen independently and uniformly from $\{0, 1, \ldots, q-1\}$. It is easy to show that if $\mathrm{OR}(x_1, \ldots, x_n) = 0$, then $p(x_1, \ldots, x_n) = 0$, and if $\mathrm{OR}(x_1, \ldots, x_n) = 1$, then $p(x_1, \ldots, x_n) = 1$ with probability at least $1 - \frac{1}{q}$. Of course, this probability of success can be easily amplified to any value. For example, if we construct l random polynomials $p_1, \ldots, p_l$ independently as described above, $p(x_1, \ldots, x_n) = (\prod_{j=1}^{l}(p_j(x_1, \ldots, x_n) - 1)) + 1 = \mathrm{OR}(x_1, \ldots, x_n)$, with probability at least $1 - \frac{1}{q^l}$.

Now consider the following theorem from [48].

Theorem 6.1.1 *[48] Let $S \subseteq \{0,1\}^n$ be a nonempty set. Let $w_1, w_2, \ldots, w_n$ be chosen uniformly and independently from $\{0,1\}^n$. Define $H_i = \{v \mid v \cdot w_i = 0\}$, $S_0 = S$, and $S_i = S_0 \cap H_1 \cap H_2 \cap \ldots \cap H_i$, $1 \leq i \leq n$. Then, Prob$(\exists i, 0 \leq i \leq n\ [|S_i| = 1]) > \frac{1}{4}$.*

Using this random reduction, Tarui[43] constructed a random polynomial $p(x_1, \ldots, x_n)$ of degree $\mathrm{O}(\log n)$ using $\mathrm{O}(\log^2 n)$ uniformly chosen random bits such that $p(x_1, \ldots, x_n) = \mathrm{NOR}(x_1, \ldots, x_n)$, with arbitrarily high probability, as follows: Let $N = 2^{\lceil \log n \rceil}$. Pick $w_1, \ldots, w_{\log N}$ uniformly and independently from $\{0,1\}^{\log N}$, and for each $0 \leq i \leq \log N$ and each $1 \leq j \leq n$, let

$$r_j^i = \begin{cases} 1 & \text{if binary}(j-1) \cdot w_1 = \cdots = \text{binary}(j-1) \cdot w_i = 0\ (r_j^0 = 1) \\ 0 & \text{otherwise,} \end{cases}$$

where binary(r) denotes the value of r written as a $\log N$-bit vector. Let

$$p(x_1, \ldots, x_n) = \prod_{i=0}^{\log N} (r_1^i x_1 + \cdots + r_n^i x_n - 1)^2.$$

It is easy to see, using Theorem 6.1.1, that if $\mathrm{NOR}(x_1, \ldots, x_n) = 0$, then $p(x_1, \ldots, x_n) = 0$, and if $\mathrm{NOR}(x_1, \ldots, x_n) = 1$, then $p(x_1, \ldots, x_n) = 1$ with probability at least $\frac{1}{4}$. As mentioned before, the probability of success can be easily amplified to any value.

With our reduction, we can construct a random polynomial $p(x_1, \ldots, x_n)$ of degree O(1), using $O(\log^2 n)$ uniformly chosen bits such that $p(x_1, \ldots, x_n) = \mathrm{NOR}(x_1, \ldots, x_n) \pmod 2$, with arbitrarily high probability. The construction is similar to that of Tarui's polynomials; the correctness follows from the construction described in the proof of Theorem 5.2.11. Let $N = 2^{\lceil \log n \rceil}$. Pick $w_1, \ldots, w_{\log N}$, $z_1, \ldots, z_{\log N}$ uniformly and independently from $\{0,1\}^{\log N}$, and for each $1 \le j \le n$, let

$$r_j = \begin{cases} 1 & \text{if } \mathrm{binary}(j-1) \cdot z_i = 0, \text{ where } i \text{ is such that } \mathrm{binary}(j-1) \cdot w_1 = \cdots \\ & = \mathrm{binary}(j-1) \cdot w_{i-1} = 0 \text{ and, if } i \le (\log N) - 1, \text{ then } \mathrm{binary}(j-1) \cdot w_i = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Let

$$p(x_1, \ldots, x_n) = (r_1 x_1 + \cdots + r_n x_n - 1)^2.$$

Then, $p(x_1, \ldots, x_n) = \mathrm{NOR}(x_1, \ldots, x_n) \pmod 2$, with probability at least $\frac{3}{16}$. Of course, $1 - p(x_1, \ldots, x_n)$ approximates $\mathrm{OR}(x_1, \ldots, x_n)$. As discussed before this probability can be amplified to any value. Hence, our reduction gives the same results as that of Smolensky's(for $q = 2$), but uses fewer random bits[1].

6.2 Random Reduction From the Polynomial-time Hierarchy to $\oplus P$

Using Theorem 6.1.1, Valiant and Vazirani constructed a random reduction from NP(equivalently, $\exists \cdot P$) to $\oplus$P (a language $L \in \oplus$P iff there exists a polynomial time-bounded nondeterministic machine M such that $x \in L \Leftrightarrow M$ accepts x on an odd

[1] Using the construction implicit in the proof of [27], the number of random bits can be reduced to $\log n$. However, the construction is not quite as straightforward as ours.

number of computation paths). Intuitively, if the NP machine guesses $p(n)$-bits, then the set S in Theorem 6.1.1 contains all correct guesses. Since, $\oplus$P is closed under Turing reduction[30], the probability of success can be amplified from $\frac{1}{4n}$ to $\frac{3}{4}$ using O(n) independent repetitions. This yields $\exists \cdot P \subseteq \mathrm{BP} \cdot \oplus \mathrm{P}$ [2]. As BP and $\oplus$P are closed under complement, Valiant and Vazirani actually showed that $(\exists \cdot \mathrm{P}) \cup (\forall \cdot \mathrm{P}) \subseteq \mathrm{BP} \cdot \oplus \mathrm{P}$. To obtain this result we do not need to isolate exactly one element; an odd number of elements suffices. If we use our theorem, we can amplify the probability of success to $\frac{3}{4}$ using O(1) repetitions. Thus, if the NP machine guesses $\mathrm{O}(p(n))$ bits, the $\oplus$P machine, constructed by using our reduction, guesses $\mathrm{O}(p(n))$ bits and has a running time bound of $\mathrm{O}(p^2(n))$. If we use the reduction in [48], the $\oplus$P machine guesses $\mathrm{O}(p^2(n))$ bits and has a running time bound of $\mathrm{O}(p^4(n))$ steps. Hence, the $\oplus$P machine with our construction needs no more nondeterministic guesses than the NP machine.

Toda[44] generalized the result of [48] to show that the entire polynomial-time hierarchy (characterized by a finite number of alternating existential and universal quantifiers over polynomial time) is contained in $\mathrm{BP} \cdot \oplus \mathrm{P}$. Toda's original proof collapses Σ_d^P into $\mathrm{BP} \cdot \oplus \mathrm{P}$ using induction on d. The existential and universal quantifiers at each level are converted into $\mathrm{BP} \cdot \oplus \mathrm{P}$ as follows: Under the induction hypothesis $\Sigma_d^P \cup \Pi_d^P \subseteq \mathrm{BP} \cdot \oplus \mathrm{P}$ we have,

$$\begin{aligned} \Sigma_{d+1}^P = \exists \cdot \Pi_d^P \quad & \subseteq \exists \cdot \mathrm{BP} \cdot \oplus \mathrm{P} \\ & \subseteq \mathrm{BP} \cdot \oplus \mathrm{P} \cdot \mathrm{BP} \cdot \oplus \mathrm{P} \\ & \subseteq \mathrm{BP} \cdot \mathrm{BP} \cdot \oplus \mathrm{P} \cdot \oplus \mathrm{P} \\ & \subseteq \mathrm{BP} \cdot \oplus \mathrm{P} \end{aligned}$$

[2] A language $L \in \mathrm{BP} \cdot K$, for some class K, iff there exists $A \in K$ and a polynomial $p(n)$ such that $\mathrm{Prob}(x \in L \Leftrightarrow \langle x, y \rangle \in A) \geq \frac{3}{4}$, where y is chosen uniformly from $\{0,1\}^{p(|x|)}$.

A simple analysis of this result shows that if $L \in \Sigma_d^P$ is accepted by a $p(n)$ time-bounded alternating Turing machine which makes d alternations, the $\oplus$P machine accepting the same language requires $O(p^{2^d}(n))$ nondeterministic steps.

An elegant algorithmic proof of some extensions of Toda's result is given by Tarui[43]. The key steps of the proof are given below:

1. Let the language $L \in \Sigma_d^P$ be accepted by a $p(n)$ time-bounded alternating Turing machine which makes d alternations. The computation tree of such an alternating machine can be viewed as a "stratified" circuit of depth d with alternating AND/OR gates. A "stratified" circuit has connections only between consecutive layers. The fan-in of each gate is $O(2^{p(n)})$.

2. [43]A "stratified" circuit of depth d with alternating AND/OR gates can be viewed as a circuit with only NOR gates.

3. Each NOR gate can be approximated by a polynomial of degree O(1) with high probability, using our construction. If, instead, we use the polynomials constructed using Theorem 6.1.1, the polynomial has degree $O(p(n))$. With $O(p(n))$ independent trials we can amplify the success probability to construct a polynomial of degree $O(p(n))$ which approximates *all* the $2^{p(n)}$ NOR gates in the circuit of part 2, with high probability.

4. As demonstrated in [43], the polynomial for each NOR gate can be substituted in the polynomial for its parent gate. By repeating this construction for all levels, we obtain one polynomial of degree $O(p^{d-1}(n))$ with $O(2^{p^d(n)})$ terms,

which approximates the whole depth d circuit. These polynomials can be used to construct a $\oplus$P machine which accepts L with a high probability using $O(p^d(n))$ nondeterministic guesses and $O(p^{2d}(n))$ deterministic steps. Tarui uses Theorem 6.1.1, and hence his polynomials have degree $O(p^{2d}(n))$ with $O(2^{p^{2d}(n)})$ terms. The corresponding $\oplus$P machine requires $O(p^{2d}(n))$ nondeterministic guesses and $O(p^{4d}(n))$ deterministic steps.

Thus, using our construction, instead of Theorem 6.1.1, gives better bounds for the $\oplus$P machine.

6.3 AC^0 and Threshold circuits

Allender[1], Tarui[43], Yao[51], and Beigel and Tarui[6], have shown that languages accepted by various extensions of AC^0 can be accepted by threshold circuits. (AC^0 is the class of languages accepted by constant-depth polynomial-sized circuits consisting of AND/OR/NOT gates with unbounded fan-in.) All of these results use Valiant and Vazirani type of reductions. We can improve the bounds on the size of threshold circuits by using our construction.

Consider a depth-d stratified AC^0 circuit such that the fan-in of each gate is bounded by n^k. The key steps for constructing a depth-2 threshold circuit, which accepts the same language as the AC^0 circuit, are given below:

1. Following the steps of Section 6.2, we can obtain a polynomial $p(x_1, \ldots, x_n)$ which approximates the circuit with high probability. Using our construction, the polynomial has degree $O(\log^d n)$ with $2^{O(\log^d n)}$ terms. If we use Theorem

6.1.1, as done in [43], the polynomial has degree $O(\log^{2d} n)$ with $2^{O(\log^{2d} n)}$ terms. The number of random bits used is $O(\log^3 n)$. This polynomial can be converted into a depth-2 circuit with a parity gate at the top and AND gates at level 2, as done in [1, 43, 6, 51], by using AND gates for each term, and the parity gate at the top for summing all the terms. Hence, the fan-in of each AND gate is bounded by $O(\log^d n)$, and the fan-in of the parity gate is bounded by $2^{O(\log^d n)}$.

2. The need for randomness is removed by taking a majority over all the possible values of the $O(\log^3 n)$ random bits. This gives a depth-3 circuit with a majority gate of fan-in $2^{O(\log^3 n)}$ at the top, parity gates at level 2, and AND gates at level 3.

3. Using modulus amplifying polynomials (proof of Lemma 5 in [6]), the majority and parity gates can be collapsed into one majority gate. This gives a depth-2 circuit of size $2^{O(\log^{d+3} n)}$, with a majority gate at the top and AND gates at level 2. Using Theorem 6.1.1, instead of our construction, gives a size of $2^{O(\log^{2d+3} n)}$. If we use the construction implicit in [27], the number of random bits used is $O(\log^2 n)$, which gives a bound of $2^{O(\log^{d+2} n)}$ on the size of threshold circuit.

6.4 Concluding Remarks

We demonstrated a very straightforward algebraic method of isolating an odd number of elements of a non-empty set with constant probability in Chapter V. This construction can be used to improve bounds on several important results of complexity theory. The most important of these applications are circuit results: languages accepted by depth-d AC^0 circuits can be accepted by depth-2 threshold circuit of size $2^{O(\log^{d+3} n)}$. Now that we know that the exponent of $\log n$ grows exactly with d, can this be improved any further? If we use the construction implicit in the proof of [27], which isolates an odd number of elements using $O(n)$ bits, the bound on the size of threshold circuit is $2^{O(\log^{d+2} n)}$. However, we are interested in making the exponent grow slower than d, even if this involves increasing the depth of threshold circuits to beyond 2. Such results should provide additional relationships between counting classes and PSPACE.

CHAPTER VII

Conclusions

Work is the greatest thing in the world. So we should save some of it for tomorrow.

— Don Herold

In this dissertation we have considered three sets of results, all related to increasing our understanding of the structure of various complexity classes and relationships among them. The most important of these (yet undetermined) relationship is the relationship between P and NP: Is P strictly contained in NP, or are they same?

As mentioned in the introduction, P $\neq$ NP if and only if NTIME(n) $\not\subseteq$ DTIME(n^k), for all $k \geq 1$. It is not difficult to show that NTIME(n) $\subseteq$ DTIME(n^k), for some $k \geq 1$ if and only if for all l, $\Sigma_l(n) \subseteq$ DTIME(n^k), for some $k \geq 1$. Thus, P $\neq$ NP if and only if for some l, $\Sigma_l(n) \not\subseteq$ DTIME(n^k), for all $k \geq 1$. Our main result in Chapter II solves an easier problem. We show that $\Sigma_2(n^k) \not\subseteq$ DTIME(n^k), for all $k \geq 1$. Thus, alternating machines with two alternations are strictly more powerful than deterministic machines.

Our main theorem also has interesting consequences for $\Sigma_1(t(n))$ (which is same as NTIME($t(n)$)); at least one of the following statements has to be true.

1. NTIME(n) $\neq$ co-NTIME(n).

2. DTIME($t(n)$) $\neq$ NTIME($t(n)$), for each "reasonable" $t(n)$.

There are several open problems related to this set of results. We would like to know if both of the above statements are true, as is generally believed. We would also like to know if NTIME(n) $\not\subseteq$ DTIME(n^2) or, in general, NTIME(n) $\not\subseteq$ DTIME(n^k)? As mentioned earlier, this is equivalent to resolving P $\stackrel{?}{=}$ NP question.

Our second set of results are related to increasing our understanding of the structure of class NP. Ogiwara and Hemachandra studied the closure of #P — functions computing the number of solutions of instances of problems in NP — under various functions. It is easy to show that #P is closed under addition and multiplication. However, they showed closure of #P under proper subtraction or integer division has drastic consequences. They also proved that closure of #P under proper subtraction or integer division implies closure of #P under every function in #P. (Such functions, closure under which implies closure under every other function in the class, are called "hard functions" by Ogiwara and Hemachandra.) Note that since the range of functions in #P is the set of natural numbers, they considered proper subtraction and integer division.

We defined two natural extensions of #P: $\mathcal{Z}$#P and $\mathcal{Q}$#P. $\mathcal{Z}$#P is the closure of #P under subtraction, and hence is closed under subtraction, by definition. $\mathcal{Q}$#P is the closure of $\mathcal{Z}$#P under division, and hence is closed under subtraction and division. However, we proved that proper subtraction and integer division remain hard functions for $\mathcal{Z}$#P and $\mathcal{Q}$#P as well; that is, for example, closure of $\mathcal{Z}$#P under proper subtraction implies that $\mathcal{Z}\#\mathrm{P}$ is closed under every function in $\mathcal{Z}\#\mathrm{P}$. Furthermore, we showed that closure of $\mathcal{Z}\#\mathrm{P}$ and $\mathcal{Q}\#\mathrm{P}$ under proper subtraction and integer division has drastic consequences as well. Unlike general subtraction and integer division, proper subtraction and integer division simplify the witness functions for $\mathcal{Z}\#\mathrm{P}$ and $\mathcal{Q}\#\mathrm{P}$. We have identified this simplification, called witness reduction, as the cause for such collapses. This thesis is supported by similar results for several function classes.

We also studied a probabilistic version of witness reduction called random witness reduction. In this context we extended the definitions of Ogiwara and Hemachandra to "almost closure" and obtained probabilistic versions of their results for several function classes. All these results provide evidence that it is unlikely that we will be able to efficiently simplify solutions of instances of problems in NP. Also, we believe that for a function to be hard for a function class, it must effect some form of witness reduction in that class.

Our third set of results continue on the theme of random witness reduction implicit in several other results in complexity theory. Valiant and Vazirani [48] constructed a random reduction from a non-empty set to a set with cardinality exactly 1. This result has been extensively applied in complexity theory to show containment of the polynomial-time hierarchy in various counting classes[44, 46, 43], and to show that languages accepted by various constant depth circuits can also be accepted by depth-2 threshold circuits of superpolynomial size[1, 43, 51, 6]. In several of these complexity theory applications, it is sufficient to have a subset with an odd number of elements, not necessarily exactly one. We constructed a random reduction from an arbitrary non-empty set to a set with an odd number of elements with a probability of success at least $\frac{3}{16}$. Thus, the probability of success, if we are only interested in isolating an odd number of elements, is independent of n.

Finally, in Chapter VI, we improved bounds on several results in complexity theory by using our construction given in Chapter V, in place of Valiant and Vazirani's construction[48]. In particular, we obtained better bounds on polynomials which approximate boolean functions, improved bounds on the running time of the $\oplus$P machines in Toda's result, and improved bounds on the size of threshold circuits accepting languages accepted by polynomial-sized constant-depth circuits with only AND/OR gates (AC^0 circuits).

The most important of these applications are circuit results: languages accepted by depth-d AC^0 circuits can be accepted by depth-2 threshold circuit of size $2^{O(\log^{d+3} n)}$. Now that we know that the exponent of $\log n$ grows exactly with d, can this be improved any further? If we use the construction implicit in the proof of [27], which isolates an odd number of elements using $O(n)$ bits, the bound on the size of threshold circuit is $2^{O(\log^{d+2} n)}$. However, we are interested in making the exponent grow slower than d, even if this involves increasing the depth of threshold circuits to beyond 2. Such results should provide additional relationships between counting classes and PSPACE.

Appendix A

Fixed tape simulation of Σ_k machines

Originality is not seen in single words or even sentences. Originality is the sum total of a man's thinking or his writing.

— Isaac Bashevis Singer

In this appendix we show that a fixed tape Σ_l-universal machine can simulate Σ_l-machines (with arbitray number of tapes) with a time-loss of only a constant factor. The idea for the lemma comes from Book, Greibach and Wegbreit [8]. They showed that two tapes are sufficient to accept any language accepted by nondeterministic TMs with an arbitrary number of tapes with a time-loss of only a constant factor. A nondeterministic TM with two tapes can guess a computation of any k-tape machine and then verify the computation of each tape separately in a serial fashion reusing the same tape. A little care needs to be taken to generalize this idea to any Σ_l-machine.

Lemma A.0.1 *Let $t(n)$ be a time-constructible function. Then, for each $l \geq 1$, there exists a fixed tape (depending only on $t(n)$) Σ_l universal TM U which on input $\langle M, x\rangle$, of length n, simulates $t(n)$ steps of M on x for at most l alternations, where M is an arbitrary alternating machine. Furthermore, U accepts $\langle M, x\rangle$ in time $(c_{\mathrm{M}} + c) \cdot t(n)$*

if and only if M accepts x with l alternations in $t(n)$ steps, where c is a fixed constant and $c_{\mathrm{M}} \geq 1$ is a constant which depends only on M.

Proof. We give a high-level description of U. The idea behind simulating an ATM with an arbitrary number of tapes, on a fixed number of tapes, is to guess the computation of the simulated machine and then verify the guess using a fixed number of tapes. Thus, if the simulated machine M accepts a language in $t(n)$ steps, the universal machine U accepts the same language in $c_{\mathrm{M}} \cdot t(n)$ steps, where c_{M} depends only on the simulated machine M. The universal machine requires an additional $ct(n)$ steps for syntax checking. A little care needs to be taken when M enters a universal state. When this happens, U must accept if it guesses the computation of M incorrectly.

The display of a TM M at any stage is the set of characters on the tape squares where the tape heads are currently pointing, together with the current state. Note that a display is sufficient to determine the next move of a TM. The algorithm described below guesses a sequence of displays and moves and then serially verifies the guessed sequence of displays for each tape separately.

<u>Universal machine U</u>

Input w;

If w is not of the form $\langle M, x\rangle$ then reject;

Let M be a m-tape machine;

clock $:= t(|w|)$; alternations $:= l$;

```
loop
  (* Simulate existential moves *)
  U enters an existential state;
  Decrement(alternations).
  repeat
    Guess a legal display for M (this consists of guessing a state of M and m
    characters, one corresponding to each tape head);
    Guess a legal move for M (this consists of guessing a new state of M, m
    characters written by each tape and movements of each tape head);
    if the guessed move does not follow from the guessed display then Reject;
    Save the legal display and the tape movement;
    Decrement(clock);
  until (clock=0 or New state is a univeral state of M);
  Verify the guessed displays with respect to tape head movements and transition
  function of M, for each tape separately, serially reusing the same work space,
  rejecting if incorrect;
  if the new state is an accepting state (or a rejecting state) then
  Accept (or reject, respectively);
  if alternations = 0 or clock = 0 then Reject;
  (* Simulate universal moves *)
  U enters a universal state; Decrement(alternations).
  repeat
```

Guess a legal display for M;

Guess a legal move for M;

if the guessed move does not follow from the guessed display **then** Accept;

Save the legal display and the tape movement;

Decrement(clock);

until (clock=0 **or** New state is an existential state of M);

Verify the guessed displays with respect to tape head movements and transition function of M, for each tape separately, *serially reusing the same work space*, accepting if incorrect;

if the new state is an accepting state (or a rejecting state) **then**

Accept (or reject, respectively);

if alternations = 0 **or** clock = 0 **then** Reject;

endloop

Note that the number of times the block of statements inside either one of the **repeat** ...**until** loop is executed is bounded by $t(n)$. Furthermore one execution of statements in each block takes only O(1) steps, the constant depends only on the description of M. Checking the guessed displays takes $O(t(n))$ steps, the constant depends on the machine M, and in particular, on how many tapes M has. The number of times this checking is done is bounded by l. Checking the syntax $\langle M, x\rangle$, and computing $t(|w|)$ takes at most $O(t(n))$ steps. Hence, the running time of U is bounded by $(c_{\mathrm{M}} + c)t(n)$. Also, it is easy to see that U is a Σ_l-machine.

To see that U accepts $\langle M, x\rangle$ if and only if M accepts x in $t(|\langle M, x\rangle|)$ steps, note that U simulates M on x for exactly $t(|\langle M, x\rangle|)$ steps. Also, U guesses all the possible computations of M on x, and hence including the correct ones. Furthermore U follows the existential and universal states of M; that is, if M enters an existential(universal) state, so does U.

In an existential state, U rejects on wrong guesses of computation of M. Thus the paths correspoding to the wrong guesses are immediately pruned off. Hence, an existential computation subtree of U is essentially same as that of M with some extra rejects. These rejects clearly do not effect the acceptance criterion. Similarly, a universal computation subtree of U is essentially same as that of M with extra *accepts*, which again does not affect the acceptance criterion. Finally, note that on the correct guesses of computation of M, U accepts if and only if M does so. $\square$

Appendix B

Notations and Definitions

Short words are best, and old words when short are best of all.

— Winston Churchill

Basic Notation

$\|x\|$	Length of string x
$\\|\{\cdot\}\\|$	Cardinality of set $\{\cdot\}$
$\langle\cdot,\cdot\rangle$	Any standard pairing function
$\mathcal{N}$	Set of natural numbers (including 0)
$\mathcal{Z}$	Set of integers
$\mathcal{Q}$	Set of rational numbers
$a \ominus b$	Proper subtraction: $\max(a-b, 0)$
$a \div b$	Integer division: $\lfloor \frac{a}{b} \rfloor$

Models of Computation

TM	Multi-tape Turing machine
NTM	Multi-tape nondeterministic Turing machine

ATM — Multi-tape alternating Turing machine.
A k-tape alternating Turing machine M is a five-tuple: $M = \langle Q, \Sigma, \delta, q_0, g\rangle$, where Q, Σ, q_0, and δ are defined as for the nondeterministic Turing machine [2]; and $g : Q \longmapsto \{\forall, \exists, \mathrm{acc}, \mathrm{rej}\}$ is a function which partitions the states of Q into respectively universal, existential, accepting, and rejecting.
Consider the computational tree of an ATM on input x, where each internal node is either existential or universal. Consider the following labeling procedure:

1. the accepting leaves are labeled 1;
2. any existential node is labeled 1 if at least one of its sons has been labeled 1;
3. any universal node is labeled 1 if all its sons are labeled 1;

The machine M accepts input x if and only if the root is labeled 1.

Transducer — Multi-tape Turing machine, deterministic or nondeterministic, that outputs values on an output tape on each computation path.

Functions on NTMs

$\mathrm{Acc}(M,x)$	Number of accepting paths of an NTM M on input x
$\mathrm{Rej}(M,x)$	Number of rejecting paths of an NTM M on input x
$\mathrm{AR}(M,x)$	$\mathrm{Acc}(M,x) - \mathrm{Rej}(M,x)$
$\mathrm{Out}(M,x)$	Set of all outputs of the transducer M on input x
$\mathrm{span}_k(a_1, a_2, \ldots, a_k)$	Number of distinct values in $\{a_1, a_2, \ldots, a_k\}$

Basic Complexity Classes

$\mathrm{DTIME}(t(n))$	Languages accepted by a deterministic TM in time $t(n)$.

NTIME$(t(n))$	Languages accepted by an NTM in time $t(n)$.
ATIME$(t(n))$	Languages accepted by an ATM in time $t(n)$.
$\Sigma_l(t(n))$	Languages accepted by an ATM that starts in an existential state and makes at most l alternations, in time $t(n)$.
DSPACE$(s(n))$	Languages accepted by a deterministic TM in space $s(n)$
NSPACE$(s(n))$	Languages accepted by an NTM in space $s(n)$.

High-level Complexity Classes

P	Languages accepted by polynomial time bounded deterministic TMs.
NP	Languages accepted by polynomial time bounded NTMs; that is, a language $L \in$ NP iff there exists a polynomial time bounded NTM M such that, for all x, $x \in L \Leftrightarrow \mathrm{Acc}(M, x) \geq 1$.
UP	A language $L \in$ UP iff there exists a polynomial time bounded NTM M such that, for all x, $x \in L \Leftrightarrow \mathrm{Acc}(M, x) = 1$ and $x \notin L$ $\Leftrightarrow \mathrm{Acc}(M, x) = 0$.
PP	A language $L \in$ PP iff there exists a polynomial time bounded NTM M such that, for all x, $x \in L \Leftrightarrow$ more than half the computation paths accept.
$\mathrm{C}_{=}\mathrm{P}$	A language $L \in \mathrm{C}_{=}\mathrm{P}$ iff there exists a polynomial time bounded NTM M, and $g \in$ PF such that, for all x, $x \in L \Leftrightarrow g(x) = f(x)$.
$\oplus$P	A language $L \in \oplus$P iff there exists a polynomial time bounded NTM M such that, for all x, $x \in L \Leftrightarrow \mathrm{Acc}(M, x) = 1 \pmod 2$.
PSPACE	Languages accepted by deterministic TMs in space n^k, for some $k \geq 1$.

Basic Function Classes

PF	A function $f \in$ PF iff there exists a polynomial time bounded deterministic transducer M such that, for all x, $f(x) =$ output of M on x.
#P	A function $f : \Sigma^* \longmapsto \mathcal{N}$ is in #P iff there exists a polynomial time bounded nondeterministic machine M such that for all x, $f(x) = \mathrm{Acc}(M, x)$.
$\mathcal{Z}$#P	A function $f : \Sigma^* \longmapsto \mathcal{Z}$ is in $\mathcal{Z}$#P iff there exists a polynomial time bounded nondeterministic machine M such that for all x, $f(x) = \mathrm{AR}(M, x)$. This class is same as GapP[12].
$\mathcal{Q}$#P	A function $f : \Sigma^* \longmapsto \mathcal{Q}$ is in $\mathcal{Q}$#P iff there exists a pair of polynomial time bounded nondeterministic machines (M^n, M^d) such that for all x, $\mathrm{AR}(M^d, x) \neq 0$ and $f(x) = \frac{AR(M^n,x)}{AR(M^d,x)}$.
OptP	A function $f : \Sigma^* \longmapsto \mathcal{N}$ is in OptP iff there exists a polynomial time bounded nondeterministic transducer M such that for all x, $f(x) = \max(\mathrm{Out}(M, x))$.
SpanP	A function $f : \Sigma^* \longmapsto \mathcal{N}$ is in SpanP iff there exists a polynomial time bounded nondeterministic Turing transducer M such that, for all x, $f(x) = \|\mathrm{Out}(M, x)\|$.

Operators on Complexity Classes

co-K	$L \in$ co-K iff $\overline{L} \in K$.
BP $\cdot K$	For a class K, a set L is in BP$\cdot K$ if there exists a polynomial b, and a set $A \in K$ such that for every $x \in \Sigma^*$, $\mathrm{Prob}(x \in L \Leftrightarrow \langle x, w\rangle \in A) \geq \frac{3}{4}$, where $w \in \{0,1\}^{b(\lvert x \rvert)}$ is chosen uniformly.
$\exists \cdot K$	For a class K, a set L is in $\exists \cdot K$ if there exists a polynomial p and a set $A \in K$ such that for every $x \in \Sigma^*$, $x \in L \Leftrightarrow \|\{y \mid \lvert y \rvert = p(\lvert x \rvert) \text{ and } \langle x, y\rangle \in A\}\| \geq 1$.

$\forall \cdot K$ — For a class K, a set L is in $\forall \cdot K$ if there exists a polynomial p and a set $A \in K$ such that for every $x \in \Sigma^*$, $x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}\| = 2^{p(|x|)}$.

$\mathrm{C} \cdot K$ — For a class K, a set L is in $\mathrm{C} \cdot K$ if there exists a polynomial p, a $f \in \mathrm{PF}$ and a set $A \in K$, such that for every $x \in \Sigma^*$, $x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}\| \geq f(x)$

$\mathrm{C}_= \cdot K$ — For a class K, a set L is in $\mathrm{C}_= \cdot K$ if there exists a polynomial p, $f \in \mathrm{PF}$, and a set $A \in K$ such that for every $x \in \Sigma^*$, $x \in L \Leftrightarrow$ $\|\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}\| = f(x)$.

$\oplus \cdot K$ — For a class K, a set L is in $\oplus \cdot K$ if there exists a polynomial p and a set $A \in K$ such that for every $x \in \Sigma^*$, $x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}\| = 1(\text{mod } 2)$.

K^A — Class K relative to oracle A.

$\mathrm{Almost}(K)$ — $L \in \mathrm{Almost}(K)$ iff $\mu(\{A \mid L \in K^A\}) = 1$, where μ denotes Lebesgue measure.

Hierarchies

PH

1. $\mathrm{P} \in \mathrm{PH}$.
2. If $K \in \mathrm{PH}$, then $\exists \cdot K \in \mathrm{PH}$ and $\forall \cdot K \in \mathrm{PH}$.
3. PH consists of only the classes described by 1 and 2.

CH

1. $\mathrm{P} \in \mathrm{CH}$.
2. If $K \in \mathrm{CH}$, then $\exists \cdot K \in \mathrm{CH}$, $\forall \cdot K \in CH$ and $\mathrm{C} \cdot K \in \mathrm{CH}$.
3. CH consists of only the classes described by 1 and 2.

Note: CH and PH usually denote both the family of classes as well as class of languages. In the latter case $L \in \mathrm{CH}$ (or, $L \in \mathrm{PH}$) iff $L \in K$ for some $K \in \mathrm{CH}$ (or, $L \in \mathrm{PH}$).

Circuit Classes

AC^a	Polynomial-sized circuits of depth $O(\log^a n)$, with AND, OR, and NOT gates of *unbounded* fan-in
NC^b	Polynomial-sized circuits of depth $O(\log^b n)$, with AND, OR, and NOT gates of *bounded* fan-in
TC^0	Constant-depth polynomial-sized circuits with threshold gates

Bibliography

[1] E. Allender. A note on the power of threshold circuits. In *Proceedings 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 580–584, 1989. A journal version is to appear as: E. Allender and U. Hertrampf, Depth reductions for circuits of unbounded fan-in, Information and Computation.

[2] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, Berlin, 1988. EATCS Monographs on Theoretical Computer Science.

[3] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* Springer-Verlag, Berlin, 1990. EATCS Monographs on Theoretical Computer Science.

[4] R. Beigel, J. Gill, and U. Hertramph. Counting classes: Thresholds, parity, mods, and fewness. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 49–57. Springer-Verlag, Berlin, 1990.

[5] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 1–9, 1991.

[6] R. Beigel and J. Tarui. On ACC. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 783–792, 1991.

[7] C. H. Bennett and J. Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{Co-NP}^A$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, February 1981.

[8] R. V. Book, S. A. Greibach, and B. Wegbrett. Time- and tape-bounded turing acceptors and AFLs. *Journal of Computer and System Sciences*, 4(6):606–621, December 1970.

[9] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy ii: Applications. *SIAM J. Comput.*, 18(1):95–111, 1989.

[10] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *JACM*, 8(1):114–133, 1981.

[11] P.W. Dymond and M. Tompa. Speedups of deterministic machines by synchronous parallel machines. In *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, pages 336–343, 1983.

[12] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. In *Proceedings of 6th Structure in Complexity Theory Conference*, pages 30–42. IEEE Computer Society Press, 1991.

[13] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. In *Proceedings of 6th Structure in Complexity Theory Conference*, pages 13–15. IEEE Computer Society Press, 1991.

[14] S. Gupta. A note on the counting hierarchy. Technical Report OSU-CISRC-8/90-TR24, The Ohio State University, Col, OH, August 1990.

[15] S. Gupta. On isolating odd number of elements and its applications in complexity theory. Technical Report OSU-CISRC-6/93-TR24, The Ohio State University, Col, OH, June 1993.

[16] J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Trans. American Mathematical Society*, 117:285–306, 1965.

[17] J.E. Hopcroft, W.J. Paul, and L.G. Valiant. On time versus space. *JACM*, 24:332–337, 1977.

[18] N. Immerman. Nondeterministic space is closed under complement. *SIAM J. Comput.*, 17:935–938, 1988.

[19] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[20] K. Kobayashi. On proving time constructibility of functions. *Theoretical Computer Science*, 35:215–225, 1985.

[21] J. Köbler, U. Schöning, and J. Tóran. On counting and approximation. *Acta Informatica*, 26:363–379, 1989.

[22] M. Krentel. The complexity of optimization problems. *J. Computer Systems Sci.*, 36:490–509, 1988.

[23] P.M. Lewis, R.E. Stearns, and J. Hartmanis. Memory bounds for recognition of context-free and contex sensitive languages. In *Proc. 6th Annual IEEE Sym. on Switching Circuit Theory and Logical Design*, pages 191–202, 1965.

[24] Van Lint. *Introduction to coding theory.* Springer-Verlag, 1992.

[25] L. Longpre. On the linear time hierarchy. Technical Report NU-CCS-93-05, Northeastern University, 1993.

[26] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1973.

[27] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proc. 22nd Annual ACM Symposium on Theory of Computing.* ACM Press, 1990.

[28] N. Nisan and A. Wigderson. Hardness vs. randomness. In *Proc. 28th Annual IEEE Symp. on Foundations of Computer Science*, pages 2–11, 1988.

[29] M. Ogiwara and L.A. Hemachandra. A complexity theory for feasible closure properties. In *Proceedings of 6th Annual Structure in Complexity Theory Conference*, pages 16–29. IEEE Computer Society Press, 1991.

[30] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proc. 6th GI Conference on Theoretical Computer Science*, pages 269–276. Lecture Notes in Computer Science vol. 145, Springer-Verlag, 1983.

[31] W.J. Paul, N. Pippenger, E. Szemerédi, and W.T. Trotter. On determinism versus non-determinism and related problems. In *Proc. 24th ACM Symp. on Foundations of Computer Science*, pages 429–438, 1983.

[32] W.J. Paul, E.J. Prauss, and R. Reischuk. On alternation. *Acta Informatica*, 14:243–255, 1980.

[33] W.J. Paul and R. Reischuk. On alternation II. *Acta Informatica*, 14:391–403, 1980.

[34] A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Mat. Zametki*, 41:598–607, 1987. An English translation appears in: Mathematical Notes of the Academy of Sciences of the USSR 41(1987) 333-338.

[35] K. Regan. Enumeration problems. 1982.

[36] K. Regan. Error-correcting codes, probabilistic polynomials and circuits. 1993.

[37] K. Regan and J.S. Royer. A simpler proof of $\mathrm{PH} \subseteq \mathrm{BP}[\oplus\mathrm{P}]$. 1990.

[38] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 480:238–250, 1989.

[39] J. Simon. *On Some Central Problems in Computational Complexity.* PhD thesis, Cornell University, 1975. Available as Cornell Department of Computer Science Technical Report, TR 75-224.

[40] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 77–82, 1987.

[41] R.E. Stearns, J. Hartmanis, and P.M. Lewis. Hierarchies of memory-limited computations. In *Conference Record on Switching Circuit Theory and Logical Design*, pages 179–190, 1965.

[42] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[43] J. Tarui. Randomized polynomials, threshold circuits, and the polynomial hierarchy. In *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science*, pages 238–250, 1991. Lecture Notes in Computer Science, vol. 480, Springer-Verlag.

[44] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20:514–519, 1989. An earlier version appeared as: S. Toda, On the computational power of PP and $\oplus P$, in: Proceedings of 30th IEEE Symp. on Foundations of Computer Science, 1989.

[45] S. Toda. The complexity of finding medians. In *Proceedings of 31st IEEE Symp. on Foundations of Computer Science*, pages 778–787, 1990.

[46] S. Toda and M. Ogiwara. Counting classes are at least as hard as the the polynomial-time hierarchy. *SIAM J. Comput.*, 21:316–328, 1992. An earlier version appeared in: Proceedings of 6th Structure in Complexity Theory Conference, (IEEE Computer Society Press) 2-12.

[47] J. Torán. An oracle characterization of the counting hierarchy. In *Proceedings of 3rd Structure in Complexity Theory Conference*, pages 213–223. IEEE Computer Society Press, 1988.

[48] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[49] K. Wagner. The complexity of combinatorial problems with succint input representation. *Acta Informatica*, 23:325–356, 1986.

[50] J. Wang and L. Longpre. Nondeterministic and alternating computations. Personal Communication, 1992.

[51] A. Yao. On ACC and threshold circuits. In *Proc. 31st IEEE Symposium on Foundations of Computer Science*, pages 619–627, 1990.

[52] S. Zák. A turing machine hierarchy. *Theoretical Computer Science*, 26:327–333, 1983.